Neural Sequence-to-Sequence Modeling for Language and Speech Translation

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften genehmigte Dissertation

vorgelegt von

M.Sc.
Parnia Bahar
aus Teheran, Iran

Berichter:	Univ.-Prof. Dr.-Ing. Hermann Ney
	Professor François Yvon
	Univ.-Prof. Dr. rer. pol. Stefan Decker

Tag der mündlichen Prüfung: 24. November 2022

Eidesstattliche Erklärung

Ich, Parnia Bahar,
erkläre hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind und selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden.

Hiermit erkläre ich an Eides statt

1. Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand dieser Fakultät und Universität angefertigt;
2. Sofern irgendein Bestandteil dieser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt;
3. Wenn immer andere eigene- oder Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt;
4. Wenn aus anderen eigenen- oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben. Diese Dissertation ist vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate;
5. Alle wesentlichen Quellen von Unterstützung wurden benannt;
6. Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde;
7. Teile dieser Arbeit wurden zuvor veröffentlicht (Details in Kapitel 1.6.1 und 8).

Aachen, den 24 November 2022

Parnia Bahar

To my father.

Acknowledgments

I would like to express my sincere gratitude to many people who supported me in one way or another during these years as a Ph.D. student.

First, I would like to thank my supervisor, Prof. Dr.-Ing. Hermann Ney, for giving me the great opportunity to work and do research at the Chair of Computer Science 6 of RWTH Aachen University. His great support, invaluable supervision, and precious suggestions truly opened new doors towards a clear and scientific way of thinking, and his immense knowledge inspired the development of new ideas during the course of my Ph.D. degree.

I also would like to thank Prof. François Yvon and Prof. Dr. rer. pol. Stefan Decker for being on my thesis committee and taking the time to give insightful comments and feedback.

My special thanks to Dr. Ralf Schlüter for sharing his ideas and insights for automatic speech recognition and being open to discussions on the topic.

Further, I am deeply grateful to all members of the institute for the cherished time spent together during my studies. It was a pleasure working with you, and your assistance and teamwork at every stage of the research contributed to making a fun Ph.D. journey for me. My thanks go to my colleagues Albert, Andreas, Christian D., Christian H., Christoph, David, Eugen, Harald, Jörn, Jan P., Jan R., Julian, Kazuki, Malte, Markus K., Mohammad, Nick, Oskar, Patrick, Pavel, Peter, Stephan, Tamer, Tobias, Volker, Wei, Weiyue, Wilfried, Yingbo, Yunsu, Zoltán, to our system administrators Stefan and Thomas, and to our secretaries Stephanie, Dhenya and Anna.

My thanks also extend to the machine translation team for their fruitful conversations on the topic. Thank you Andreas, Christian H., David, Jörn, Jan P., Jan R., Julian, Malte, Stephan, Tamer, Weiyue, Yingbo, and Yunsu. Additionally, I would acknowledge Albert for his great guidance on RETURNN implementation and deep discussions on speech recognition, and Evgeny from AppTek for the great discussion and plentiful experience on speech translation.

I also appreciate all the support I received from supervising talented students during their Bachelor's and Master's studies. In particular, Christopher, Nikita, and Tobias significantly influenced the direction of this dissertation.

I would like to thank the following people for their insightful comments and suggestions in proofreading my dissertation despite their tight schedules: Christian H., David, Jan R., Mohammad, Nick, Weiyue, and Yingbo.

Finally and most importantly, my acknowledgments would not be complete without giving thanks to my friends and family for their encouraging support, especially my mother, brother, and grandmother. Your patient, loving support through all stages of life, and tremendous belief in me over these years were the main source of confidence and motivation to complete this thesis. I owe this work first and foremost to you.

Abstract

In recent years, various fields in human language technology have been advanced by the success of neural sequence-to-sequence modeling. The application of attention models to automatic speech recognition, text and speech machine translation has become dominant and well-established. Although the effectiveness of such models has been documented in scientific papers, not all aspects of attention sequence-to-sequence models have been explored, and some essential concepts are still missing. Therefore, the main contribution of this thesis centers around redesigning attention models by proposing novel alternative models in terms of architecture and mathematical formulation for language technology.

As attention models do not make any conditional dependence assumption on previous attention information, inspired by statistical word alignments, this work first extends recurrent attention models by implicitly including more alignment information from previous output positions.

Furthermore, from a modeling perspective, this research goes beyond current sequence-to-sequence backbone models to directly incorporate input and output sequences in a 2D structure where an attention mechanism is no longer required. This model distinguishes itself from attention models in which inputs and outputs are treated as one-dimensional sequences over time and then combined with an attention mechanism. In contrast to attention models which do not reinterpret encoder states while decoding, the proposed model enhances the degree of variance in context vectors by refining input representations to be sensitive to the partial translation.

Current state-of-the-art attention models also lack an explicit alignment, a core component of traditional systems. Instead, their attention mechanism may be considered to produce an implicit alignment. Such a gross simplification of a complex process complicates the extraction of alignments between input and output positions. To enable attention models to be explainable and their output to be better controlled, the next part of this study integrates the attention model into the hidden Markov model formulation by introducing alignments as a sequence of hidden variables. Since marginalization has an exponential number of terms in the alignment dependency order of the model, a zero-order assumption that is simpler and more efficient is explored.

Finally, an exciting research direction is to combine speech recognition with text machine translation for speech-to-text translation. Besides advancing a cascade of independently trained speech recognition and machine translation systems, this thesis sheds light on multiple end-to-end models to directly translate speech inputs to target texts. In this context, promising methods are borrowed from speech recognition, and best practices are established for direct modeling. Addressing and revisiting already proposed methods in the literature, the last part of this study investigates and develops new approaches to leverage all types of available training data, i.e., speech-to-source, source-to-target, and speech-to-target text data. Ultimately, it is shown that end-to-end models can practically translate speech utterances as a substitute solution to cascaded speech translation.

Kurzfassung

In den letzten Jahren wurden verschiedene Bereiche menschlicher Sprachtechnologie durch den Erfolg neuronaler Sequenz-zu-Sequenz-Modellierung vorangebracht. Die Anwendung von Attention-Modellen in der automatischen Spracherkennung sowie in der maschinellen Übersetzung von geschriebener und gesprochener Sprache hat sich inzwischen etabliert. Obwohl die Effektivität solcher Modelle in wissenschaftlichen Arbeiten dokumentiert wurde, sind noch nicht alle Aspekte von Sequenz-zu-Sequenz-Modellen erforscht worden, die auf Attention-Modellen basieren, und einige wesentliche Konzepte fehlen noch. Daher konzentriert sich diese Arbeit auf die Umgestaltung von Attention-Modellen, indem neue Alternativen für die Sprachtechnologie vorgeschlagen werden.

Attention-Modelle haben keine bedingten Abhängigkeiten zu früheren Attention-Informationen. Inspiriert von statistischen Wortalignierungen erweitert diese Arbeit zunächst das rekurrente Attention-Modell durch die implizite Einbeziehung weiterer Alignierungsinformationen aus früheren Ausgabepositionen.

Desweiteren geht diese Arbeit hinsichtlich der Modellierung über die derzeitigen Sequenz-zu-Sequenz-Modelle hinaus, indem Eingabe- und Ausgabesequenzen direkt in eine 2D-Struktur einbezogen werden, für die ein Attention-Mechanismus nicht mehr erforderlich ist. Dieses Modell unterscheidet sich von Attention-Modellen, bei denen die Eingabe und Ausgabe als eindimensionale Sequenzen über die Zeit behandelt und dann mit einem Attention-Mechanismus kombiniert werden. Im Gegensatz zu Attention-Modellen, welche die Encoder-Zustände während der Dekodierung nicht neu interpretieren, erhöht das vorgeschlagene Modell die Expressivität der Kontextvektoren, indem es die Eingangsrepräsentationen so verändert, dass sie von der partiellen Übersetzung abhängen.

Den aktuellen Attention-Modellen fehlt auch eine explizite Alignierung, eine Kernkomponente traditioneller Systeme. Stattdessen verfügen sie über einen effektiven Attention-Mechanismus, der als eine implizierte Form der Alignierung betrachtet werden kann. Eine solche starke Vereinfachung eines komplexen Prozesses erschwert die Extraktion von Alignierungen zwischen Eingabe- und Ausgabepositionen. Um die Erklärbarkeit von Attention-Modellen zu verbessern und eine besser kontrollierbare Ausgabe zu ermöglichen, wird im nächsten Teil dieser Arbeit das Attention-Modell in die Formulierung des Hidden Markov Modelles integriert, indem Alignierungen als eine Folge von verborgenen Variablen eingeführt werden. Da die Marginalisierung in der Ordnung der Alignierungsabhängigkeit exponentiell ist, wird eine einfachere und effizientere Approximation, in der keine Abhängigkeit angenommen wird, erforscht.

Eine interessante Forschungsrichtung ist schließlich die Kombination von automatischer Spracherkennung und Textübersetzung hin zu Sprache-zu-Text-Übersetzung. Neben der Aneinanderreihung von unabhängig voneinander trainierten Spracherkennungs- und maschinellen Übersetzungssystemen beleuchtet diese Arbeit verschiedene End-to-End-Modelle zur direkten Übersetzung von gesprochener Sprache in einen Text. In diesem Zusammenhang werden vielversprechende Methoden aus der Spracherkennung aufgegriffen und erfolgreiche Verfahren für die direkte Modellierung etabliert. Der letzte Teil dieser Studie untersucht und entwickelt neue Ansätze zur Nutzung verfügbarer Trainingsdaten, wobei bereits in der Literatur vorgeschlagene Methoden aufgegriffen werden. Letztlich wird gezeigt, dass End-to-End-Modelle eine praxistaugliche Alternative zur Kombination von Spracherkennung und maschineller Übersetzung sein können.

Contents

1. Introduction

Spoken and written languages are regarded as one of the main features of human intelligence and a natural means of communication between people. Developing intelligent systems that can understand and generate human language is important to the science and engineering domains and has become an objective, particularly in human language technology. It is also necessary for computers to achieve their full potential in augmenting human intelligence. Early methods towards this goal have attempted to simulate language capacity through predefined rules that are too restrictive to capture the complexity of human language. But such manually-created rules require human language experts; consequently, their creation is expensive and time-consuming.

Seen through the lens of statistical approaches [Manning & Schütze 99], one can use mathematical models to automatically learn rules from data, often referred to as data-driven methods. These methods rely on learning from hundreds of thousands to millions of examples and enable the development of language-independent techniques and the treatment of uncertainty. Over the last decades, deep neural networks as a subcategory of the powerful machine learning field have had remarkable success on a wide range of tasks. The standard way to learn a function that maps an input sequence to an output sequence is to have supervision where a large number of annotated examples are provided for training. The term supervised sequence modeling includes most tasks where input sequences of data are aligned with output sequences of discrete labels. The main difference from the standard supervised classification is that both inputs and outputs form a strong correlation in sequence modeling. The underlying assumption here is that the individual positions in sequences cannot be assumed to be independent. Since the alignment between inputs and outputs is unknown in many cases, this requires algorithms that adequately define the correspondence between the positions of these sequences.

This work studies neural-based sequence modeling for language technology in particular well-known applications of machine translation (MT), automatic speech recognition (ASR), and coupling the two, as speech-to-text translation or simply speech translation (ST). In common terminology, we refer to machine translation as the translation of languages or texts. To be precise, speech denotes audio signals containing spoken utterances, while a spoken language refers to the verbatim text form of the spoken utterances. Written language is the contrasting case of text originally composed as text without a spoken utterance as its basis. After a short introduction of terminologies and notations in this chapter, we review some fundamental concepts for a better understanding of this work. We will only give a short overview and refer to the corresponding literature for more details. The concepts described in this chapter are not part of our contributions and are included for the sake of completeness.

1.1 Terminology and Notation

Given an input sequence, the gold standard output is called a reference. The output of a model is referred to as a hypothesis. A collection of text/audio data is referred to as a corpus that is split into three sets: training data, which is a large portion used to train models, one or several smaller development sets (dev), and unseen test sets, the latter two of which are respectively utilized for parameter tuning and final evaluation. To build meaningful and generalizable models, all three sets should have a similar domain but should be disjointed and contain as few duplicates as possible. In practice, most training data are generated automatically or mixed with manually-created data. Dev and test sets, on the other hand, are often human-created for this particular purpose.

Making extensive use of probability theory, we use $Pr(\cdot)$ to refer to the true distribution, being unknown in practice. Instead, we estimate $Pr(\cdot)$ with another distribution, the model distribution denoted as $p(\cdot)$. The distribution $p(\cdot)$ is usually described by a statistical model that determines its parameters based on a large corpus. This procedure is called training. The exact algorithm depends on how $p(\cdot)$ is modeled, but the general process regarding the data is the same. The model is first trained to optimize a predefined training criterion using training data, with the best hyperparameters determined according to the model performance on the development set. The algorithm performing the search for the best hypothesis given the underlying model is interchangeably referred to as decoding, inference, or search. Finally, to judge the performance of a model compared to other models trained on the same data, they are evaluated based on the same test set. According to this, we need four main ingredients [Ney 19] for a data-driven method:

- probabilistic models with suitable structures to capture the dependencies within and between input and output sequences and to replace the true distribution
- training criterion to estimate the free model parameters from the training data
- search algorithm to generate the output sequence
- performance measure to judge the quality of a system's output

Throughout this work, the input and output sequences differ for different tasks. In machine translation, both input and output sequences contain discrete text information, whereas in automatic speech recognition and translation, the input is a sequence of continuous speech signals in a waveform, and the outputs are sequences of strings, i.e., a text. In machine translation, both input and output sequences are usually sentence-like, whereas outputs can correspond to a segment of a sentence in speech applications. In almost all languages, a text is composed of sequences of words usually accompanied by punctuation. The words can be split into smaller pieces like subwords, characters, and phonemes. To be general enough, we employ the term *token* to refer to all components of a sentence. Considering this, we use token and word interchangeably, likewise sentence and sequence.

1.2 Machine Translation

Machine translation is the task of automatically translating an arbitrary text written in one language, called the source language, into written text in the target language using machines[1]. The human translation seems impossible when considering thousands of existing natural languages that can be translated into one another. In contrast, machine translation fulfills these requirements as it automates the entire process of translation. In general, the translation task can be broken down into two steps. First, the meaning of the source text must be decoded, and then that meaning must be re-encoded into the target text. The task is more complicated than initially observed, mainly due to languages containing an almost immeasurable number of rules, meanings, grammar,

[1]Throughout this work, the term machine translation always refers to machine translation of written texts. For machine translation of speech, we use the term speech-to-text translation or simply speech translation.

and other linguistic properties. The grammar rules often used to describe languages are rarely adhered to in practice, especially when looking at spoken or colloquial contexts. Furthermore, translation itself involves the reordering of words and phrases and producing syntactically and semantically correct outputs [Manning & Schütze 99]. Hence, machine translation is a challenging task and has a wide range of applications, each of which requires a different speed and quality.

Besides the high demand for translating texts like articles and books, accessing information that is available in foreign languages is a de-facto need. A machine translation system does not necessarily have to generate a grand-goal high-quality translation, but it can also be used to create a draft translation. Some examples are assimilation, dissemination, and communication. In some scenarios, a post-editor or a human-computer assistant system is required to improve the final output [Koehn 10].

There are several perspectives on the nature of translation. One method is the transfer system in which one modifies the structure of the source sentence to conform to the rules of the target language. This is obviously a suboptimal solution for the translation process as it requires a set of transfer rules, syntactically, semantically, or lexically defined. An alternative is to extract the meaning of source sentences, obtain a knowledge-based representation, and generate the corresponding target sentences. This process is known as the interlingual approach. One implication for such methods is the need to exhaustively analyze the semantic relations between words; however, an interlingual system has to disambiguate. Another way is a direct mapping between the words of the source sentence and those of the target sentence, which is direct translation. In this case, the syntactic and semantic of languages are not explicitly considered [Koehn 10].

Since translation involves many decisions that are hard to formalize, it is preferable to employ data-driven approaches, particularly statistical methods. In principle, all of the mentioned methods can be implemented based on probabilistic models and viewed as a mathematical optimization problem. Statistical machine translation employs large corpora to infer the parameters of a statistical model. This model then translates the source sentence into the target sentence by finding the hypothesis with the maximum likelihood given the input data. Statistical methods generally lack the interpretability of rule-based approaches but have been highly successful in practical applications. The methods are language-independent, meaning that the same model structure can be applied to several language pairs. Building on top of statistical methods, neural machine translation approaches use neural networks as statistical tools. These approaches often train all the components of the system together to maximize translation performance and are currently considered to be state-of-the-art.

To be formal, in machine translation, $f_1^J = f_1, \ldots, f_j, \ldots, f_J$ denotes an arbitrary input sentence of source tokens of length J, and $e_1^I = e_1, \ldots, e_i, \ldots, e_I$ represents an arbitrary target token sentence whose length I is unknown. Historically, f stands for French or foreign and e for English. We note that in our notation, the letters with both subscript and superscript stand for sequences of symbols, and lower-case letters for individual symbols. Bayes' decision rule [Bayes 63] in its standard form, i.e., assuming uniform sequence loss, requires maximization of the output or class posterior given an input observation. According to Bayes' decision rule, given a source sentence, we find the best translation candidate with maximum posterior probability given by

$$
\begin{aligned}
f_1^J \to \hat{e}_1^{\hat{I}}(f_1^J) &= \underset{I, e_1^I}{\operatorname{argmax}} \left\{ Pr(e_1^I | f_1^J) \right\} && (1.1) \\
&= \underset{I, e_1^I}{\operatorname{argmax}} \left\{ \frac{Pr(f_1^J | e_1^I) Pr(e_1^I)}{Pr(f_1^J)} \right\} && (1.2) \\
&= \underset{I, e_1^I}{\operatorname{argmax}} \left\{ Pr(f_1^J | e_1^I) Pr(e_1^I) \right\}. && (1.3)
\end{aligned}
$$

Using Bayes' theorem and ignoring terms independent of the target sentence, it has been common to decompose the posterior probability and derive Equation 1.3 with the generative class-conditional distribution, which is the so-called inverse translation model $Pr(f_1^J|e_1^I)$, and the target sentence prior $Pr(e_1^I)$, the so-called language model discussed later in Section 1.3.2. The former requires annotated parallel bilingual data, while the latter can be modeled using large amounts of monolingual data. Besides the data condition, this decomposition arguably captures different aspects of the transduction problem, thus has complementary strengths.

Therefore, in order to implement Bayes' decision rule in this form, we need to model the posterior probability of the source sentence and the probability of the target sentence, and perform efficient maximization over all possible target sentences which are candidates for the corresponding source sentence. In practice, computing argmax is a considerable computational challenge that needs crucial restrictions for tractability. All of these aspects determine the quality of a system based on a performance measure.

1.2.1 Word Alignment

The first statistical translation models rely on words, where the product of the probabilities of every word represents the probability of a sentence. *Word alignments* play an important role in estimating translation models as conditional probabilities between the source and target words. They are also crucial for phrase-based machine translation, which is addressed next. Thus, the first step is to create word alignments in a bilingual data setup. Given a source sentence and its translation in the target language, a word alignment $A \subseteq I \times J$ is used to capture complex word-by-word correspondences between two sentences from which we define a relation over the source position j and the target position i. In principle, this relation does not have any restriction, but it is formed as a mapping for the sake of computational simplicity. In this concept, each source word f_j is mapped to only one target word e_i.

A sequence of alignments is denoted as a sequence of latent variables $a_1^J = a_1, \ldots, a_j, \ldots, a_J$ that establishes a mapping from the source position j to the target position i, i.e., $j \rightarrow a_j = i$. This refers to the position i of e_i, with which f_j is aligned. The posterior translation probability of the source sentence f_1^J given the target sentence e_1^I using the alignment is defined as

$$Pr(f_1^J|e_1^I) = \sum_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \tag{1.4}$$

$$= \underbrace{p(J|e_1^I)}_{\text{length model}} \sum_{a_1^J} \prod_{j=1}^{J} \underbrace{p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I, J)}_{\text{alignment model}} \underbrace{p(f_j|f_1^{j-1}, a_1^j, e_1^I, J)}_{\text{lexicon model}}. \tag{1.5}$$

The inverse translation posterior is decomposed into three distinct components: the *length*, *alignment*, and *lexicon* models. This establishes the well-known count-based models such as IBM [Brown & Cocke+ 88, Brown & Cocke+ 90, Brown & Pietra+ 93] and the hidden Markov model (HMM) [Vogel & Ney+ 96]. These models mainly share the same length and lexicon models, while they differ in their dependency assumptions of the alignment probability [Ney 19].

The length model is considered to be either stored in a table (model-free approach) or modeled with the Poisson distribution. Both cases only assume a dependency on length I of target sentence e_1^I, meaning $p(J|e_1^I) = p(J|I)$.

The lexicon model makes a zero-order dependency on e_i in the position $i = a_j$. Hence, it can be rewritten as $p(f_j|f_1^{j-1}, a_1^j, e_1^I, J) = p(f_j|e_{a_j})$.

In terms of the alignment model, these models have different assumptions as listed below:

IBM-1 considers a naïve way of alignments with a uniform distribution given by

$$p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I, J) = \frac{1}{I+1}. \tag{1.6}$$

To maximize the log-likelihood of the model, the expectation-maximization (EM) algorithm [Dempster & Laird$^+$ 77] is used based on the relative frequency counts of the source and target words seen in the training data. IBM-1 training is not strictly a convex optimization problem; however, finding a good optimum can be expected.

IBM-2 is a zero-order model and assumes a dependency on the absolute position j, i.e.,

$$p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I, J) = p(a_j|j, J, I). \tag{1.7}$$

Since the training of IBM-2 has no global optimum, the IBM-1 model can be used to initialize its parameters.

HMM makes a first-order Markov assumption on the immediate predecessor position as

$$p(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I, J) = p(a_j|a_{j-1}, J, I). \tag{1.8}$$

This implies that the current alignment depends only on the alignment from the last step, rather than on other steps before that or on higher-order dependencies. It shows the fact that words should be considered together as a group in translation. Contrary to the IBM-1 and IBM-2 models, the sum over the alignments is performed inside the product in HMM. This leads to an exponential number of terms which can be effectively calculated using dynamic programming [Bellman 54] by dividing the problem into incremental steps. For faster computation, a maximum approximation called the Viterbi approximation is often used [Viterbi 67] in practice. Such an approximation justifies that a single position dominates the distribution, and it is written as

$$\sum_{a_1^J} \prod_{j=1}^{J} p(a_j|a_{j-1}, J, I) p(f_j|e_{a_j}) \approx \max_{a_1^J} \prod_{j=1}^{J} p(a_j|a_{j-1}, J, I) p(f_j|e_{a_j}). \tag{1.9}$$

There are some extensions to the above models named IBM-3 through IBM-5.

IBM-3 is a reparameterization of IBM-2 using fertility $\varphi(e_i)$. Fertility indicates the number of source words that have to be translated to the given target word e_i. Note that the source length J directly depends on fertility, i.e, $J = \sum_i \varphi(e_i)$. The model first assigns a fertility probability to each target word. Once the target words are translated, the generated source words are reordered to form the final sentence.

IBM-4 is an inverted alignment model equipped with word classes. It also determines the reordering probabilities based relatively on previously translated words.

IBM-5 satisfies the normalization constraints leading to the non-deficient version of IBM-4, where probability mass is assigned to impossible translations as well as where multiple source words are placed in the same position. The model does this by tracking vacant positions across the translation.

In practice, the simpler models are used to initialize the complex ones. A typical setup is to employ IBM-1, then HMM followed by IBM-4, each trained for five EM iterations. The alignments are generated in both translation directions and then combined by the grow-diagonal-final-and heuristic [Koehn & Och$^+$ 03], resulting in general undirected alignment. Reasonably high-quality alignments can be computed with the GIZA++ toolkit [Och & Ney 03].

Although the word-level models are equipped with more complex translation concepts in later variations, they have limited within-word context since translation is generated word by word. Even though, the decoding procedure applies an external language model to offer lexical context beyond word boundaries.

1.2.2 Phrase-based Translation

In order to overcome the limited context issue, phrase-based machine translation has been proposed by [Zens & Och$^+$ 02, Koehn & Och$^+$ 03]. As the name suggests, phrase-level models work on contiguous sequences of words on both source and target sides, thus contain a larger local context. They differ inherently from word-level models as the whole phrase probability score is taken into account at each translation step. Hence, such modeling seems more successful in the translation of idioms.

The core idea of the phrase-based system is phrase extraction, in that a sentence pair (f_1^J, e_1^I) is segmented, and contiguous phrase pairs are obtained as an atomic unit of translation. Some constraints need to be met for a valid segmentation. All source and target words should be covered by exactly one phrase, i.e., no uncovered words, gaps, and overlapping phrases. The extracted valid bilingual phrase pairs are stored in a phrase table. The size of this table can grow exponentially. In practice, a maximum for phrase length is specified, and the table is often pruned. The source phrases are transferred into the target language by a look-up in the phrase table, and then the target phrases can be reordered to produce fluent translation if needed. The phrase-based machine translation system defines several models and applies the log-linear framework to include different translation modeling aspects. Next, we briefly describe log-linear modeling as a key concept of traditional translation systems. Further details of phrase-based translation are not discussed in this thesis. Descriptions of its main components can be found in [Koehn & Och+ 03, Zens 08].

Log-Linear Modeling

In order to directly represent the posterior probability $Pr(e_1^I|f_1^J)$, the log-linear model combination [Och & Ney 02] is given by

$$Pr(e_1^I|f_1^J) = \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)\right)}{\sum_{I', {e'}_1^{I'}} \exp\left(\sum_{m=1}^{M} \lambda_m h_m({e'}_1^{I'}, f_1^J)\right)}. \tag{1.10}$$

It allows an arbitrary number of M models $h_m(.,.)$ called feature functions. These different models usually capture various aspects of a problem, and each is multiplied with a corresponding scaling factor λ_m [Papineni & Roukos+ 98]. To differently weigh the models according to their importance for the final decision, the scaling factors are tuned on the development set separately, one at a time, while the others are fixed. Minimum error rate training [Och 03] is the most common algorithm for optimizing λ_m. During search, by dropping the denominator as being constant with respect to the optimization arguments, and with the exponential function being strictly monotonous, we derive a linear combination of models as

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I}{\operatorname{argmax}} \left\{Pr(e_1^I|f_1^J)\right\} = \underset{I, e_1^I}{\operatorname{argmax}} \left\{\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)\right\}. \tag{1.11}$$

The log-linear model is also widely used in neural approaches where we combine the scores of different models on the probability level. It is a common practice to integrate the acoustic and language model scores in automatic speech recognition, as is addressed next.

1.3 Automatic Speech Recognition

The goal of automatic speech recognition is to transcribe speech signals into a string of words. It is a technology with a long history of development and plays a vital role in many tasks where full language communication is needed. Some examples are human-machine interaction, dictation as part of augmented communication, telephony, voice search, etc. [Jurafsky & Martin 00]. It can become challenging to automatically transcribe the speech of any (multi) speaker in any environment (with or without background noise) and thus speech recognition involves many other signal processing methods. Firstly, successful techniques work with a viable quality in certain limited domains. Secondly, they are based on the fundamentals of speaker-independent large-vocabulary continuous speech recognition. Thirdly, the words are spoken together naturally, without intentional preceding or following pauses. Finally, systems are able to recognize the speech of a speaker that they have never been exposed to.

Given a speech signal, we obtain a time-discrete frame-wise sequence of observations $x_1^T = x_1, \ldots, x_t, \ldots, x_T$ of length T, where a feature extraction pipeline has to be performed. We aim to find the corresponding sequence of N transcribed words w_1^N. Following the decomposition in Bayes' decision rule in Equation 1.3, we rewrite

$$x_1^T \rightarrow \hat{w}_1^{\hat{N}}(x_1^T) = \operatorname*{argmax}_{N, w_1^N} \left\{ Pr(x_1^T | w_1^N) Pr(w_1^N) \right\}. \tag{1.12}$$

1.3.1 Acoustic Model

An *acoustic model* (AM) estimates conditional posterior probability $Pr(x_1^T | w_1^N)$. The majority of state-of-the-art acoustic models follow the statistical hidden Markov model in which a sequence of HMM states represents a sequence of words [Baum & Eagon 67, Rabiner 89]. Based on the fact that it is impossible to cover all existing words of a natural language in the training data, words are converted to smaller units, called phonemes, using a pronunciation lexicon. Phonemes are defined as the smallest phonetic elements capable of changing the meaning of a word.

Acoustic models can be context-dependent or independent based on the models' dependency on other adjacent phonemes. There, each HMM state represents the defined unit like a triphone, where we take both left and right context into account. The HMM models introduce a sequence of latent stochastic variables or state sequence s_1^T, as alignments relating each input frame x_t to some states. Formally, we have

$$p(x_1^T | w_1^N) = \sum_{s_1^T} p(x_1^T, s_1^T | w_1^N) \tag{1.13}$$

$$\approx \sum_{s_1^T} \prod_{t=1}^{T} \underbrace{p(x_t | s_t, w_1^N)}_{\text{emission}} \underbrace{p(s_t | s_{t-1}, w_1^N)}_{\text{transition}}. \tag{1.14}$$

Assuming a first-order Markov process and simplified dependence assumption on the previous frames, we obtain Equation 1.14 with the *emission* and *transition* models. The emission probabilities are applied whenever a state is visited. More details can be found in [Rabiner 89, Rabiner & Juang 93]. The emission probabilities of HMM states can be formed by the continuous Gaussian mixture model (GMM) and its extension to use neural networks called the Tandem approach [Hermansky & Ellis$^+$ 00]. The transition probability is used if the path moves from one state to another.

Similar to many applications, speech recognition is also affected by deep neural networks (DNNs) in acoustic modeling. A Hybrid DNN/HMM approach that incorporates neural networks into HMM-based models to improve the performance of the emission probability using feedforward [Waibel & Hanazawa$^+$ 88, Bourlard & Wellekens 90, Baker 90, Robinson 94] and recurrent neural networks [Schuster & Paliwal 97, Graves & Jaitly$^+$ 13a] have become the de-facto approach. Considering x_1^T and s_1^T as the input and output sequences, neural networks are built for state posterior estimation, i.e., $p(s_t | x_t)$. For an in-depth discussion, we refer to [Bourlard & Morgan 93].

Once all parameters have been learned from the data during training, the search for the best word sequence corresponding to maximizing the posterior probability is performed. Due to noisy channel decomposition, models deal with complicated search algorithms that impose limitations concerning tractability. Hence, it is common to approximate the sum in Equation 1.14 using the Viterbi approximation and to perform dynamic programming.

1.3.2 Language Model

A statistical *language model* (LM) provides a probability distribution that assigns to each word sequence of a language the probability that exactly this sequence is to be represented. As stated in Bayes' decision in Equation 1.12, the prior $Pr(w_1^N)$ is the language model. Given the sequence of words w_1^N, the model probability is expressed using the chain rule as

$$p(w_1^N) = \prod_{k=1}^{N} p(w_k|w_1^{k-1}) = \prod_{k=1}^{N} p(w_k|w_{k-n+1}^{k-1}). \tag{1.15}$$

A method to estimate probabilities lies in counting how often a specific word sequence occurs in the training corpora, known as the *count-based* model. Of course, most long-word sequences might not even occur once. Therefore, the standard *n-gram* language model breaks down the probability estimation into smaller parts for which we can estimate the probability. Taking into account the Markov assumption of having a limited number of previous words, we reformulate the language model to the n-gram language model given a fixed $n \in \mathbb{N}$. Thus, the overall probability estimation is simplified to the last $n-1$ words, referred to as *history.*

The biggest problem associated with the n-gram model is that it gives a zero probability to unseen word sequences according to Equation 1.15. No matter how large the training dataset, it is unlikely to contain all feasible sequences. This is where the problem of data sparsity arises. To retain probability mass for unseen events, the so-called smoothing technique is used in which an unseen n-gram steps back to a smaller history [Kneser & Ney 95, Chen & Goodman 99]. The neural language models discussed briefly in Section 3.3 solve this problem.

The most common way to evaluate a language model is perplexity (Ppl), defined as the geometric mean of the inverse probability and formulated by

$$\text{Ppl} = p(w_1^N)^{-\frac{1}{N}} = \exp\big(-\frac{1}{N}\sum_{n=1}^{N}\log p(w_n|w_0^{n-1})\big) \tag{1.16}$$

where w_0 denotes the artificial token at the first position. A good language model assigns a higher probability, then lower perplexity to the more likely words. The exponent yields the cross-entropy criterion described in more detail in Section 3.6.

1.4 Speech Translation

An exciting direction for machine translation is its integration with speech recognition to address an outstanding research goal, speech-to-text translation or simply speech translation. Speech translation aims to convert speech signals to written texts in foreign languages, the target texts. The task combines several challenging aspects associated with both recognition and translation. It entails bridging the gap between spoken language being informal with disfluencies and the more formal and grammatically correct written domain. It opens up diverse applications ranging from translating dialog-based telephone conversations in limited domains, parliament speeches, or audio broadcasts to accessing information and interpersonal communication. For speech-to-speech translation systems, a speech generator is added as the final step to generate a continuous speech waveform as the output. In this thesis, we do not address this last step related to speech generation.

1.4.1 Cascade Model

Borrowing the notation of machine translation and speech recognition, we define three sequences with different variable lengths: the sequence of frame-wise acoustic vectors x_1^T, the sequence of transcripts in the source language f_1^J, and the target language sequence e_1^I. We then rewrite the Bayes' decision rule that maximizes the posterior probability of the target sentence given the speech input and derive

$$x_1^T \rightarrow \hat{e}_1^{\hat{I}}(x_1^T) = \underset{I,e_1^I}{\operatorname{argmax}} \left\{ Pr(e_1^I|x_1^T) \right\}. \tag{1.17}$$

Equation 1.17 implies *direct modeling* (see Chapter 6) where we search over all possible target sequences that are candidates for the given speech utterance. We can further introduce the source sequence f_1^J as a hidden variable into the conditional probability and marginalize over all possible source sequences to obtain [Ney 99]

$$\begin{aligned} x_1^T \rightarrow \hat{e}_1^{\hat{I}}(x_1^T) &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \sum_{f_1^J} Pr(e_1^I, f_1^J|x_1^T) \right\} && (1.18) \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \sum_{f_1^J} Pr(e_1^I|f_1^J, x_1^T) Pr(f_1^J|x_1^T) \right\} && (1.19) \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \sum_{f_1^J} Pr(e_1^I|f_1^J) Pr(f_1^J|x_1^T) \right\} && (1.20) \\ &\approx \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \max_{f_1^J} Pr(e_1^I|f_1^J) Pr(f_1^J|x_1^T) \right\}. && (1.21) \end{aligned}$$

$Pr(e_1^I, f_1^J|x_1^T)$ can be computed using the log-linear framework. By further decomposition, we yield two distributions in Equation 1.19. Here, we can make a natural assumption that the target sentence has a strong dependency on the source sentence but no dependency on the speech signal. Then we derive Equation 1.20. According to this equation, the decision is made based on the translation model $Pr(e_1^I|f_1^J)$ and the acoustic model $Pr(f_1^J|x_1^T)$. We can also approximate the sum over all possible source sequences with the maximum. Therefore, Equation 1.20 is reformulated into Equation 1.21.

Due to the exponential search space of the source language, the decision process is usually divided into two steps. The first step transcribes the speech signal into the source sequence f_1^J using an automatic speech recognition model by

$$\hat{f}_1^{\hat{J}} = \underset{J,f_1^J}{\operatorname{argmax}} \left\{ p(f_1^J|x_1^T) \right\}. \tag{1.22}$$

Then, given the predicted source sentence, a target sentence is generated through a machine translation model written as

$$\hat{e}_1^{\hat{I}} = \underset{I,e_1^I}{\operatorname{argmax}} \left\{ p(e_1^I|\hat{f}_1^{\hat{J}}) \right\}. \tag{1.23}$$

This pipeline is referred to as a *cascade* system composed of two main statistical models. Any progress in speech recognition and machine translation can result in speech translation improvements.

In practice, a close-to-zero recognition error rate can hardly be expected, and recognition errors are propagated to the translation model because of early decisions. However, it is arguable that different representations of multiple hypotheses for a single acoustic speech utterance can be considered for translation to solve the early-decision problem [Matusov 09].

The first best ASR hypothesis can be passed on to the MT model. Although such a naïve approach has no additional computational overhead, this restricts the translation process to a hard decision made by the ASR system, which cannot be corrected afterward. A possible alternative to the single best hypothesis is to feed the list of N-best hypotheses to the MT model. Then, the best translation is selected according to the score of both models. In this case, instead of a hard decision, we give the MT model more chances of a better coupling with the ASR output. Obviously, this approach is not efficient unless we choose a small size for N. Based on the fact that most of the hypotheses only differ in a few tokens, we can barely get further translation improvements [Matusov & Kanthak$^+$ 05, Matusov 09].

The ASR output can also be an ambiguous output in the form of word lattices [Matusov & Hoffmeister$^+$ 08, Matusov 09, Sperber & Neubig$^+$ 17, Sperber & Neubig$^+$ 19b], and confusion networks [Matusov & Kanthak$^+$ 05, Bertoldi & Zens$^+$ 07]. Such outputs give the MT system many more word hypotheses than the N-best list representation. However, the MT model needs to be modified for a robust coupling. The word lattice is a directed acyclic graph that compactly represents the most likely hypotheses of the ASR first-pass decoding. Each arc stores acoustic and language model scores, and each node represents time stamps. One of the main benefits of a lattice structure is to provide an efficient yet effective way of representing the ASR search space. The confusion network is obtained from a word lattice but with a linear structure.

Regardless of recognizers' output representations, cascade systems encounter other problems associated with interfacing the ASR output and the MT input. Since the recognizer and translator systems are trained separately, the ASR output likely differs from what the MT model expects, given the data it has been trained on. To improve translation quality, we can also tighten the coupling between spoken-style ASR outputs and written-style MT inputs for reducing the mismatch between them. We can optionally carry out sentence segmentation and punctuation prediction [Matusov & Mauser$^+$ 06, Fügen & Kolss 07], disfluency removal [Fitzgerald & Hall$^+$ 09], and domain adaptation [Liu & Gu$^+$ 03] to have the recognizer's outputs as close as possible to the translator's input. Instead of augmenting the source transcriptions, we can also adapt the source side of the machine translation system to look like the output of the speech recognizer [Dixon & Finch$^+$ 11, Peitz & Wiesler$^+$ 12, Sperber & Neubig$^+$ 17, Gangi & Enyedi$^+$ 19]. However, all of these methods only reduce the mismatch between the two components and diminish the loosely coupled cascade models, rather than addressing the actual problem of error through early decisions.

1.5 Evaluation Measures

To compare several systems, their quality can be quantified by evaluation metrics. An automatic metric highly correlated with human evaluation is needed to evaluate the huge number of hypotheses produced by different systems such that each hypothesis is compared to a human-generated reference. In the following section, we briefly address four metrics used in the course of this thesis.

1.5.1 Word Error Rate

The word error rate (Wer) is a measure defined as the Levenshtein distance [Levenshtein 66] or edit distance divided by the number of words in the reference. The Levenshtein distance between a sequence and its corresponding reference is computed by the minimum number of edit operations, including deletions, insertions, and substitutions. All types of edits are assigned an equal cost.

Formally, we have

$$\textsc{Wer} = \frac{\text{number of edit operations}}{\text{total number of words in reference}}. \tag{1.24}$$

This metric is a commonly used measure in automatic speech recognition to determine the quality of recognized sequences compared to a human reference transcription.

1.5.2 Bilingual Evaluation Understudy

The bilingual evaluation understudy (Bleu) [Papineni & Roukos+ 02] is the most commonly used evaluation metric for machine translation. It is a precision measure that combines a modified n-gram precision with a brevity penalty (BP). Bleu works by counting how many n-grams from the hypothesis are the same as in the reference, computing a modified n-gram precision.

The precision between the hypothesis $\hat{e}_1^{\hat{I}}$ and the reference translation e_1^I is defined as

$$prec_n(e_1^I, \hat{e}_1^{\hat{I}}) = \frac{\sum\limits_{w_1^n} \min\left\{\text{count}(w_1^n|e_1^I), \text{count}(w_1^n|\hat{e}_1^{\hat{I}})\right\}}{\sum\limits_{w_1^n} \text{count}(w_1^n|\hat{e}_1^{\hat{I}})} \tag{1.25}$$

where w_1^n is the n-gram of n consequent words and $\text{count}(w_1^n|e_1^I)$ determines the number of times w_1^n appears in e_1^I. In Equation 1.25, the denominator is the total number of n-grams in the sentence. The brevity penalty to penalize short hypotheses is computed by

$$\text{BP}(I, \hat{I}) = \begin{cases} 1 & \text{if } \hat{I} \geq I \\ \exp\left(1 - \frac{I}{\hat{I}}\right) & \text{if } \hat{I} < I. \end{cases} \tag{1.26}$$

Finally, the overall score is usually determined by taking the geometric mean of all n-gram precisions from 1 to 4, multiplied with the brevity penalty for sentences shorter than the reference. Putting this all together, we derive

$$\textsc{Bleu}(e_1^I, \hat{e}_1^{\hat{I}}) = \text{BP}(I, \hat{I}) \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4} \log prec_n(e_1^I, \hat{e}_1^{\hat{I}})\right). \tag{1.27}$$

Bleu is computed at document rather than single-sentence level by counting all n-grams of translations over the entire dataset. This avoids an n-gram precision equaling zero. Since the score ranges from 0 to 1, it can be interpreted by percent. Thus, we exchangeably report it with points and percentages. However, the former sounds more precise. Although Bleu is the main metric to determine improvements in translation models, as it is quick and easy to use, it has problems such as having a relatively low correlation with human evaluation and ignoring sentence structure and the meanings of words [Reiter 18].

1.5.3 Translation Edit Rate

Similar to Wer, the translation edit rate (Ter) [Snover & Dorr+ 06] is an error metric based on the Levenshtein distance where the minimum number of edit operations is required to modify a hypothesis so that it exactly matches the reference. It is then normalized by the average length of the reference. Besides the possible edit operations listed for Wer, it also allows shifts of word sequences within the hypothesis. A contiguous sequence of words may be shifted within the hypothesis to another location. We assign an equal cost to all edits, including shifts of any number of words, by any distance. TER is defined as

$$\text{TER} = \frac{\text{number of edits}}{\text{average number of words in reference}}. \tag{1.28}$$

Again, the metric is from 0 to 100, but lower is better. We also report document-level TER, counting the edit rate over the entire dataset.

1.5.4 Alignment Error Rate

It is well known that manually performing a word alignment is a complicated and ambiguous task. One of the key quantitative metrics for the analysis of alignments in machine translation is the alignment error rate (AER) [Och & Ney 00], which gives the error of a given alignment in comparison to a golden reference. It employs an annotation scheme that explicitly allows for two different types of alignments. Unambiguous alignments are represented by an S ("sure") and a P ("possible") alignment that indicates ambiguous ones that might or might not exist. The words on the reference alignment might contain many-to-one and one-to-many relationships. The quality of a set of alignment points $A = \{(i, b_i) | b_i > 0\}$ is then computed by AER, which appropriately redefines precision and recall measures as

$$\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}. \tag{1.29}$$

The lower the value, the more satisfactory the result.

These are the most prominent metrics, and we will use them in this thesis as the main quantitative assessment tools. Unless explicitly stated otherwise, we use the official scripts of the machine translation evaluation campaigns to compute BLEU, i.e., `mteval-v13a`[2]. TER and WER are computed by `tercom`[3] and `sclite`[4], respectively. It is important also to note that there are many other different metrics, such as CHARACTER [Wang & Peter+ 16] or CHRF [Popovic 15], which are not used in this thesis.

1.6 About this Thesis

From the general perspective of Bayes' decision rule, posterior decomposition is not the only possible modeling approach. Recent methods, which are the main focus of this thesis, directly approximate the posterior probability in Equation 1.1. The main family of this group of approaches is based on *sequence-to-sequence modeling* [Sutskever & Vinyals+ 14, Cho & van Merrienboer+ 14a] and its extension to attention-based models [Bahdanau & Cho+ 15, Luong & Pham+ 15, Vaswani & Shazeer+ 17], often known as *end-to-end* methods. Attention sequence-to-sequence models have become an alternative to remarkably simplify the modeling process through single yet deep neural networks. They remove the use of complicated components and linguistic knowledge and make for a simpler search procedure. This thesis presents multiple concepts connecting neural sequence-to-sequence modeling for various applications, including machine translation, automatic speech recognition, and speech translation.

In Chapter 2, we present the scientific goals of this thesis. Chapter 3 gives an overview of the basic concepts of neural networks and explains the core neural architectures used as

[2]`ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13a.pl`. The results are identical to case-sensitive results computed by SacreBleu [Post 18].

[3]`http://www.cs.umd.edu/~snover/tercom/`

[4]`https://github.com/usnistgov/SCTK`

the fundamental design choices in this work. Section 3.5.3 is the starting point of this work's contribution. We explicitly highlight which parts are our contributions and which parts are based on previous publications. The main contributions of this thesis are elaborated in Chapters 4, 5, and 6. Chapter 4 describes our novel two-dimensional sequence-to-sentence modeling. Chapter 5 addresses a new integration of attention-based models into zero-order direct hidden Markov model formulation. Chapter 6 gives a comprehensive methodology of speech translation and includes novel contributions. Each chapter motivates our work with a short introduction. After a detailed description of models and methods, we discuss implementations and experimental results where we perform an extensive evaluation of the models and analysis on several large-scale tasks. The final sections of each chapter explicitly highlight the conclusions and put them into perspective with respect to related works. Finally, Chapters 7 and 8 summarize how we achieve our scientific goals and state individual contributions in contrast to team work.

1.6.1 Previously Published

The author has published the following scientific publications at peer-reviewed conferences during the course of this dissertation. More details on the author's individual contributions as opposed to teamwork can be found in Chapter 8.

- Machine translation
  - [Bahar & Alkhouli$^+$ 17] Empirical Investigation of Optimization Algorithms in Neural Machine Translation (EAMT 2017).
  - [Bahar & Brix$^+$ 18] Towards Two-Dimensional Sequence to Sequence Model in Neural Machine Translation (EMNLP 2018).
  - [Brix & Bahar$^+$ 20] Successfully Applying the Stabilized Lottery Ticket Hypothesis to the Transformer Architecture (ACL 2020).
  - [Bahar & Makarov$^+$ 20a] Investigation of Transformer-based Latent Attention Models for Neural Machine Translation (AMTA 2020).
  - [Bahar & Brix$^+$ 21] Two-Way Neural Machine Translation: A Proof of Concept for Bidirectional Translation Modeling using a Two-Dimensional Grid (SLT 2021).
- Speech recognition
  - [Bahar & Zeyer$^+$ 19a] On using 2D Sequence-to-Sequence Models for Speech Recognition (ICASSP 2019).
  - [Zeyer & Bahar$^+$ 19] A Comparison of Transformer and LSTM Encoder Decoder Models for ASR (ASRU 2019).
  - [Bahar & Makarov$^+$ 20b] Exploring a Zero-Order Direct HMM based on Latent Attention for Automatic Speech Recognition (ICASSP 2020).
- Speech translation
  - [Bahar & Bieschke$^+$ 19] A Comparative Study on End-to-End Speech to Text Translation (ASRU 2019).
  - [Bahar & Zeyer$^+$ 19b] On using SpecAugment for End-to-End Speech Translation (IWSLT 2019).
  - [Bahar & Bieschke$^+$ 21] Tight Integrated End-to-End Training for Cascaded Speech Translation (SLT 2021).

- International evaluation campaigns
  - [Peter & Toutounchi$^+$ 15] The RWTH Aachen German to English MT System for IWSLT 2015 (IWSLT 2015).
  - [Peter & Guta$^+$ 17] The RWTH Aachen University English-German and German-English Machine Translation System for WMT 2017 (WMT 2017).
  - [Bahar & Rosendahl$^+$ 17] The RWTH Aachen Machine Translation Systems for IWSLT 2017 (IWSLT 2017).
  - [Schamper & Rosendahl$^+$ 18] The RWTH Aachen University Supervised Machine Translation Systems for WMT 2018 (WMT 2018).
  - [Matusov & Wilken$^+$ 18] Neural Speech Translation at AppTek (IWSLT 2018).
  - [Rosendahl & Herold$^+$ 19] The RWTH Aachen University Machine Translation Systems for WMT 2019 (WMT 2019).
  - [Bahar & Wilken$^+$ 20] Start-Before-End and End-to-End: Neural Speech Translation by AppTek and RWTH Aachen University (IWSLT 2020).

2. Scientific Goals

The overall goal of this thesis is to improve the quality of machine translation with both forms of input, text and speech, and to reduce the error rate of speech recognition by developing novel stand-alone sequence-to-sequence models. Although attention-based sequence-to-sequence models are becoming a dominant approach in these applications, some essential concepts are still missing in their architectures. We therefore concentrate on rethinking the design of attention models and propose novel alternative models. However, many new challenges emerge when introducing new stand-alone models. The main consideration is to overcome some problems and limitations associated with attention models in one or more aspects. Given that, we outline the following scientific goals in this thesis:

Attention Refinements

One problem associated with attention models is that there is neither coverage nor previous attention information when computing attention scores at each position. The goal is to extend the recurrent attention model by providing implicit information from the last output positions without any help from external alignments. In this view, we study whether the extended models revise the attention component for better final translation.

Two-Dimensional Sequence-to-Sequence Modeling

In state-of-the-art attention models, both recurrent and self-attentive, input and output sequences are handled separately as one-dimensional sequences over time and then incorporated with an attention mechanism. The underlying attention output is generally a linearly weighted sum of input representations, and encoder states are computed only once at the beginning and are left untouched with respect to output histories. Since the encoder states are invariant to decoder states across output positions, context vectors can be quite similar, and thus are insufficient in discriminatively predicting output tokens over time. Hence, we go beyond current sequence-to-sequence backbone models and view translation as a two-dimensional mapping to define the correspondence between input and output sequences where an attention mechanism is no longer required. From a modeling perspective, the general goal is to study new two-dimensional sequence-to-sequence models as an alternative to attention models and to investigate potential improvements using a different point of view. In contrast to attention models which do not reinterpret encoder states while decoding, two-dimensional models enable the computation of input encodings as a function of previously generated output tokens. By providing encoder states with a greater capacity to remember what has been generated and what needs to be output, we aim to achieve higher translation quality.

Zero-order Hidden Markov Modeling

Attention components do not fulfill the same role as alignment models in conventional phrase-based machine translation. In particular, self-attentive models do not guarantee high-quality alignments. In contrast to IBM and hidden Markov models, where a transparent process defines the correspondence between inputs and outputs, it is often challenging for attention models to extract meaningful alignments. Alignments play a significant role in improving translation quality, and extracting comprehensible alignments is desirable when users tend to influence the translation output. An explicit stochastic latent variable can be a potential solution to define an alignment sequence in such models. Therefore, in a new modeling viewpoint, we propose the integration of attention sequence modeling into the hidden Markov model formulation. Accordingly, we develop an end-to-end model parameterized using either recurrent or self-attentive networks without any external alignment information. Since marginalization has an exponential number of terms with respect to the alignment dependency order of the model, we explore a zero-order assumption where latent variables have no dependency on their histories. The marginalization over the latent variable becomes simple and efficient for zero-order models and can be easily applied in both training and decoding. We also examine whether the proposed model is able to potentially generate higher quality alignments.

Development of End-to-End Models in the Context of Speech Translation

The traditional speech translation methods are based on a consecutive cascade pipeline of independently trained speech recognition and text translation systems, but they suffer from error propagation. Alternatively, this thesis aims at studying end-to-end models to directly translate speech observations into target texts. However, training such end-to-end models requires a moderate amount of paired translated speech-to-text data which is not easy to acquire. Consequently, the performance is often behind cascaded systems under realistic data conditions. Therefore, the next goal is to improve the performance of end-to-end models by revisiting existing techniques and developing new methods to leverage all types of training data, i.e., speech-to-source, source-to-target, and speech-to-target data. As the general goal, we intend to establish and advance best practices for end-to-end models with the ultimate goal of improving translation quality and providing recipes for real applications.

The rest of this work is structured according to the above goals. We present a series of extensive experiments performed on publicly available large-scale machine translation, automatic speech recognition, and speech translation tasks. By comparing with internal and external groups, we can assess the effectiveness and performance of the developed methods. They have all been made freely available to the research community and are accessible via the open-source toolkit RETURNN [Doetsch & Zeyer$^+$ 17, Zeyer & Alkhouli$^+$ 18].

3. Neural Sequence-to-Sequence Modeling

3.1 Introduction

In recent years, various fields in the spoken and written language technology have been boosted by the success of deep neural networks. The main idea of neural networks (NNs) concept dates back to the 1950s [Rosenblatt 58], and the first real application was carried out in the 1980s [Lippmann 87, Rumelhart & Hinton$^+$ 86]. However, for a long time, the use of neural networks in machine translation has been relatively shallow compared to automatic speech recognition. Neural networks have been used as a component in both conventional phrase-based translation and hybrid HMM recognition systems. The success of such methods is significantly reflected not only by the number of scientific papers but also by their wide deployment in production systems.

This chapter gives a brief overview of the fundamental concepts of neural networks in Section 3.2. We refer to [Bishop 07, Goodfellow & Bengio$^+$ 16] for a general in-depth discussion of neural networks. The chapter continues with neural language and sequence-to-sequence modeling in Sections 3.3 and 3.4, followed by the core concept of attention models in Section 3.5. It provides a comprehensive overview of the main neural architectures used as baselines and also the main design choices made when developing new models in this thesis. Most sections are not part of our contributions but are included to provide relevant context for the rest of this thesis. Our contribution begins in Section 3.5.3, where we extend the attention mechanism by keeping track of attention weights of the past output positions to provide more informative alignments. We use shorthand notations x and y as input and output vectors on neural networks discussions. Otherwise, throughout this thesis, we prefer a more general notation and use the standard machine translation one. The reason for this is to support unified and consistent formulation. However, all models discussed are also applied to speech recognition and speech translation. Experimental results including our baselines are presented in Section 3.8.1 for machine translation and in Section 3.8.2 for automatic speech recognition. The chapter ends with a conclusion and related works.

3.2 Neural Networks

Neural networks are a category of statistical methods in machine learning and have become a tool of choice, widely used to model and parameterize various distributions. According to the Universal Approximation Theorem [Cybenko 89], given enough data and nodes, neural networks are generic functions that can approximate any function to a certain degree and can be seen as the composition of functions.

3.2.1 Feedforward Neural Network

The basic building blocks of neural networks are layers consisting of a fixed number of nodes, also called neurons. Each feedforward layer contains a certain number of nodes, and it performs

Table 3.1: Commonly used activation functions in neural networks.

name	**function** $g(.)$
ReLu	$\max(0, x)$
sigmoid	$\frac{1}{1+\exp(-x)}$
hyperbolic tangent, $\tanh(x)$	$\frac{\exp(2x)-1}{\exp(2x)+1}$
softmax, for node k	$\frac{\exp(x_k)}{\sum_{k'} \exp(x_{k'})}$

very general operations. A fixed-size vector is used as an input to a layer. Assume an input vector $x \in \mathbb{R}^{d_x}$ and a new output vector $\hat{y} \in \mathbb{R}^{d_y}$, with d_x and d_y being the input and output dimensions. Each component in the input vector is multiplied by a trainable parameter, also known as a weight W. Each node then sums the result of each entry together and optionally adds another trainable parameter, called the bias b. A scalar value is then passed through the activation function g. The result of this function is the layer output at the current node. The results of all nodes are concatenated to form the output vector $\hat{y}$. These operations can be executed in parallel using matrix-vector multiplication. An example for such a definition is written by

$$\hat{y} = g(Wx + b) = g \circ \text{linear}(x). \tag{3.1}$$

Here, $W \in \mathbb{R}^{d_y \times d_x}$ and $b \in \mathbb{R}^{d_y}$ are trainable parameters, meaning that their values are changed to approximate a given function. We distinguish between the network output $\hat{y}$ and the true output y. linear represents a linear projection with trainable parameters. Hence, a layer in a neural network is the composition of multiple affine linear functions interleaved with a differentiable and nonlinear activation function, as stated in this formula. For sake of legibility, we use the symbol $\circ$ denoting the composition of functions[1]. Here, $\circ$ denotes the composition of linear and activation functions. The activation function can be chosen arbitrarily. Table 3.1 lists the common activation functions often used.

The softmax and sigmoid functions are used at the final or output layer of neural networks to obtain a categorical or Bernoulli distribution, respectively. The former is an especially interesting function because it normalizes the output to be a valid normalized distribution that sums to one. It allows a node to represent the probability of a certain discrete output. Therefore, in many classification tasks, the softmax function is applied to the output layer. The sigmoid function maps its input to a value between $(0, 1)$. In addition to the output and input layers, all intermediate layers are referred to as hidden layers. Besides sigmoid, ReLu and hyperbolic tangent, outputting values in the range $(0, \inf)$ and $(-1, 1)$, respectively, are mainly used for hidden layers.

In practice, to increase modeling capacity and approximate more complex functions, multiple layers are stacked on top of each other, forming a feedforward network. The network is often denoted as a function with a set of parameters $\theta = (W, b)$.

Regarding notation, each weight W, bias b, and activation function g is given a superscript l denoting the layer where the weight is used, with L being the last layer. In addition, each output of the intermediate hidden layer l is represented with $h^{(l)}$, and the last layer L is abbreviated with $\hat{y} = h^{(L)}$. Based on the fact that a combination of several linear functions can be seen as another linear function, the number of nonlinear activation functions is the component that mainly determines the depth of neural networks.

The generation of the entire network output $\hat{y} = g(x; \theta)$ is done by sequentially computing the output of each feedforward layer l, using the output of the previous layer $l-1$ as the input. For

[1]Given $f(x)$ and $g(x)$ as functions of $x \in \mathbb{R}$ such that $f : \mathbb{R} \to \mathbb{R}, g : \mathbb{R} \to \mathbb{R}$, then $f \circ g = f\big(g(x)\big)$.

the first layer, the input is x directly. This sequential pass is known as the forward pass. Neural networks can thus be viewed as the concatenation of individual layers, as expressed in

$$g(x;\theta) = g^{(L)}\Big(W^{(L)}g^{(L-1)}\big(W^{(L-1)}\dots g^{(1)}(W^{(1)}x + b^{(1)})\dots + b^{(L-1)}\big) + b^{(L)}\Big). \tag{3.2}$$

The parameters $W^{(l)}$ and $b^{(l)}$ for each layer l have no restrictions, other than their dimensions need to match up to have a valid matrix multiplication. In the case of feedforward networks, we have $\theta = \Big\{W^{(l)}, b^{(l)} | l \in \{1,\dots,L\}\Big\}$. Every layer typically has its own separate parameters, but the parameters can be shared across different layers, known as tying or sharing [Press & Wolf 17]. Such parameter sharing implies an inductive bias that aids better generalization. For simplicity, we drop the superscript l in the rest of the discussions.

3.2.2 Recurrent Neural Network

Feedforward networks can model highly complex distributions but are limited by their fixed-size inputs, making them unsuitable for variable-length inputs. Recurrent neural networks (RNNs) [Rumelhart & Hinton$^+$ 86] are specialized for processing sequential data of variable length and generating an output while carrying through a hidden state containing information about the previous time steps. Every such input sequence consists of vectors $x_t \in \mathbb{R}^{d_x}$ where $t \in \{1,\dots,T\}$ denotes the time step index or simply the position in the sequence for non-temporal data. Clearly, the sequence length T varies across different inputs.

In contrast to feedforward networks, RNNs can easily generalize to previously unseen sequence lengths. This is accomplished by parameter sharing where several parts of a model share the same parameters by a deep unfolded computational graph. Every member of the output is produced by applying the same function with the same parameters as the values from the previous time step. This allows for much better and more efficient generalization across different sequence lengths. As an example of an application that benefits hugely from using an RNN topology, consider the task of extracting certain information from a written sentence. Depending on the sentence, the relevant part could be at the beginning, middle, or end of the sentence. Feedforward networks being trained on sentences of a fixed length thus has to learn to detect time information at every single position separately. This requires lots of parameters and lots of training data. Recurrent neural networks, in contrast, share the same weights for several input features and allow for detecting the time information position independently, with fewer training examples, and also a generalization to sentence lengths that have not yet been seen in the training data [Goodfellow & Bengio$^+$ 16].

Almost all recurrent neural networks use recurrence in the definition of the values of their hidden units. An RNN that is trained to predict future values from the past, for example, can use the hidden state h_t as a fixed-size summary of the relevant aspects of the past inputs $x_1,\dots,x_t$. Formally speaking, the hidden state at time step t is computed as

$$h_t = g(W_x x_t + W_h h_{t-1} + b), \tag{3.3}$$
$$\hat{y}_t = g(W_y h_t) \tag{3.4}$$

where the same set of parameters θ for the activation function g is used at every single time step. $W_x \in \mathbb{R}^{d_h \times dx}, W_h \in \mathbb{R}^{d_h \times d_h}, W_y \in \mathbb{R}^{d_y \times d_h}$ are hidden-to-hidden, input-to-hidden and hidden-to-output matrices, and b is the bias term. $h_t \in \mathbb{R}^{d_h}$ is the hidden state at time step t with d_h nodes. At every time step, the RNN furthermore produces the output $\hat{y}_t$. Equation 3.3 is recurrent because the definition of h at time step t refers back to the same definition at time step $t-1$. For any finite t, the recurrence can be unfolded by repeatedly applying the definition, i.e.,

$$h_t = g\Bigg(W_x x_t + W_h \dots \bigg(g\big(W_x x_2 + W_h g(W_x x_1 + W_h h_0 + b) + b\big)\dots\bigg) + b\Bigg). \tag{3.5}$$

An RNN layer is commonly shortened when only presenting the inputs and output as

$$h_t = \mathrm{RNN}\left(x_t, h_{t-1}\right). \tag{3.6}$$

By comparing Equations 3.2 and 3.5, one notices that the RNN structure can be considered as a deep feedforward network with T layers over the time axis. Parameter sharing in RNN has two main benefits:

1. The learned model input size is independent of the input sequence length, while the length of the input of h_t depends linearly on t.

2. The same transition function with the same set of parameters θ can be used at every time step. Therefore, only relatively few parameters need to be learned, and the model is able to generalize to unseen sequence lengths.

So far, the benefits of parameter sharing in recurrent neural networks have been only considered. However, optimizing these parameters in a stable way that allows distant dependencies to have sufficient influence on the current state can be pretty difficult. This phenomenon can become problematic for longer sequences since training involves computing the gradient (see Section 3.6). This problem is known as vanishing and exploding gradients. Gradients close to zero give only a little insight into the directions that should be used for optimizing the parameters. Exploding gradients, on the other hand, can lead to very unstable learning behavior [Hochreiter 91, Bengio & Frasconi$^+$ 93, Bengio & Simard$^+$ 94].

In the next section, we look at a type of recurrent network with a gating mechanism that tries to overcome this problem. It is based on the idea of creating paths through time that neither vanish nor explode. We consider one subtype of gated RNNs which has proven to be very effective in practice, the so-called long short-term memory [Hochreiter & Schmidhuber 97]. There are several variations and alternatives to this and to gating mechanisms in general, such as the gated recurrent unit [Cho & van Merrienboer$^+$ 14a]. Since the performance of gated cells is similar, we only cover and employ the most common one.

3.2.3 Long Short-Term Memory

Long short-term memory (LSTM) has been proposed by [Hochreiter & Schmidhuber 97] and augmented with forget gates [Gers & Schmidhuber$^+$ 00], as well as extra peephole connections [Gers & Schraudolph$^+$ 02]. The main building block of LSTM is a cell that replaces the traditional hidden units. Each cell can accumulate and forget some states. The LSTM network consists of such cells that are connected with each other and have the network regular recurrences in addition to the self-recurrence within the cell.

The ability to learn long-term dependencies makes LSTM a valuable resource for sequence modeling. The building block of the variant of LSTM used in this work is shown in Figure 3.1. At every time step t, it gets the input x_t and the hidden state h_{t-1} from the previous time step. Each LSTM unit maintains some state information in an internal cell state c_t via a linear self-loop with a delay of one time step.

Three gates control information flow. The input gate g_{i_t} controls how much of the input should be added to the cell state. The forget gate g_{f_t} decides which information of the last cell state should be forgotten and can clear or reduce the information contained in the state by controlling the weight of the self-loop. The output gate g_{o_t} controls the output flow. All gates have access to the input x_t and the previous hidden state h_{t-1}. As written in Equation 3.11, the internal cell state is computed using the previous cell state c_{t-1} multiplied by the forget gate, and the candidate $\tilde{c}_t$ multiplied by the input gate. Finally, the output gate decides which contents of the obtained cell state should be part of the output. The gate units use the sigmoid nonlinearity σ

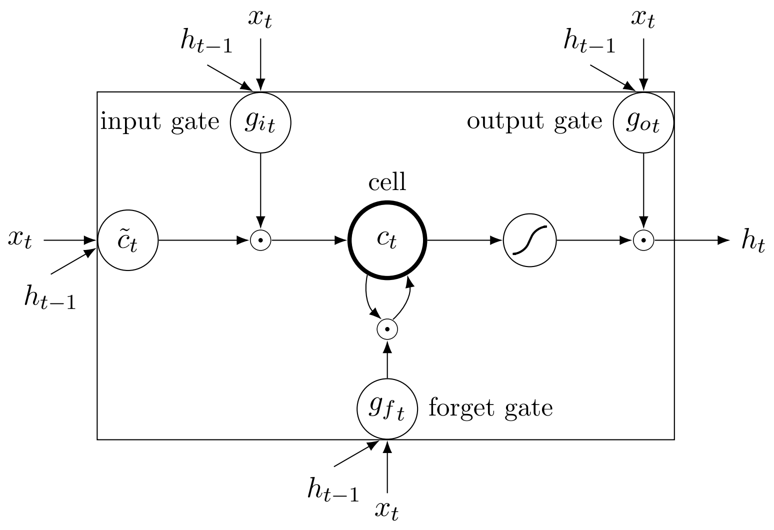


Figure 3.1: An example diagram of LSTM for the input x_t. All gates use the sigmoid activation function; the candidate cell arbitrarily uses tanh in this example.

to ensure a value between 0 and 1. The candidate unit cannot access c_t and has an arbitrary nonlinearity. Like in most implementations, we choose tanh. Considering that, the LSTM update equations can be formulated by

$$g_{i_t} = \sigma(W_i[x_t; h_{t-1}] + b_i), \tag{3.7}$$

$$g_{f_t} = \sigma(W_f[x_t; h_{t-1}] + b_f), \tag{3.8}$$

$$g_{o_t} = \sigma(W_o[x_t; h_{t-1}] + b_o), \tag{3.9}$$

$$\tilde{c}_t = \tanh(W_c[x_t; h_{t-1}] + b_c), \tag{3.10}$$

$$c_t = g_{f_t} \odot c_{t-1} + \tilde{c}_t \odot g_{i_t}, \tag{3.11}$$

$$h_t = \tanh(c_t) \odot g_{o_t} \tag{3.12}$$

where $W_{\{i,f,o,c\}} \in \mathbb{R}^{d_h \times [d_x + d_h]}$ are weight matrices and $b_{\{i,f,o,c\}} \in \mathbb{R}^{d_h}$ are bias vectors. $\odot$ indicates the element-wise multiplication. Additional peephole connections can be added. The LSTM layer is often shortened as a function if only presenting the inputs and output and defined by

$$h_t = \mathrm{LSTM}\left(x_t, h_{t-1}\right). \tag{3.13}$$

3.2.4 Bidirectional Long Short-Term Memory

We have seen how the output at time step t can depend on the input at time step t and previous time steps $1, \ldots, t-1$. In some applications, the output at time step t might also depend on the input values at future time steps $t+1, \ldots, T$. For instance, in machine translation, it seems necessary to consider the whole source sentence when generating the target sentence. The current target token might depend on previous and following source words, especially for language pairs requiring word reordering. If there are two similar interpretations of one part of the source sentence, one can choose the one that creates the more likely combination of words. Thus, a suitable context representation has to look far into the past of the sequence and arbitrarily far into the future.

To allow RNNs to tackle this situation, bidirectional RNNs were invented by simply introducing an additional hidden layer with the same connections as in the original hidden layer but moving backward in time [Schuster & Paliwal 97]. Later an extension to LSTM units has been also introduced [Graves & Jaitly+ 13b, Sundermeyer & Alkhouli+ 14], known as *bidirectional LSTM (BiLSTM)*. $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are used to denote the forward and backward hidden layers. As a combination of two LSTMs, BiLSTM creates outputs that focuses on the inputs right around time step t but may also depend on input values that are far in the future or past. The hidden state h_t is the concatenation of the hidden states from the forward and backward layers written as

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]. \tag{3.14}$$

Similar to LSTM as an abstract function, a shorthand notation for BiLSTM is represented as

$$h_t = \text{BiLSTM}\left(x_t, h_{t-1}, h_{t+1}\right). \tag{3.15}$$

The concept of combining several recurrent networks can be extended, e.g., for handling two-dimensional inputs such as images. We discuss an extension of LSTM to two dimensions later in Section 4.2, with the bidirectional concept in Section 4.3.4. Unless explicitly stated otherwise, in the course of this thesis, we leave out weight matrices and bias vectors for the sake of simplicity and consider layers as functions whose equations show the dependency among variables.

3.3 Neural Language Model

As an alternative to the count-based n-gram models discussed in Section 1.3.2, the probability of a word sequence can be modeled using neural networks. Neural network-based language models aim to solve the problems of count-based models, particularly data sparsity and lack of generalization. Lack of generalization in this context means similar word sequences are not assigned similar probabilities because word representation is discrete. Neural language models alleviate these problems by representing words by continuous vectors [Bellegarda 97, Bengio & Ducharme+ 03] where learned continuous word representations capture morphological, syntactic, and semantic similarity across words [Collobert & Weston 08], thus exhibiting better generalization compared to count-based models.

To map each discrete token in a finite vocabulary to a continuous representation, one-hot vectors of the vocabulary size are utilized. In the one-hot representation, all components are zero other than the k-th element which corresponds to the k-th token of the vocabulary. Working with machine translation notations, we convert a discrete word e_i to a real-valued vector $\tilde{e_i}$ by multiplying its one-hot vectors with an embedding matrix. This multiplication simplifies to a look-up table. The word embedding $\tilde{e_i}$ has a much smaller dimension than the vocabulary. The embedding matrices are shared between positions and usually trained jointly with the rest of the network. They are passed to the upper hidden layers. Eventually, the softmax layer predicts the probability of a word given the history, as stated in Equation 1.15. The hidden layers can be modeled by both feedforward and recurrent neural networks. The former limits the history to n-gram tokens, while the latter provides the entire context and a powerful solution to the language modeling task. For more insight into neural language models, we refer the reader to [Bengio & Ducharme+ 03, Schwenk & Gauvain 04, Mikolov & Karafiát+ 10, Sundermeyer & Oparin+ 13, Liu & Saleh+ 18, Al-Rfou & Choe+ 19].

Assuming an LSTM layer to model the recurrence, we have

$$o_i = \text{linear} \circ \text{LSTM}(s_{i-1}, \tilde{e}_{i-1}), \tag{3.16}$$

$$p(e_i = w|e_0^{i-1}) = \frac{\exp(o_{iw})}{\sum_{v=1}^{|V_e|} \exp(o_{iv})} = \text{softmax}(o_i)|_w \tag{3.17}$$

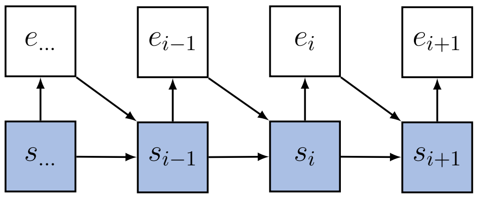


Figure 3.2: A simple recurrent-based language model with its dependencies.

where $\tilde{e}_{i-1}$ and s_{i-1} are the embedding vector of the previous token and the previous LSTM hidden state. e_0 represents the word at the first position. The linear function maps the output dimension to the vocabulary size. After that, softmax predicts the current token. To reduce the total number of parameters, the embedding and the output projection matrices can be tied [Press & Wolf 17].

Figure 3.2 illustrates the recurrent neural network language model and its dependencies. In the next section, we see that such the recurrent auto-regressive language model conditioned on the input representations can be applied as a decoder in sequence-to-sequence modeling.

3.4 Sequence-to-Sequence Model

Sequence-to-sequence models are mainly based on a family of prevailing architectures known as *encoder-decoder networks*, single and stand-alone, yet strong neural models that generate contextually appropriate, arbitrary-length output sequences. Encoder-decoder architectures have been used in a wide range of applications [Kalchbrenner & Blunsom 13, Sutskever & Vinyals$^+$ 14, Cho & van Merrienboer$^+$ 14a]. The key idea underlying these methods is based on two components: an *encoder* and a *decoder*. The encoder reads a variable-length source sequence and encodes it into a single continuous vector representation. Then, the source representation is passed to the second component, the decoder, which generates a target sequence step by step. The decoder is considered to be a neural language model conditioned on source representations. In such architectures, the input and output sequences are separately handled.

As explained earlier in Chapter 1, our main goal is to model a conditional distribution of the target sentence given the source sentence. Using the chain rule, the posterior probability of the target sentence is factorized into

$$p(e_1^I|f_1^J) = \prod_{i=1}^{I} p(e_i|e_0^{i-1}, f_1^J). \tag{3.18}$$

$p(e_i|e_0^{i-1}, f_1^J)$ can be modeled in different ways. At each time step, the model computes the word-level probability. Convolutional neural networks (CNNs) [Kalchbrenner & Blunsom 13, Gehring & Auli$^+$ 17a, Gehring & Auli$^+$ 17b], deep self-attentive layers [Vaswani & Shazeer$^+$ 17], and deep stacked recurrent networks [Sutskever & Vinyals$^+$ 14, Cho & van Merrienboer$^+$ 14a] can be employed to model the encoder, decoder, or both.

This work concentrates solely on the encoder-decoder topology equipped with either the RNN-based LSTM described below or self-attentive layers described in Section 3.5.4. The models differ in the way these architectures are used to compute hidden representations. In the first layer of the network, a source sentence of J words is projected into a series of embedding vectors $\tilde{f}_1^J$ by applying the look-up table to each of its tokens. In the next layer, an encoder reads the series of vectors and encodes them in a sequence of hidden representations. Here, the LSTM layer is taken, and the encoder is formulated as

$$h_j = \text{LSTM}\left(h_{j-1}, \tilde{f}_j\right). \tag{3.19}$$

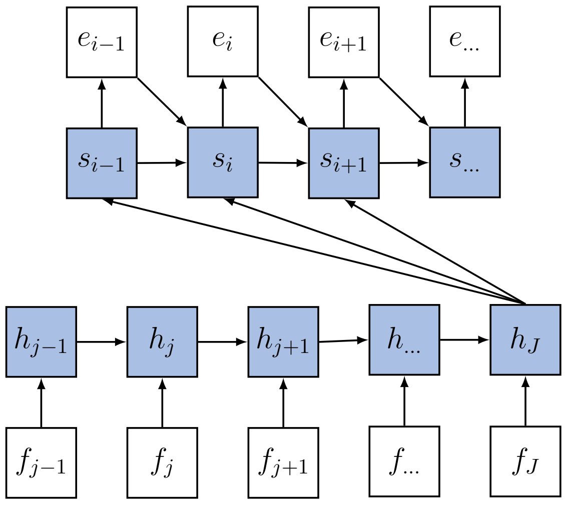


Figure 3.3: The architecture of the encoder-decoder network with a fixed-length source sentence representation. The blue blocks depict the encoder and decoder recurrent layers.

h_j represents the LSTM encoder state. The final hidden state is the summary of the whole source sentence. We call it the *context vector* $c(f_1^J) = h_J$ that contains source information.

The decoder is a recurrent-based language model on the target side conditioned on the source sentence as written in Equation 3.20 [Kalchbrenner & Blunsom 13, Cho & van Merrienboer$^+$ 14a]. Once the source sentence has been encoded, the decoder starts generating the first target token e_0, which is then fed back to the decoder network to produce the second token, and so on. The algorithm terminates when the network generates the end-of-sentence token </S>. Therefore, given the last decoder state, the embedding of the previous target token, and the context vector, we update the decoder state s_i and predict the current token by

$$s_i = \text{LSTM}\left(s_{i-1}, \tilde{e}_i, c(f_1^J)\right), \tag{3.20}$$

$$p(e_i|e_0^{i-1}, f_1^J) = \text{softmax} \circ \text{linear}\left(s_{i-1}, \tilde{e}_{i-1}, c(f_1^J)\right). \tag{3.21}$$

Alternatively, $c(f_1^J)$ can be used only to initialize the first decoder state s_0 corresponding to the target beginning-of-sentence token [Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15].

Figure 3.3 shows an abstract overview of the encoder-decoder network. In contrast to traditional phrase-based machine translation with a very large number of highly engineered features, the neural encoder-decoder models are simple and efficient.

In sequence-to-sequence modeling, particularly offline modeling, the assumption is that the entire source sentence is available. Unlike the standard way in which the tokens to the encoder are fed in the natural order they appear in the sentence [Cho & van Merrienboer$^+$ 14b], one can also read the sentence in reverse order [Sutskever & Vinyals$^+$ 14]. A bidirectional LSTM in the encoder reads the source sentence in the backward direction. Attention sequence-to-sequence models, which are powerful extensions to the standard encoder-decoder topology, read the source sentence in both forward and backward directions, as discussed next.

3.5 Attention Sequence-to-Sequence Model

One of the problems associated with a fixed-size context vector of the source sequence is ignorance of its length. Clearly, sequences with different lengths convey different amounts of information, and a fixed-size vector limits the capacity of the model, and it is show that the translation quality is considerably worse on long sequences [Cho & van Merrienboer$^+$ 14a]. To

Table 3.2: Attention scoring functions. W and ν denote trainable parameters of the network.

name	**scoring function** score(K, Q)
additive	$\nu^\top \tanh(W[Q; K])$
multiplicative	$K^\top W Q$
scaled multiplicative	$\frac{K^\top W Q}{\sqrt{d}}$

mitigate this issue for machine translation, the concept of *attention* is proposed by [Bahdanau & Cho+ 15] as an extension to the encoder-decoder model to avoid having a fixed-length source sentence representation. Instead of a single vector, the model encodes the source sequence into a series of vectors and chooses a subset of them adaptively by a weighting mechanism in the decoder. The attention layer does a soft search for a set of positions on the source sentence where the most relevant information is concentrated. We first address attention as a general concept before describing the architecture in detail in Section 3.5.2.

3.5.1 Attention

Generally speaking, attention is a mapping operation that takes J key-value pairs and I query vectors [Vaswani & Shazeer+ 17]. It can be seen as a search problem over certain keys given a query, returning a value that the keys represent. For simplicity, we assume all vectors have the same dimension d, and squeezing the vectors results in the key $K \in \mathbb{R}^{J\times d}$, value $V \in \mathbb{R}^{J\times d}$, and query $Q \in \mathbb{R}^{I\times d}$ matrices. Then, the attention concept is formally formulated as a function of matrix operations expressed as

$$\text{attention}(K, V, Q) = \text{softmax}\big(\text{score}(K, Q)\big)|_j V. \tag{3.22}$$

The output is the weighted sum of the values based on the mapping of the query to the keys, provided that the query, keys, and values are all vectors. In the end, for each query vector, the weighted sum of the value vectors is obtained. A similarity scoring function between the query vector and all the keys calculates weights. It results in a $(J \times I)$ matrix of unnormalized scores. Softmax normalizes the weights over the columns of that matrix so that the weights for each query vector sum to one. There are a couple of variants of scoring functions, such as additive [Bahdanau & Cho+ 15], multiplicative or dot-product, which can be grouped into either general multiplicative [Luong & Pham+ 15] or scaled multiplicative [Vaswani & Shazeer+ 17]. In the latter case, unnormalized scores are divided by the square root of the key vector dimension for more stable gradients. Table 3.2 summarizes the most common attention scoring functions.

Given the encoder-decoder architecture, the decoder hidden states s_0^I are used as the query vectors, and the encoder hidden states h_1^J play the role of the key and value vectors. In an uncommon topology, the encoder states is divided into two parts. The first and second halves are used as the key and value vectors, respectively [Mino & Utiyama+ 17]. Once we have processed the whole source and target sentences, the attention matrix $\mathrm{A} \in \mathbb{R}^{J\times I}$ is calculated given by

$$\mathrm{A} := \text{softmax} \circ \text{score}(s_i, h_j) \quad \forall j \in [1, J] \text{ and } \forall i \in [1, I]. \tag{3.23}$$

The entry $\mathrm{A}_{j,i}$ represents the attention score between positions i and j. As it is obtained from a distribution, we use probabilistic notations for it, as $\mathrm{A}_{j,i} = \alpha(j|i)$. However, it is often challenging for fuzzy attention weights to extract comprehensible alignments compared to count-based IBM and HMM alignment models, where a hard alignment is a transparent process. Figure 3.4 depicts

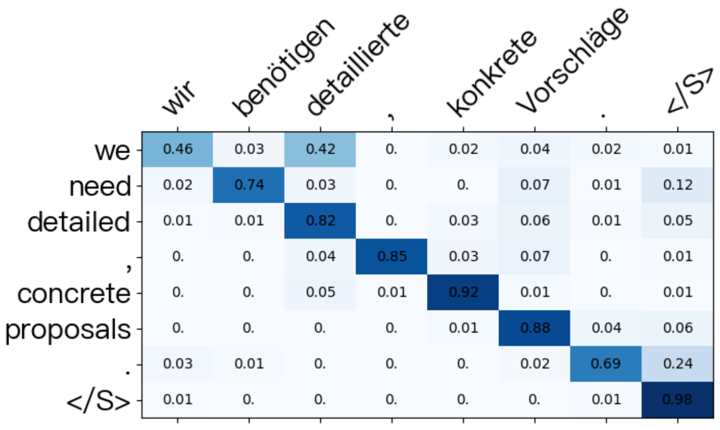


Figure 3.4: Attention weight matrix A for German→English translation. Each cell presents the normalized attention weights $\alpha(.|.)$ between the corresponding source (column) and target (row) tokens. The darker the blue color, the higher the attention weight is. As shown, the weights over the rows for each English target word sum up to one. </S> represents the end-of-sentence token.

an example of the attention matrix A in machine translation using additive attention. In this example, the source sentence "wir benötigen detaillierte , konkrete Vorschläge ." is translated to "we need detailed , concrete proposals ." The attention matrix captures cross-lingual word relations such as "benötigen" being relevant for generating "need" or the aligned word for "Vorschläge" in English being "proposals" thus emitting a high attention weight for this pair.

Soft, Hard, and Local Attention

In the literature, the aforementioned mechanism is often called *soft attention*. Soft attention provides a way to observe what influences the model decision-making process. Even when the model has very sharp values in soft attention, it still has some degree of spread around other tokens, as shown in Figure 3.4. This property is in contrast to *hard attention* introduced for image caption generation [Xu & Ba$^+$ 15]. In their work, soft attention weights are placed softly over all patches in the source image. On the other hand, hard attention attends to only one patch of the image at a time. In hard attention, each output token focuses only on one input position, resembling the concept of conventional alignments discussed Section 1.2.1. Although the hard attention model is much more efficient at inference time and well-trained models succeed in certain settings, it is not differentiable, is harder to train, and the gradient of the attention weights is subject to high variance. Therefore, it requires techniques like an approximate variational lower bound or REINFORCE [Williams 92].

As said, attention weights are normalized over all source positions, meaning that it attends to all tokens. This inspires the term global attention. The problem of such global attention approaches is that firstly they are impractical for streaming purposes where there is no access to the entire input sequence, and the output sequence is immediately generated as long as a portion of the input is received. Simple examples are ASR streaming or simultaneous translation. Secondly, global attention can be expensive for long sequences, e.g., paragraphs or documents. To solve these deficiencies, local attention focuses only on a small subset of the source positions per target token [Luong & Pham$^+$ 15]. Local attention lies between the soft and hard attention models. Unlike hard attention, it is differentiable almost everywhere, making it easier to implement. Since its performance in both speech recognition [Zeyer & Merboldt$^+$ 18, Merboldt & Zeyer$^+$ 19] and machine translation [Luong & Pham$^+$ 15] is behind the global attention methods, this work leaves it out of its scope. Instead, we propose an alternative model to soft attention that is akin to hard attention and resembles conventional alignments. We cover it in detail in Chapter 5.

Multihead Attention

A further improvement upon the basic attention mechanism is its extension to multiple memory networks [Cheng & Dong$^+$ 16]. A similar idea is used to retrieve different context vectors, instead of just one, as first described in [Vaswani & Shazeer$^+$ 17]. The intuition behind multihead attention is that the model receives information from different representation subspaces at different positions. Hence, it might capture different sources of knowledge. The query, key, and value vectors are linearly projected H times with different trainable linear projections. H is the number of heads performing the attention scoring function in parallel. This results in an H normalized distribution to represent different attention weights. The concatenation of the output of each attention head obtains the final output of multihead attention. In practice, to avoid increasing the number of parameters, the dimension of the query, key, and value vectors is divided by the number of heads. Each part calculates a context vector separately, with all resulting context vectors being concatenated together to form the final one. Once again, after concatenation, the final dimensionality remains the same. Mathematically, it means

$$\begin{aligned}\text{MultiHeadAttention}(K, V, Q) &= [\text{head}_1; \ldots; \text{head}_H]W_O \\ \text{with}\,\text{head}_m &= \text{attention}(KW_K^m, VW_V^m, QW_Q^m) \end{aligned} \tag{3.24}$$

where W_K^m, W_V^m, $W_Q^m \in \mathbb{R}^{d\times\frac{d}{H}}$ are the trainable parameters for the m-th head of the key, value, and query matrices, and $W_O \in \mathbb{R}^{H\frac{d}{H}\times d}$ keeps the final dimensionality. Having multiple attention heads is challenging for deriving a single attention weight matrix A.

Self-Attention

Up until now, we have examined attention only in the situation of the decoder attending to the encoder in the context of the encoder-decoder sequence-to-sequence architecture. Based on generalized attention, attention can be used in the encoder, using linearly transformed encoder hidden states for the query, keys, and values [Vaswani & Shazeer$^+$ 17]. Such an attention application is called *self-attention.* Self-attention calculates attention weights between two elements of the same sequence. Given Equation 3.22, the query, keys, and values are identical vectors. The same concept can be applied in the decoder, as we see later in Section 3.5.4.

The attention mechanism is an augmentation to the sequence-to-sequence architecture for better integration between the encoder and decoder. In this thesis, we refer to it as the *attention sequence-to-sequence* or simply *attention model.*

3.5.2 Recurrent Attention Model

Recurrent-based attention models are based on three main components. An additional attention layer incorporates the encoder and decoder, which are based on LSTM cells. In contrast to the standard encoder-decoder models in which a stack of unidirectional LSTM layers is used to encode the source sentence, in recurrent-based attention models, the encoder is equipped with L_e bidirectional LSTMs stacked on top of each other [Bahdanau & Cho$^+$ 15, Luong & Pham$^+$ 15]. Using Equation 3.15, we write

$$h_j = \text{BiLSTM}^{(L_e)} \circ \ldots \circ \text{BiLSTM}^{(1)}(\tilde{f}_j, h_{j-1}, h_{j+1}). \tag{3.25}$$

Once the whole source sequence has been read and encoded to the sequence of hidden states, the decoder starts to generate the output sequence token by token. At each time step, the attention mechanism allows the model to focus on certain source positions. All the attention scoring functions listed in Table 3.2 can be employed to compute the attention weights. Here, we use single-head additive attention that is a feedforward neural network, even though it can be expanded to multihead. Again, we write

$$\alpha'_{j,i} = \text{score}(s_{i-1}, h_j) = \text{linear} \circ \tanh\big(s_{i-1}, h_j\big), \tag{3.26}$$

$$\alpha(j|i) = \text{softmax}(\alpha'_{j,i})|_j \tag{3.27}$$

where $\alpha'_{j,i} \in \mathbb{R}$ is an unnormalized score which is normalized by softmax to produce the attention weights $\alpha(j|i)$. It calculates the importance of each encoder state h_j in relation to the last decoder state s_{i-1}. The attention weights are then used as a weight for mixing the encoder states together and computing the context vector as

$$c_i = \text{attention}(K = h_j, V = h_j, Q = s_{i-1}) = \sum_{j=1}^{J} \alpha(j|i) h_j. \tag{3.28}$$

The context vector is usually appended to the hidden state of the LSTM decoder to calculate the posterior distribution. It depends on the target position i, in contrast to the non-attention encoder-decoder model, where a fixed vector c is used over all target positions. Therefore, the dependencies of Equations 3.20 and 3.21 are updated to

$$s_i = \text{LSTM}\big(s_{i-1}, \tilde{e}_i, c_i(f_1^J)\big), \tag{3.29}$$

$$p(e_i|e_0^{i-1}, f_1^J) = \text{softmax} \circ \text{linear} \circ \text{maxout}\big(s_{i-1}, \tilde{e}_{i-1}, c_i(f_1^J)\big). \tag{3.30}$$

Here, the decoder contains one layer of LSTM as shown in Figure 3.5. But it is usually deeper in some settings. There is also an extra layer called maxout [Goodfellow & Warde-Farley$^+$ 13] which reduces the hidden dimension before the output layer to lower the size d_m. This leads to an easier matrix-vector product, hence speeding up training a bit. The maxout layer operates

$$\text{maxout}(\tilde{o}) = \big[\max\{\tilde{o}_{i,2k-1}, \tilde{o}_{i,2k}\}\big]_{k=1,\dots,d_m}^{\top} \tag{3.31}$$

where $\tilde{o}_{i,k}$ is the k-th element of the vector $\tilde{o}_i$ to be an input to the maxout function.

From the modeling perspective, c_i adds indirect information about the source sentence into the decoder, thus making a dependency on f_1^J compared to non-attention-based models. It is important to highlight that, in some configurations, the decoder state gets updated first, and then the target token is generated, as in [Bahdanau & Cho$^+$ 15]. This means that in the attention computation, s_{i-1} does not contain implicit information about $\tilde{e}_{i-1}$, but $\tilde{e}_{i-2}$ instead. The hidden states of the decoder are shifted by one. Adding an explicit term $\tilde{e}_{i-1}$ in attention calculation solves the issue. In our implementation, we first predict the target token $\tilde{e}_i$ and then update the decoder state s_i, correspondingly. Therefore, s_{i-1} implicitly contains information about $\tilde{e}_{i-1}$.

For future reference, the sequence-to-sequence model with attention and LSTM cells is often simply termed the *recurrent-based attention* or *RNN attention model*.

3.5.3 Attention Refinement

This section serves as the starting point of our contributions where we verify several extensions of the attention mechanism from the literature and further refine them. The RNN attention model is highly effective in relating the source and target sentences. However, it does not make any conditional dependence assumption on previous attention information. This is a zero-order model that ignores past alignments, and in fact, it exhibits a lack of coverage. In phrase-based machine translation, a coverage vector indicates whether a source word has been translated or not. The

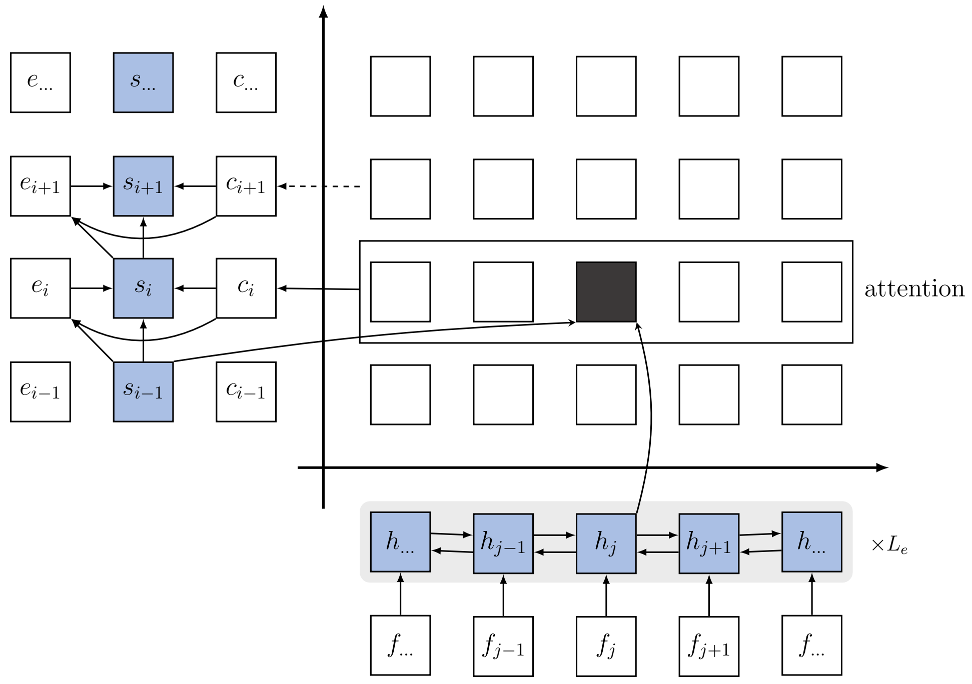


Figure 3.5: The architecture of the encoder-decoder network equipped with an attention layer. The blue cells show the encoder and decoder recurrent states. The bidirectional dependency between h_js in the encoder (left-to-right and right-to-left) is in contrast to the unidirectional property of the decoder (bottom-to-up). In this work, a deeper encoder with L_e layers and one decoder layer are taken.

search procedure is terminated if all source tokens are covered. In attention-based models, there is no coverage that might lead to over-translation, where some words are unnecessarily translated multiple times or under-translation where some words are not translated at all.

The problem of the standard attention model which ignores past alignments is addressed by [Tu & Lu$^+$ 16]. They refer to this issue as the lack of coverage. In their work, they maintain a coverage vector to keep track of previous attention weights. This vector is fed back to adjust future attention weights. In other words, it gives the model a flag to consider more about untranslated source tokens. If a source token has been already translated, it is less likely to be translated again and should be assigned a lower score. They also propose the concept of fertility inherited from word-level machine translation [Brown & Pietra$^+$ 93]. Similarly, RNN attention models include structural biases from word-based alignment models, positional bias, Markov conditioning, fertility, and agreement over translation directions [Cohn & Hoang$^+$ 16]. Next, we overview and extend some of these ideas, and in the experimental section, we compare them.

Last Attention Weight We feedback the immediate predecessor attention weight in order to compute the unnormalized score at the current position i. The Equation 3.26 is rewritten as

$$\alpha'_{j,i} = \text{linear} \circ \tanh\big(s_{i-1}, h_j, \alpha(j|i-1)\big). \tag{3.32}$$

Accumulated Attention Weight We can also pass on all attention weights from the first target position, i.e., from position 1 to $i-1$ by accumulating them through the sum. Therefore, we have

$$\alpha'_{j,i} = \text{linear} \circ \tanh\big(s_{i-1}, h_j, \sum_{k=1}^{i-1} \alpha(j|k)\big) \tag{3.33}$$

where the third term is accumulated attention.

Fertility Similar to [Tu & Lu+ 16, Cohn & Hoang+ 16], the fertility concept (especially for machine translation) can be added as a better way of providing coverage. To this end, the fertility value ϕ_j of f_j is multiplied by the accumulated attention weights given by

$$\alpha'_{j,i} = \text{linear} \circ \tanh\left(s_{i-1}, h_j, \phi_j \sum_{k=1}^{i-1} \alpha(j|k)\right) \tag{3.34}$$

where ϕ_j depends on the encoder states because it can vary if the word is used in a different context and is defined as

$$\phi_j = \frac{\sigma \circ \text{linear}(h_j)}{2} \tag{3.35}$$

where the denominator specifies the maximum value for fertility, which is set to 2 in our experiments.

Last Context in Attention A possible extension to consider past attention weights is to include the last context vector in the computation of the current attention score, i.e.,

$$\alpha'_{j,i} = \text{linear} \circ \tanh(s_{i-1}, h_j, c_{i-1}). \tag{3.36}$$

Last Context in Decoder An alternative is to include the past context vector in the computation of decoder state s_i and target word e_i. Although the layers get information from the current context vector, it may be possible to benefit from past ones recursively. Therefore, we rewrite Equations 3.29 and 3.30 as

$$s_i = \text{LSTM}\big(s_{i-1}, \tilde{e}_i, c_i, c_{i-1}\big), \tag{3.37}$$

$$p(e_i|e_0^{i-1}, f_1^J) = \text{softmax} \circ \text{linear} \circ \text{maxout}\left(s_{i-1}, \tilde{e}_{i-1}, c_i, c_{i-1}\right). \tag{3.38}$$

Embedding in Attention As discussed before, the attention scoring functions take the encoder and decoder states to compute a distribution over input positions. It is often arguable that the encoder hidden states include the entire source context and the decoder hidden states include all histories until the current position. In order to explicitly have information for the corresponding source and target tokens, we add word embeddings to the attention computation that is we add $\tilde{f}_j$ and $\tilde{e}_{i-1}$ for the source position j and the target position i, respectively, which gives

$$\alpha'_{j,i} = \text{linear} \circ \tanh(s_{i-1}, h_j, \tilde{f}_j, \tilde{e}_{i-1}). \tag{3.39}$$

3.5.4 Self-Attentive Model

Using the multihead attention and self-attention concepts, another model, named *transformer* or generally the *self-attentive* model has been proposed [Vaswani & Shazeer+ 17]. The self-attentive model closely resembles the RNN attention model, both having encoder and decoder components. But the main intuition behind this network is to remove all recurrent components and substitute them with feedforward layers. It differs from its RNN counterparts in that, in RNN, the input is processed sequentially, and the model produces the output depending on the relation between the current input and all previous ones. This sequential property makes processing slow due to its dependency on predecessors.

This fact motivates self-attentive models to completely replace the LSTM cells in both the encoder and decoder with self-attention and feedforward layers. Exploiting the power of graphical processing units (GPUs), self-attentive models can be trained much faster compared to RNN attention. In comparison, self-attentive models are usually trained with deeper layers. We note that the architecture presented here is not a strict copy of the one presented in the main paper, but rather a highly similar version, as implementation details can strongly affect the model performance. In the following, we explain the main components of the architecture.

Positional Encoding Since there is no recurrent components, the model has to preserve the order of the sequence. Hence, positional encoding is introduced. There are two ways to include positional encoding. It can be learned with the network parameters [Gehring & Auli$^+$ 17b] or it can be computed based on sinusoidal functions. As they both lead to almost identical results, we use the fixed version that adds a sinusoid wave based on the positions in the sentence, and to some extent, it allows the model to infer sequences longer than those that have been seen during training [Vaswani & Shazeer$^+$ 17]. The sinusoidal positional encoding is calculated by

$$\mathrm{PE}(j,k) = \begin{cases} \sin\left(\frac{j}{10000^{(2k/d)}}\right) & \text{if } k \text{ is even} \\ \cos\left(\frac{j}{10000^{(2k+1/d)}}\right) & \text{otherwise.} \end{cases} \tag{3.40}$$

The function returns the positional encoding (PE) at position j for the k-th feature of the vector of size d, which is the model size.

Encoder Once the positional encoding has been computed, it is added or concatenated to the word embedding. The positional encoding is used both in the encoder and decoder as the first step. In our notations, we refer to embedding as if they are the output of the 0-th layer. Applying dropout [Srivastava & Hinton$^+$ 14] as a regularization technique, we have

$$f_j \rightarrow h_j^{(0)} = \text{dropout} \circ \big(\mathrm{PE}(j,k) + \frac{\tilde{f}_j}{\sqrt{d}} \big). \tag{3.41}$$

Each encoder layer is a stack of two sublayers: a multihead self-attention and a feedforward layer with the ReLu activation. Each sublayer applies preprocessing and postprocessing operations. Layer normalization [Ba & Kiros$^+$ 16] is used as preprocessing, whereas dropout and residual connection [He & Zhang$^+$ 16] are chosen for the postprocessing step. One encoder layer can then be expressed as

$$\text{TransEncLayer} = \text{FFBlock} \circ \text{MHeadSelfAttBlock} \tag{3.42}$$

where MHeadSelfAttBlock and FFBlock refer to the multihead self-attention and feedforward sublayers with a predefined sequence of operations like

$$\text{MHeadSelfAttBlock}(x) = x + (\text{dropout} \circ \text{linear} \circ \text{self-attention} \circ \text{layer-norm})(x), \tag{3.43}$$

$$\text{FFBlock}(x) = x + (\text{dropout} \circ \text{linear} \circ \text{ReLu} \circ \text{layer-norm})(x) \tag{3.44}$$

with x being the input.

In the first layer, the input of TransEncLayer is the embedding vectors $h_j^{(0)}$, while in the upper layers, the previous encoder states are used as its input. Self-attention weights at layer $(l+1)$ for each j position are computed using the previous layer, defined as

$$\alpha_m^{(l+1)}(j'|j) = \text{softmax} \circ \text{DotAtt}(h_{j,m}^{(l)}, h_{j',m}^{(l)}) \quad \forall m \in [1, \ldots, H] \tag{3.45}$$

where H is the number of heads, $j' \in [1, J]$, and the scoring function is chosen to be the scaled multiplicative function noted by DotAtt, as defined in Table 3.2. Having these weights, the next layer representation is defined by

$$h'^{(l+1)}_{j,m} = \sum_{j'=1}^{J} \alpha_m^{(l+1)}(j'|j) h^{(l)}_{j',m}. \tag{3.46}$$

According to Equation 3.24, the final output of the self-attention layer with multihead is computed by concatenating the representations of all heads given by

$$h'^{(l+1)}_{j} = [h'^{(l+1)}_{j,1}; \ldots ; h'^{(l+1)}_{j,H}]. \tag{3.47}$$

We distinguish between $h'^{(l)}_j$ and $h^{(l)}_j$ such that the former is the output of the self-attention layer while the latter represents the output of the encoder at layer l. Similar to the RNN attention model, the encoder can be the stack of TransEncLayer on top of each other, given by

$$h_1^{J(L_e)} = (\text{layer-norm} \circ \text{TransEncLayer}^{(L_e)} \circ \ldots \circ \text{TransEncLayer}^{(1)})(f_1^J) \tag{3.48}$$

where L_e is the number of encoder layers. Figure 3.6 illustrates the abstract overview of the self-attentive model.

Decoder Similar to the encoder, the first layer is an embedding layer, given as

$$e_i \to c_i^{(0)} = \text{dropout} \circ \big(\text{PE}(i,k) + \frac{\tilde{e}_i}{\sqrt{d}}\big). \tag{3.49}$$

The decoder is composed of FFBlock and MaskedMheadSelfAttBlock sublayers. The latter is similar to the MheadSelfAttBlock which masks out future context. The decoder contains an additional multihead cross-attention sublayer to incorporate the representation of the encoder and decoder. Assume a decoder of L_d layers, each is defined as

$$\text{TransDecLayer} = \text{FFBlock} \circ \text{CrossAttBlock} \circ \text{MaskedMheadSelfAttBlock} \tag{3.50}$$

in which

$$\text{CrossAttBlock}(x) = x + (\text{dropout} \circ \text{linear} \circ \text{cross-attention} \circ \text{layer-norm})(x). \tag{3.51}$$

Self-attention weights between target positions using MaskedMheadSelfAttBlock are stated as

$$\alpha_m^{(l+1)}\big(i'|i-1\big) = \text{softmax} \circ \text{DotAtt}(c^{(l)}_{i-1,m}, c^{(l)}_{i'-1,m}) \quad \forall m \in [1, \ldots, H] \tag{3.52}$$

where $i' \in [1, i]$, and $c^{(l)}_{i-1,m}$ is the context vector in the l-th decoder layer and corresponds to the m-th head. Given this, the decoder hidden state at position i for the m-th head is given by

$$s^{(l+1)}_{i-1,m} = \sum_{i'=1}^{i-1} \alpha_m^{(l+1)}\big(i'|i-1\big)\, c^{(l)}_{i'-1,m}. \tag{3.53}$$

Given the final encoder and decoder states at each layer, cross-attention computes a normalized probability distribution $\alpha_m^{(l+1)}(j|i-1)$ over the input positions. This distribution can be considered as an alignment between source and target sentences and is defined as

$$\alpha_m^{(l+1)}(j|i) = \text{softmax} \circ \text{DotAtt}(s^{(l)}_{i-1,m}, h^{(L_e)}_{j,m}), \tag{3.54}$$

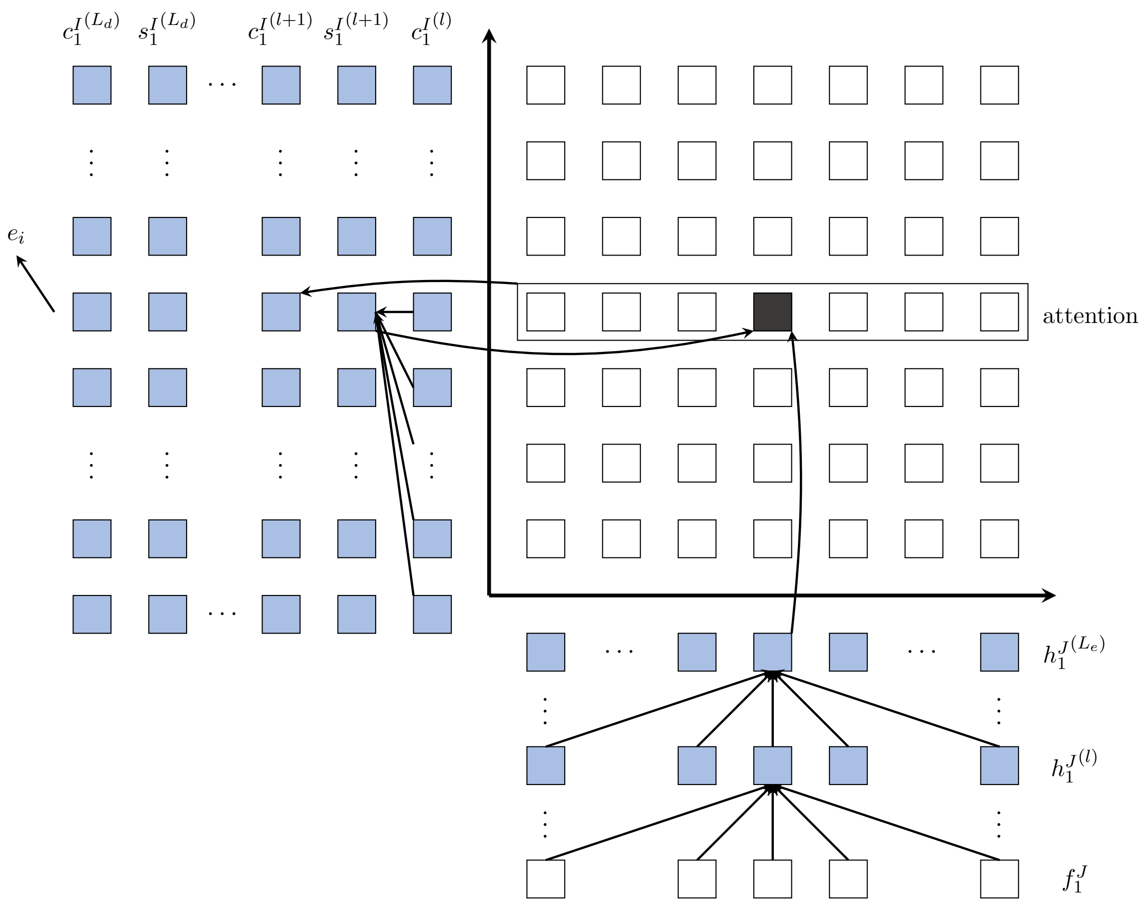


Figure 3.6: The self-attentive architecture composed of L_e stacked TransEncLayer, respectively L_d stacked TransDecLayer. The attention layer represents cross-attention between the source and target sentences. The blue cells highlight the encoder and decoder states. The encoder contains the entire source context (using the MheadSelfAttBlock sublayer), while the decoder only entails to target histories until position $i-1$ (using the MaskedMheadSelfAttBlock sublayer).

and the context vector for each head is derived by

$$c_{i,m}^{(l+1)} = \sum_{j=1}^{J} \alpha_m^{(l+1)}(j|i) h_{j,m}^{(L_e)} \quad \text{with} \quad c_{i,m}^{(0)} := \tilde{e}_i. \tag{3.55}$$

The final attention context vector is obtained by concatenating the representations of all heads given by

$$c_i^{(l+1)} = [c_{i,1}^{(l+1)}; \ldots ; c_{i,H}^{(l+1)}]. \tag{3.56}$$

Again, the decoder is often the stack of L_d layers. Therefore, for a deep model, we obtain

$$c_i^{(L_d)} = (\text{layer-norm} \circ \text{TransDecLayer}^{(L_d)} \circ \ldots \circ \text{TransDecLayer}^{(1)})(e_1^{i-1}). \tag{3.57}$$

The output layer finally generates the next target token as

$$p(e_i|e_0^{i-1}, f_1^J) = \text{softmax} \circ \text{linear}(c_i^{(L_d)}). \tag{3.58}$$

Figure 3.6 summarizes the dependencies within the encoder and decoder self-attentive layers. The model is usually composed of deep encoder and decoder layers. The attention layer refers to cross-attention between the source and target sequence. Although the model is known as the transformer architecture, we use our terminology and call it the self-attentive model in this work.

3.6 Training

Neural network training involves solving a non-convex optimization problem with no closed-form analytical solution in which the parameters θ are numerically estimated. In training neural networks, the goal is to either minimize or maximize a given objective function $\mathcal{L}(\theta)$ by iteratively modifying θ in the hope that it improves an intractable performance measure evaluated on the training set. This is where a set of training examples for supervised learning plays an important role. This is different from traditional optimization algorithms where the cost function is minimized as a final goal in and of itself. In neural networks, optimization is often implied in terms of minimizing an objective function or criterion[2], hence it is also called the cost, loss, or error function. Typically, the cost function can be defined as an average over training data; however, we prefer to minimize the cost function where the expectation is taken over the true underlying distribution. Since the true distribution is unknown in practice, we replace it with the empirical distribution defined by the annotated training set. For a given corpus of $R = |data_{train}| \in \mathbb{N}$ parallel training examples $data_{train} = \Big[\Big(F_r = (f_r)_1^{J_r}, E_r = (e_r)_1^{I_r}\Big)\Big]_{r=1}^{R}$, we define

$$\begin{aligned}\mathcal{L}(\theta) &= \sum_{F,E} Pr(F,E) \log p\big(E|F;\theta\big) &(3.59)\\ &= \frac{1}{R}\sum_{r=1}^{R} \log p\big(E_r|F_r;\theta\big). &(3.60)\end{aligned}$$

Formally, the training criterion is defined to minimize the cost function to find the optimal set of parameters $\hat{\theta}$ given by

$$\begin{aligned}\hat{\theta} &= \operatorname*{argmin}_{\theta} \big\{ -\mathcal{L}(\theta) \big\} &(3.61)\\ &= \operatorname*{argmin}_{\theta} \Big\{ -\sum_{F,E} Pr(F,E) \log p\big(E|F;\theta\big) \Big\} &(3.62)\\ &= \operatorname*{argmin}_{\theta} \Big\{ -\sum_{F} Pr(F) \sum_{E} Pr(E|F) \log p\big(E|F;\theta\big) \Big\} &(3.63)\\ &= \operatorname*{argmax}_{\theta} \Big\{ \sum_{r=1}^{R} \log p\big(E_r|F_r;\theta\big) \Big\}. &(3.64)\end{aligned}$$

By crossing out the terms that are consistent with respect to θ, we simplify Equation 3.62 to 3.63. In information theory, the term $-\sum_E Pr(E|F) \log p\big(E|F;\theta\big)$ refers to *cross-entropy.* Here, the training criterion is either to minimize cross-entropy or to maximize negative cross-entropy, which is equivalent to maximizing log-likelihood of training data written in Equation 3.64 [Ney 19].

3.6.1 Back-Propagation with Stochastic Gradient Descent

The optimization phase of neural networks involves two passes. Starting with a random initialization of the parameters and a given input, the network output is driven by the forward

[2]Maximization can be done by minimizing $-\mathcal{L}(\theta)$.

pass. Given a predefined differentiable cost function, a new set of parameters is obtained in a backward pass according to gradients. The gradient of $\mathcal{L}$ is a vector containing all partial derivatives, denoted as $\nabla_\theta \mathcal{L}(\theta)$. The k-th element of the gradient represents the partial derivative of the cost function with respect to the k-th parameter, which determines how to move θ_k in the opposite sign of the derivative to decrease the cost function. That is why the method is known as gradient descent [Cauchy 47]. As computing the cost function over the entire training data with size R is usually expensive, in practice, we randomly select a small number of samples as a (mini-) batch $B \subset \{1, \dots, R\}$ and take the average over them. Given the cost function in Equation 3.60, the gradient is therefore given by

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{B} \nabla_\theta \sum_{r=1}^{B} \log p\big(E_r | F_r; \theta\big). \tag{3.65}$$

Optimization algorithms that employ batches of training data are not deterministic and are thus called (batch) stochastic gradient descent (SGD), resulting in mini-batch gradient calculations [Robbins & Monro 51]. Larger batches provide a more accurate estimate of the gradient, whereas smaller sizes return more updates and have a more regularizing effect due to noisy gradient calculation [Goodfellow & Bengio$^+$ 16, Wilson & Martinez 03]. SGD updates the parameters by means of an initial learning rate η. It is a positive scalar determining how large the updates are. The update rule is then defined as

$$\theta_{\tau+1} \leftarrow \theta_\tau - \eta \nabla_\theta \mathcal{L}(\theta) \tag{3.66}$$

where the index τ represents the iteration step and $\theta_{\tau+1}$ is a new set of parameters.

The initial learning rate is one of the important hyperparameters of training that should be carefully tuned. Unlike SGD, adaptive optimizers offer a separate learning rate for each parameter. One of the most prominent optimizers widely used in sequence-to-sequence modeling is Adam [Kingma & Ba 15]. Allowing for more complex convergence and robustness against stochasticity, Adam with its learning rate annealing scheme enables fast convergence and finds better local minima [Bahar & Alkhouli$^+$ 17]. Algorithm 1 explains Adam update rule. It considers the decaying average of the past gradients n_τ and provides a decaying mean of past squared gradients m_τ. $\hat{m}_\tau$ and $\hat{n}_\tau$ are the bias-corrected terms for instability against zero initialization considered as the first and second moments. In the case of Adam, β_1, β_2, and ϵ are additional hyperparameters which are usually set to 0.9, 0.999, and 10^{-8}, respectively. The initial learning rate depends highly on the model. Throughout this thesis, the Adam optimizer is applied with the hyperparameters above unless otherwise specified.

Algorithm 1 Adam

```
1: $n_\tau \leftarrow \beta_1 n_{\tau-1} + (1-\beta_1)\nabla_\theta \mathcal{L}(\theta)$
2: $\hat{n}_\tau \leftarrow \frac{n_\tau}{1-\beta_1^\tau}$
3: $m_\tau \leftarrow \beta_2 m_{\tau-1} + (1-\beta_2)\nabla_\theta^2 \mathcal{L}(\theta)$
4: $\hat{m}_\tau \leftarrow \frac{m_\tau}{1-\beta_2^\tau}$
5: $\theta_{\tau+1} \leftarrow \theta_\tau - \frac{\eta}{\sqrt{\hat{m}_\tau}+\epsilon}\hat{n}_\tau$
```

To compute the gradient of the defined cost function with respect to the parameters, a method called back-propagation is used [Rumelhart & Hinton$^+$ 88]. The back-propagation algorithm simply applies the chain rule of calculus to compute all necessary derivatives one after another.

RNNs are trained using the same concept as feedforward networks. They use back-propagation with SGD to update the parameters, though with the added sequential component. Computing the gradient needs to start at the end of time and move backward in time to step 1. Since the gradient values are propagated backward through time, this variant of back-propagation is called back-propagation through time [Robinson & Failside 87, Werbos 88]. Conceptually, this method is not complicated, as recurrent networks are inherently sequential, and the computational graph is unfolded over time. The gradients are computed for each weight in each time step, and finally, the gradients are combined together, after which SGD is applied. However, to simplify the gradient computations for long sequences, a truncated variant [Werbos 90] can be conducted in which the time history is limited to the last few time steps.

3.7 Search

As stated, there are four main ingredients for data-driven methods. We have already discussed the various performance measures that are dependent on the task in Section 1.5. Probabilistic models with suitable structures to capture the dependencies within and between input and output sequences are the second aspect to be considered. All models addressed in this chapter can be used to model $p(e_1^I|f_1^J)$ for a sentence pair (e_1^I, f_1^J). The training criterion to learn the free model parameters is cross-entropy, which is correlatively linked to performance measures like perplexity. Training such models involves an efficient numerical algorithm like SGD.

So far, we have addressed how (neural) probabilistic models define the output posterior probability. To use such definitions, both the source and target sentences have to be given. However, obtaining probability scores is not the ultimate goal in many sequence-to-sequence tasks like machine translation; rather it is to generate a target sentence given the source. The task of finding the most plausible target hypothesis with the maximum probability score according to the model is search. This is the fourth ingredient using the Bayes' decision rule written in Equation 1.1. The sequence posterior probability can be decomposed using the chain rule left-to-right such that the final score is the product of probabilities at each position, i.e.,

$$p(e_1^I|f_1^J) = \prod_{i=1}^{I} p(e_i|e_0^{i-1}, f_1^J). \tag{3.67}$$

Since $p(e_i|e_0^{i-1}, f_1^J)$ is less than one, the product gets smaller for longer sequences, thus the sequence probability scores ascend with increasing target length I. Length normalization is a practical remedy to avoid assigning high scores to short sequences and disadvantaging the longer ones. Formally, the decision rule is expressed by

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \operatorname*{argmax}_{I, e_1^I} \left\{ \frac{1}{I} \prod_{i=1}^{I} p(e_i|e_0^{i-1}, f_1^J) \right\}. \tag{3.68}$$

Let us assume an auxiliary quantity for each unknown partial string e_0^i, where e_0 is the artificial beginning-of-sentence token at the first position. Then, this quantity at position i is given by

$$Q(i; e_0^i) := \prod_{i'=1}^{i} p(e_{i'}|e_0^{i'-1}, f_1^J). \tag{3.69}$$

The search algorithm expands partial hypotheses from e_0^{i-1} to e_0^i for all positions $i = 1, \ldots, I$. Therefore, it involves a recursive method in which the auxiliary quantity at the current position depends on the previous one, and it can be formulated as

$$Q(i; e_0^i) := p(e_i|e_0^{i-1}, f_1^J) Q(i-1; e_0^{i-1}). \tag{3.70}$$

Doing so from the first target position till the end results in the final score of

$$p(e_0^I|f_1^J) = Q(I; e_0^I). \tag{3.71}$$

Bayes' decision rule requires maximization over all possible target sequences, which is a large search space. The search space grows exponentially with the sequence length. For instance, given the vocabulary size $|V_e|$, there are $|V_e|^{20}$ possible sentences with 20 tokens. A naïve solution is greedy search that picks the single best expansion at each position. Any mistake cannot be corrected later by greedy search as it is already committed to a single path.

To remedy the risk of ignoring a candidate with a higher score in the next position, an efficient and practical algorithm known as *beam search* is widely used [Jelinek 97]. Beam search narrows down the search space to the top N-best hypotheses, where N is the beam size. It does not keep the single best partial hypothesis as it passes N possible candidates to the next time step. Beam search organizes the search tree of the partial hypotheses e_0^i with the score $Q(i; e_0^i)$, which is synchronous with target positions or known as position (label) synchronous search because hypotheses refer to the same target position, and a subset of them is selected for expansion in the next time step [Ney 19]. To terminate the search path, either the end-of-sentence token is generated, or a maximum number of tokens is reached, which is in practice a factor of the source sequence length.

3.8 Experimental Results

In this section, results for the neural sequence-to-sequence models are presented on different tasks. This mostly includes our baselines based on the RNN attention and the self-attentive architectures. We refer to these baseline results in the next chapters. All evaluated experiments are carried out within the RWTH extensible training framework for universal recurrent neural networks (RETURNN) [Doetsch & Zeyer+ 17, Zeyer & Alkhouli+ 18], which provides a single implementation for almost all needed features for running experiments on neural models. The publicly available RWTH Aachen University speech recognition toolkit (RASR) [Rybach & Gollan+ 09, Wiesler & Richard+ 14] is also employed to read the speech signal and extract frame-wise features with an integrated interface in RETURNN. A comprehensive discussion of empirical results and a comparison of the models with state-of-the-art methods from the literature on large-scale tasks are provided.

3.8.1 Machine Translation

Setups

Experiments are carried out on the WMT 2018 German→English shared translation task on which we do all ablation studies for machine translation. The corpora statistics are given in Appendix A.1.1 with full details in Table A.1. All systems are trained on all bilingual data with 5.9M sentence pairs. `newstest2017` and `-2018` are selected as the blind test sets. `newstest2015` is used as the development set to tune hyperparameters. The training data is shuffled and split into 8 subepochs, and the models are evaluated every 750k steps.

We also perform experiments on the WMT 2018 Chinese→English task with larger training data, including 17M sentence pairs. Appendix A.1.2 gives details about data with its statistics in Table A.2. `newsdev2017` is used as the development set, while `newstest2017` and `-2018` are used for blind evaluation. The training data is shuffled and divided into 30 subepochs, and each checkpoint is evaluated based on 560k steps.

As the translation research community considers WMT 2014 English→German to be one of the benchmarking tasks, additional experiments are done for the purpose of comparison with other works in the literature. The corpora statistics are given in Appendix A.1.3. The task has almost 4.5M training sentence pairs, where we use almost 4.0M pairs as training data and the rest as the development set (Table A.3). We report the performance of models based on `newstest2013` and `newstest2014`, with more focus on the latter as the benchmark test set. Each subepoch is chosen to have approximately 650k shuffled samples by an epoch split of 6.

For all tasks, to deal with out-of-vocabulary (OOV) and rare words, byte-pair-encoding (BPE) [Sennrich & Haddow$^+$ 16b] is used in order to have a sequence of subwords on either side. All details about data preparation are presented in Appendix. Almost all the experiments in this thesis are conducted on a single Nvidia 1080 GPU. The best checkpoints are chosen based on Bleu and Ter scores of the development sets for all tasks.

WMT 2018 German→English The RNN attention model has been described in Section 3.5.2. It has an embedding dimension of 512 on either side. The encoder is the stack of 6 bidirectional LSTM layers of size 1024. The concatenation of the forward and backward LSTMs with a total dimension of 2048 is used as the encoder. The decoder is composed of one unidirectional LSTM layer of size 1024. The network is equipped with various attention scoring functions extended with the refinements described in Section 3.5.3 with either single or 8 heads. In the case of 8 attention heads, we split the encoder state vectors on the feature axis to have 256-dimensional vectors for each head. The attention layer is applied to each, resulting in 8 distributions. Computing 8 different context vectors, we concatenate them for the final representation of size 2048.

The Adam update rule uses an initial learning rate of 0.001 and reduces it by a factor of 0.9 if the development perplexity does not improve for 6 consecutive checkpoints. Controlling the learning rate, we wait for 3 epochs before decreasing it. We batch the sequences with similar lengths to avoid huge padding with a batch size of 3k tokens. To increase the effective batch size for better convergence [Popel & Bojar 18], gradients are accumulated by a factor of 10. We note that the batch size might differ in various setups, but as a general rule, it is usually specified to be as big as the memory of GPUs allows. Dropout is applied on almost all layers at a rate of 0.3, except the embeddings and attention components. The output label smoothing is set to 0.1. No gradient clipping and no gradient noise are applied. Sequences longer than 100 subwords are removed during training. Embedding matrices are not shared with the projection layers prior to softmax, and no weight-tying is used.

To have better convergence for deep RNN models, we use a layer-wise construction strategy for the first steps [Zeyer & Alkhouli$^+$ 18]. We start with only one bidirectional LSTM layer in the encoder, train the built network for 5 subepochs (approx. 3.7M batches), and add new layers iteratively during training. The label smoothing is deactivated in layer-wise construction.

Decoding is done using beam search with a beam size of 12. The stop criterion is either the end-of-sentence token generation or the maximum length of hypotheses, which is chosen to be 3 times longer than the source sentence. The final hypotheses from all beams are normalized by their length to avoid bias towards shorter candidates.

If not noted otherwise, the self-attentive model for the experiments on machine translation is mainly based on the *base transformer* topology in [Vaswani & Shazeer$^+$ 17]. In its setup, both the encoder and decoder are composed of 6 stacked layers. The multihead attention has 8 heads, and the internal hidden dimension of the model is set to 512 nodes all over the network except for each feedforward layer, which is set to 2048. This internal hidden dimension often refers to the model dimension noted by d as before.

For the sake of fair comparison, we also report the results on a larger model. When the model size is increased to 1024 denoted by $d = 1024$, we correspondingly set the number of heads to 16 and the feedforward hidden dimension to 4096; however, the number of the encoder and decoder layers remains at 6. This setup is often called *big transformer*.

Table 3.3: Comparison of the attention scoring functions on WMT 2018 German→English.

model	attention function	heads	dev		newstest2017		newstest2018		# parameters
			BLEU [%]	TER [%]	BLEU [%]	TER [%]	BLEU [%]	TER [%]	
RNN attention	additive	1	31.6	55.3	32.6	54.9	39.6	46.8	212M
		8	31.6	55.3	32.6	55.2	39.6	46.9	217M
	scaled multiplicative	1	31.9	55.3	32.7	55.1	39.9	46.5	217M
		8	31.8	55.4	32.5	55.3	39.8	46.8	217M

The models are trained using Adam with a learning rate of 0.0003 and no warm-up technique. We randomly batch the sequences with a batch size of 4k to 5k tokens and accumulate the gradient by factors of 4 to 8. No gradient clipping and no gradient noise are applied. During training, we use label smoothing of 0.1 and dropout of 0.1 on almost all components: the embeddings, multihead attention, feedforward, and postprocessing layers. The learning rate is lowered with a decay factor of 0.9 if the perplexity on the development set does not improve for 6 consecutive checkpoints. For the self-attentive models, no layer-wise construction is applied.

WMT Chinese→English The RNN attention model has the same configuration as used for the German→English task. For the self-attentive-based models, we apply the warm-up technique of 10 steps at the beginning of training in which the initial learning rate gradually increases from 8×10^{-5} to 3×10^{-4}. A batch size of 4k subwords and a gradient accumulation of factor 5 are used. For the larger model of size 1024, the batch size has to be reduced by a factor of 2, i.e., 2k tokens, and accumulate the gradient for 15 steps.

WMT 2014 English→German The model structures are as before, except that we wait for 20 subepochs to lower the learning rate.

Results

Attention Scoring Functions Table 3.3 shows the results on WMT German→English for main attention functions addressed in Section 3.5.1. To reveal the difference between the models, we also report the number of parameters in the right-most column. We compare two main attention functions with either a single head or 8 heads, where the encoder and the decoder are modeled based on the LSTM layers. By comparing lines 1 and 3, the scaled multiplicative attention function leads to a small improvement of 0.3% in BLEU and TER on `newstest2018`, but inconsistent improvements over all test sets. This may partly be due to noise and optimization artifacts. There is no real benefit in using multihead attention in the RNN models. This observation is in contrast to the self-attentive-based architecture leveraging multihead self-attention layers. It also uses feedforward layers with residual connections, dropout, and layer normalization. These may be significant aspects of stabilized training in such architectures.

Attention Refinements We also compare the extensions of the attention layer developed on top of the additive attention function, as described in Section 3.5.3. Table 3.4 reports the results of this comparison. The single-head additive attention model is taken as the baseline listed in the first line of the table. Neither the last weight nor accumulating all weights brings an improvement. Adding the last context vector c_{i-1} in the attention scoring function performs as good as the RNN attention model (cf. lines 1 and 4), while the fertility concept (line 5) slightly boosts the performance of the pure additive function by 0.2 points in BLEU on average across all sets. Combining the last context vector in the decoder with fertility (line 6) helps TER, whereas

Table 3.4: Comparison of the attention refinements on WMT 2018 German→English.

model	dev		newstest2017		newstest2018		#
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]	parameters
additive attention	31.6	55.3	32.6	54.9	39.6	46.8	212M
+ last weight	31.6	55.4	32.7	55.0	39.6	46.6	212M
+ accumulated weights	31.5	55.3	32.5	54.9	39.6	46.8	212M
+ last context in attention	31.8	55.4	32.4	55.2	39.7	46.7	213M
+ fertility	31.9	55.3	32.9	55.2	39.7	46.8	217M
+ last context in decoder	32.0	55.2	32.8	55.0	39.8	46.6	226M
+ embedding in attention	31.7	55.6	32.6	55.3	39.6	47.1	252M

including embedding vectors in the attention component does not yield better results. However, the last two require slightly more parameters. In general, adding attention variants to the pure additive function leads to mixed results. It seems the performance of the attention component is almost saturated, and such variants can achieve no further notable gains. Nevertheless, fertility as expressed in Equation 3.34 is included in the RNN attention model as it has a small impact. Therefore, in the rest of this thesis, all the RNN attention models include fertility in their attention component; correspondingly, the combination of fertility and the RNN attention model (line 5) is referred to as the RNN attention baseline model in all translation tasks.

Comparison of the Fundamental Architectures We build two baselines based on the RNN attention and the self-attentive models and compare all other methods with them in the following chapters. The self-attentive model consistently outperforms the RNN attention model; however, the number of parameters is lower. The results of this comparison are shown in Tables 3.5, 3.6, and 3.7 for the German→English, Chinese→English, and English→German translation tasks. Increasing the self-attentive model size to 1024 performs best, as expected.

Table 3.5: The baseline results on WMT 2018 German→English. The RNN attention model represents the additive attention model with fertility (line 5 in Table 3.4). $d = 1024$ shows the model size of the self-attentive model that corresponds to the big architecture in the original paper [Vaswani & Shazeer$^+$ 17].

model	dev		newstest2017		newstest2018		#
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]	parameters
RNN attention	31.9	55.3	32.9	55.2	39.7	46.8	217M
self-attentive	32.4	55.8	33.2	55.3	40.4	46.8	102M
+ $d = 1024$	33.2	54.9	34.2	54.5	41.0	46.5	291M

Table 3.6: The baseline results on WMT 2018 Chinese→English.

model	dev		newstest2017		newstest2018		#
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]	parameters
RNN attention	21.2	63.1	23.2	61.1	22.7	65.9	217M
self-attentive	22.0	62.8	23.1	61.7	23.0	66.1	101M
+ $d = 1024$	22.7	61.7	23.5	61.0	23.8	65.3	290M

Table 3.7: The baseline results on WMT 2014 English→German.

model	newstest2013		newstest2014		#
	BLEU [%]	TER [%]	BLEU [%]	TER [%]	parameters
RNN attention	25.8	56.8	26.9	56.3	224M
self-attentive	26.3	57.0	27.6	56.0	108M

Table 3.8: Comparison of the baselines with the other works recently published on WMT 2014 English→German. Massive hardware (HW) refers to massive GPU budgets to train models: "no" means single-GPU training, while "yes" means multiple GPU access in that the number of nodes/machines can vary.

method	BLEU [%]	#	massive
	newstest2014	parameters	HW
DeepL translator (evaluated in 2020)	33.1	-	-
RNN attention (GNMT) [Wu & Schuster$^+$ 16]	24.7	-	yes
deep RNN attention (RNMT+)[Chen & Firat$^+$ 18]	28.5	378.9M	yes
CNN attention [Gehring & Auli$^+$ 17b]	25.2	263.4M	yes
CNN attention (DynamicConv) [Wu & Fan$^+$ 19]	29.7	213M	yes
self-attentive [Vaswani & Shazeer$^+$ 17]	27.3	93.3M	yes
deep self-attentive [Ott & Edunov$^+$ 18]	29.3	210M	yes
deep self-attentive [Liu & Duh$^+$ 20]	30.1	256M	no
RNN attention	26.9	224M	no
self-attentive	27.6	108M	no

"-": not available

Comparison with Other Works There are three main networks for neural sequence-to-sequence modeling: recurrent, self-attention, and convolution. In general, all can be categorized as encoder-decoder networks. Despite the fact that how these components are used in the encoder and decoder, they all employ the attention mechanism. In contrast to RNN models, where recurrency needs sequential computing, in convolution and self-attention layers, computations over all elements can be parallelized during training since there is no dependency on the previous steps. This makes for efficient use of GPUs, an order of magnitude faster speed and eases optimization due to fixed-size contexts. Therefore, they need to model the length of dependencies explicitly, like positional encoding. On the other hand, the multilayer hierarchical structure of such models works with a shorter path to capture long-range dependencies. These have to be equipped with residual connections to encourage gradient flow. This contrasts with RNN models designed to learn long-term dependencies.

The convolutional components are outside the scope of the main contribution of this work when comparing the baselines with CNN-based models from the literature. Table 3.8 compares the baselines with the state-of-the-art works on WMT 2014 English→German. To be comparable to these works, we report tokenized BLEU computed by the `multi-bleu.pl` script from the Moses toolkit [Koehn & Hoang$^+$ 07]. To keep the results of practical interest, the table also shows the GPU budget of each model training.

Google neural machine translation (GNMT) [Wu & Schuster$^+$ 16] applies deep layers of RNN components. The self-attentive model exploits the deep multihead self-attention layers, and the convolutional sequence-to-sequence model [Gehring & Auli$^+$ 17b] is equipped by convolutional layers. In order to have working systems, all models use different techniques and additional components in their topology. The combination of these choices allows us to tackle large-scale problems in various scenarios. A notable mixed architecture is the recurrent neural machine translation model [Chen & Firat$^+$ 18, Werlen & Pappas$^+$ 18], where the model combines a self-attention-based encoder and a recurrent-based decoder. This work successfully builds a deep recurrent-based model with explicit layer normalization and residual connections known as the new recurrent neural machine translation (RNMT+). As the model is deeper equipped with better gradient flow, the results improve by almost 4 Bleu points (cf. Table 3.8 lines 2 and 3).

In [Gehring & Auli$^+$ 17a], a convolutional encoder is fused with a recurrent decoder. Also, two encoders can be trained in parallel, one self-attention and one RNN [Hao & Wang$^+$ 19], or one can put a convolutional decoder together with a recurrent decoder [Lin & Sun$^+$ 18]. However, both combinations lead to no notable gain over their baselines. Encouraged by fully parallelized and fast training, similar to CNN-based attention models, dynamic convolutions (DynamicConv) is introduced [Wu & Fan$^+$ 19] in which separate convolution kernels determine the importance of context per position (row 5 in the table).

Many recent works aim to make deeper and larger self-attentive models with small modifications for stable training in a tractable time. In [Ott & Edunov$^+$ 18], deep models are trained while improving training time. They apply very large batches with lower floating-point precision computation. The model architecture gets even bigger with 60 layers in the encoder and 12 layers in the decoder [Liu & Duh$^+$ 20]. They apply an effective initialization technique and show that their deep model can be trained on a single GPU. Building deep yet possible models on limited hardware, their design choice might greatly impact the research community.

As seen in the table, our RNN attention model outperforms the Google recurrent model by 2.2 points in Bleu, and it is also better than the CNN model (line 4). Our self-attentive model is on par with the one from the original paper. As soon as the models get deeper, their performance increases regardless of their base architectures. On top of all models, the DeepL translation engine (evaluated in 2020) outperforms all mentioned systems since it probably uses cleaner training data.

3.8.2 Automatic Speech Recognition

Setups

We conduct the experiments on two well-known tasks: Switchboard 300h and LibriSpeech 960h. Information about Switchboard is given in Appendix A.2.1. We use 40-dimensional Gammatone features [Schlüter & Bezrukov$^+$ 07] extracted using RASR and subword BPEs with 1k symbols on the output side. Following a common practice, word error rates are measured on `Hub5'00` with both its `Switchboard (SWB)` and `CallHome (CH)` parts as well as on `Hub5'01`, while `Hub5'00` is used as the development set to select the best checkpoint, tune language model scale, etc., and `Hub5'01` is considered as the blind test set.

For the LibriSpeech task, the 40-dimensional mel-frequency cepstral coefficient (MFCC) features [Davis & Mermelstein 80] are extracted using the `librosa` library [McFee & McVicar$^+$ 17]. BPE of 10k merge operations is used. Appendix A.2.2 provides information about the corpus. The checkpoint selection is based on both `dev-clean` and `dev-other`. The total number of epochs over the training data is either 12.5 or 25 full epochs. Specifically, once the best checkpoint has been found from the first training run, training is resumed by resetting Adam hyperparameters to the initial values and continue training for 12.5 more epochs resulting in 25 iterations over the entire training data. A simple curriculum learning is also applied in which for the first 20 subepochs, the clean part of the LibriSpeech corpus, i.e., `clean-100` and `clean-360` is only used. Additionally, the sequences are sorted according to their length. In this thesis, we use a variant of spectrogram augmentation [Park & Chan$^+$ 19] for all neural ASR models. The concept of spectrogram augmentation (SpecAugment) is covered in more detail later in Section 6.2.2.

Switchboard 300h The acoustic model architectures are based on both the RNN and self-attentive models. Like the machine translation setting, the models are based on a deep bidirectional LSTM encoder with 6 layers of 1024 units for each forward and backward direction. The frame-wise speech input sequence is much longer than source sentences in MT as the feature extraction is done every 10ms. To reduce the frame-wise speech sequence length, max-pooling is applied in the time dimension. The higher the reduction factor, the simpler training is, and less memory is required [Zeyer & Irie$^+$ 18, Zeyer & Merboldt$^+$ 18]. However, a very large reduction factor has shown to reduce the performance. Therefore, a compromise between the time reduction factor and recognition performance needs to be considered. It is common to directly use time reduction on the feature vectors [Zeyer & Irie$^+$ 18, Zeyer & Merboldt$^+$ 18]. We also apply max-pooling between the first and second BiLSTM layers with a factor of 3 and 2, respectively resulting in a total of 6. For the RNN attention models, we follow a similar setup to [Zeyer & Irie$^+$ 18] where we combine layer-wise construction in the encoder and time reduction with a larger factor for a few subepochs. We then switch to the default setup with a final total time reduction factor of 6 once the entire network has been built. Concurrently, the hidden dimension of LSTM is linearly incremented where the network starts with a small hidden size and grows to 1024 while training. A 512-dimensional word embedding is used. The Adam optimizer trains the model with an initial learning rate of 0.001 with 10 linear warm-up steps. The minimum learning rate after decaying is chosen to be 50 times lower than the initial learning rate. The batches include 20k frames and a maximum output sequence length of 75 tokens. Gradients are neither accumulated nor clipped. We wait for at least 3 subepochs to decay the learning rate by a factor of 0.7 and reduce it based on the development perplexity of the last 6 subepochs. Dropout and label smoothing with a rate of 0.3 and 0.1 are used. If not noted otherwise, we iterate for 33 epochs over the training data. Decoding uses a beam size of 12. The output generation is stopped as soon as the maximum length of the hypothesis reaches the input sequence length or the end-of-sentence token is predicted.

The main configuration of the self-attentive model is similar to those in machine translation, except that we include two bidirectional LSTM layers of size 1024 interleaved with max-pooling layers on top of the feature vectors. The reason for this is to preserve the order of a sequence and reduce the time sequence, as done in the RNN attention model. They are also deeper in the ASR setups. All self-attentive models use 12 self-attention encoder and decoder layers. The dimension of feedforward layers of the self-attentive models is chosen to be 1024. The self-attentive internal model size and embeddings are 512. Dropout on all components is used with a rate of 0.3, except on the embedding with a rate of 0.2. The label smoothing softmax is set to 0.1. The models are trained using Adam with an initial learning rate of 0.0008 with a decay factor of 0.9 if the perplexity on the development set does not improve for 6 consequent checkpoints. A warm-up of 10 steps at the beginning is taken, and the minimum learning rate is selected to be 50 times smaller than its initial value. In contrast to the MT setup where the layer-wise construction scheme is not used within the self-attentive model, we do use it for speech recognition [Zeyer & Bahar$^+$ 19]. Besides layer-wise construction of the encoder, we also grow the decoder layers as it is deeper than the RNN attention models. In this scheme, we start with two layers of encoders and decoders each and double the number of layers after every construction iteration. Simultaneously, the hidden dimensions are increased linearly. After observing from initial experiments, the self-attentive models are trained a bit longer for a maximum of 50 epochs. No positional encoding in the encoder or decoder is used, as suggested in [Irie & Zeyer$^+$ 19]. In contrast to the RNN models, the connectionist temporal classification (CTC) loss is added on top of the encoder for the self-attentive models. More detail on the CTC loss is given in Section 6.2.1 for speech translation.

LibriSpeech 960h The same RNN attention acoustic model is used for LibriSpeech. The only difference is that we use an additional CTC loss.

Table 3.9: The language model subword-level perplexities on Switchboard and LibriSpeech. For LibriSpeech, dev and test correspond to the concatenation of `dev-clean` and `dev-other` sets, respectively `test-clean` and `test-other`.

LM	**Switchboard**		**LibriSpeech**	
	Hub5'00	**Hub5'01**	**dev**	**test**
RNN	25.3	22.1	45.0	47.1
self-attentive	23.4	20.4	37.3	38.9

Table 3.10: Comparison of the attention scoring functions on Switchboard. Either a single head or eight heads in the recurrent attention model is used. AM and LM abbreviate acoustic and language models, respectively.

AM			**LM**	WER[%]			
model	**attention function**	**heads**		**Hub5'00**			**Hub5'01**
				CH	**SWB**	**Σ**	
RNN attention	additive	1	none	22.0	10.9	16.4	16.0
		8		22.4	11.1	16.8	16.0
	scaled multiplicative	1		21.6	10.7	16.2	15.6
		8		21.4	10.6	16.0	15.4

The self-attentive model has the same architecture, with 12 encoder and decoder layers of size 512. The dimension of feedforward layers is set to 2048. Here, we use a smaller dropout rate of 0.1 and no embedding dropout. The effective batch size is chosen to be 100k with a regular size of 25k and gradient accumulation of 4 batches.

In order to decrease the recognition error, external language models are also applied during decoding. Appendices A.2.1 and A.2.2 address brief information without going into detail on the language model training. For more detail about how the language models are trained, we refer to [Irie 20, Zeyer & Irie$^+$ 18, Zeyer & Bahar$^+$ 19]. Table 3.9 shows the subword-level perplexity of language models for the both tasks.

Results

Attention Scoring Functions Similar to Section 3.8.1, we measure the performance of speech recognition models trained with either the additive or multiplicative attention functions with single or 8 heads. Table 3.10 shows that the multiplicative function is slightly better by 0.3% absolute on average when single-head attention is used. Moving from one head to 8 improves WER with the multiplicative function, whereas it slightly decreases the performance of the additive function.

Attention Refinements Since we can easily extend the attention function with previous attention information, we tend to investigate whether such extensions are also helpful for the ASR task. The results shown in Table 3.11 indicate that equipping the attention function with fertility leads to the best WER. It outperforms the pure attention function by 1% absolute WER on `Hub5'01`. By comparing line 5 in Table 3.11 and line 4 in Table 3.10, an improvement of 0.4% absolute WER over the 8-head multiplicative attention function is seen. This is consistent with our observations on the MT experiments. In addition to recognition quality in terms of WER, the number of free parameters is reported for each model. As seen, the improvement in WER comes at the cost of

Table 3.11: Comparison of attention refinements on Switchboard.

AM	LM	WER[%]				#
		Hub5'00			Hub5'01	parameters
		CH	SWB	Σ		
additive attention	none	22.0	10.9	16.4	16.0	152M
+ last weight		21.5	10.6	16.0	15.6	152M
+ accumulated weights		22.0	10.9	16.5	16.0	152M
+ last context in attention		23.1	11.2	17.2	16.7	153M
+ fertility		21.1	10.1	15.6	15.0	157M
+ last context in decoder		21.7	10.5	16.1	15.5	165M

Table 3.12: The baseline results on Switchboard. Training speed is the average over 10 epochs reported with the number of characters per second. AM and LM are combined by shallow fusion during decoding as expressed in Equation 3.72.

AM	LM	WER[%]				training speed	#
		Hub5'00			Hub5'01	[char/sec]	parameters
		CH	SWB	Σ			
RNN attention	none	21.1	10.1	15.6	15.0	2.7k	157M
	RNN	20.6	9.2	14.9	14.1		
	self-attentive	20.2	9.1	14.7	13.9		
self-attentive	none	21.2	10.3	15.8	15.3	3.2k	99.5M
	RNN	20.7	9.6	15.2	14.3		
	self-attentive	20.7	9.5	15.1	14.0		

only 5M more parameters (see lines 1 and 5). Analogously to machine translation, we use additive attention with fertility as the RNN attention baseline for all ASR tasks throughout this thesis.

Comparison of the Fundamental Architectures Table 3.12 presents the Switchboard results of the models used as the baselines with their training speed. We observe that both models have a better performance on the easier `SWB` subset than on `CH`. In general, the recurrent model negligibly outperforms the self-attentive by 0.2%, while the latter has almost 57M fewer parameters and provides faster training (lines 1 and 4). It is also important to note that the models are still behind the hybrid HMM models, with WER of 10.7% on the average of `Hub5'00` [Kitza & Golik+ 19].

The results on LibriSpeech can be seen in Table 3.13. On this task, the performance of the two models is on par on average. In the first set of experiments, we deliberately limit the number of training epochs to 12.5 for a fair comparison of internal results. To keep the results of practical interest and bridge the gap with external research groups, training is continued from the best checkpoint for 12.5 more epochs. We note that this is still much less than what is reported in the literature, for example the models are trained for over 600 full epochs in [Park & Chan+ 19], taking about 24 days on 32 tensor processing units (TPUs) [Jouppi & Young+ 17]. Longer training is found to be effective in improving results such that on `test-other`, we further gain 0.9% and 0.8% absolute WER for the RNN and self-attentive models, respectively. Following this observation, unless explicitly stated otherwise, the models are trained for 25 full epochs for LibriSpeech.

Language Model Combination We also integrate the language models in decoding by using the shallow fusion technique [Gülçehre & Firat+ 15, Toshniwal & Kannan+ 18], where the scores

Table 3.13: The baseline results on LibriSpeech. The number of epochs is chosen to be 12.5 or 25.

AM	**# epochs**	**LM**	WER[%]				**# parameters**
			dev		**test**		
			clean	**other**	**clean**	**other**	
RNN attention	12.5	none	4.1	11.1	4.3	11.7	188M
		RNN	2.9	8.3	3.2	9.1	
		self-attentive	2.8	7.7	3.1	8.1	
	25.0	none	3.7	10.2	3.8	10.8	
		RNN	2.8	7.7	3.2	8.4	
		self-attentive	2.6	7.2	2.7	7.7	
self-attentive	12.5	none	4.4	11.1	4.5	11.5	139M
		RNN	2.9	8.3	3.2	8.8	
		self-attentive	2.8	7.5	3.1	8.1	
	25.0	none	3.9	10.2	4.1	10.7	
		RNN	2.9	7.6	3.2	8.1	
		self-attentive	2.7	7.3	3.1	7.7	

of the acoustic and language models are combined on the probability level, as in the log-linear combination. Such a fusion method requires the same vocabulary for both models, similar to ensemble-like integration. Formally, the combined score is defined as

$$x_1^T \to \hat{w}_1^{\hat{N}} = \operatorname*{argmax}_{N,w_1^N} \left\{ \sum_{i=1}^{N} \log p_{\text{AM}}(w_n|w_0^{n-1}, x_1^T) + \lambda_{\text{LM}} \log p_{\text{LM}}(w_n|w_0^{n-1}) \right\} \tag{3.72}$$

where λ_{LM} is the language model scale. Here, the scale of acoustic attention models is chosen to be always 1, while the scale of the language model is optimized by a grid search based on the development set.

We fuse with both the RNN and self-attentive language models. The comparison of LM perplexities can be found in Table 3.9. As it has been proven, there is a high linear correlation between perplexities and word error rates [Bahl & Jelinek$^+$ 83, Sundermeyer & Ney$^+$ 15, Ney 19]. Since the self-attentive-based LMs obtain better perplexity on both tasks (see Table 3.9), we hence expect more reduction in recognition error by fusing the self-attentive model rather than RNN. First of all, as shown in Tables 3.12 and 3.13, language model combination has a large impact on the performance of the systems and consistently improves the recognition performance. As expected, switching the language models from RNN to self-attentive leads to a slightly better error rate due to lower perplexity.

Another observation is the amount of improvement that each language model brings. As presented in these tables, language model fusion brings greater improvements in LibriSpeech than Switchboard. For LibriSpeech, the language models have been trained using an additional data of 800M words fully independent of 10M words corresponding to the transcription of 1000 hours of audio. The extra data is 80 times larger. In contrast, in the case of Switchboard, we utilize 27M words, including 3M words overlapped with the transcripts for language model training. As the domain is conversational telephone speech, it is harder to acquire the in-domain text data compared to the read speech [Zeyer & Irie$^+$ 18]. In fact, this difference in the amount of external data directly affects the performance improvements by using the external language models, as shown by the experimental results depicted in these tables.

Table 3.14: Comparison of the baselines with the recently published works on LibriSpeech. The models have not necessary been trained with a comparable number of parameters over the same number of epochs.

method	**AM**	**LM**	WER$^{[\%]}$		**# parameters**	**# epochs**
			test		**(AM†)**	**(AM†)**
			clean	**other**		
hybrid HMM	RNN [Lüscher & Beck+ 19]	self-attentive	2.3	5.0	160M	12.5
	self-attentive [Wang & Mohamed+ 20]	self-attentive	2.3	4.9	149M	100
CTC	CNN [Kriman & Beliaev+ 20]	self-attentive	2.7	7.3	19M	400
attention	RNN [Park & Chan+ 19]	RNN	2.5	5.8	>200M	600
	RNN [Karita & Wang+ 19]	RNN	3.3	10.8	-	100
	self-attentive [Karita & Wang+ 19]	RNN	2.6	5.7	-	200
	CNN [Hannun & Lee+ 19]	CNN	3.3	9.8	37M	-
	self-attentive [Synnaeve & Xu+ 20]	self-attentive	2.4	5.2	270M	-
transducer	self-attentive [Zhang & Lu+ 20]	self-attentive	2.0	4.6	139M	-
	CNN-RNN [Han & Zhang+ 20]	RNN	1.9	4.1	113M	-
	Conformer [Gulati & Qin+ 20]	RNN	1.9	3.9	119M	-
this work	RNN attention	self-attentive	2.7	7.7	188M	25

"-": not available.
† measures only for the acoustic models.

Comparison with Other Works We also take the LibriSpeech task and compare our best model with a few recently published state-of-the-art models, such as hybrid HMM using recurrent [Lüscher & Beck+ 19] or self-attentive [Wang & Mohamed+ 20] networks, ContextNet [Han & Zhang+ 20], the self-attentive transducer [Zhang & Lu+ 20], and QuartzNet [Kriman & Beliaev+ 20]. One comparison is always to conventional frame-wise hybrid HMMs, as addressed in the introduction. It can be trained using both RNN layers, specifically bidirectional LSTMs or self-attentive layers. In the following, we briefly explain each model and refer to the main papers for further details.

After generating GMM alignments in [Lüscher & Beck+ 19], phonetic classification and regression tree (CART) labels are used for the state-tied phones. The acoustic model is composed of 6 layers of bidirectional LSTMs with 1k nodes. The model also applies sequence discriminative training performed using a lattice-based version of the state-level minimum Bayes risk (sMBR) criterion [Gibson & Hain 06]. A deep self-attentive LM with 96 layers is used in the second-pass rescoring of lattices. In [Wang & Mohamed+ 20], a 24-layer self-attentive model with sMBR training is used for acoustic modeling, and a deep self-attentive LM is employed in the first-pass decoding.

As a special HMM topology, CTC [Graves & Fernández+ 06] models the label sequence posterior directly. It favors integrated observation and alignment model training using neural networks. Due to its simple implementation, CTC training is used in CNN-based acoustic modeling. This model, known as QuartzNet [Kriman & Beliaev+ 20], is structured with convolutional layers followed by normalization layers and ReLu activations, with a small number of parameters at the end. The best-achieved model has been trained for 400 epochs on 8 Tesla V100 GPUs. The LM used in decoding is based on deep self-attentive layers similar to [Dai & Yang+ 19].

Deep RNN attention models are trained with data augmentation [Park & Chan+ 19]. The models are based on deep BiLSTM encoder and unidirectional LSTM layers [Chan & Jaitly+ 16]. Using 32 TPUs, they have been trained for 24 days with a complicated training procedure that turns out to be an essential factor. The language model is selected to be a two-layer LSTM. In the work by [Karita & Wang+ 19], both self-attentive and RNN acoustic models with a 7-layers LSTM-based language model are fused. Building a fully convolutional sequence-to-sequence model with a time-depth separable (TDS) structure [Hannun & Lee+ 19], one can train deep TDS-based

CNN networks on 8 V100 GPUs as the acoustic model fused with a convolutional LM. Having been inspired by the success of self-attentive acoustic and language models, the recognition errors are improved by [Synnaeve & Xu$^+$ 20].

Unlike the end-to-end label-synchronous attention-based methods, which pose a challenge for streaming, transducers are another group of modeling techniques that enable streaming in automatic speech recognition. They are based on the encoder-decoder framework, map audio frame-wise sequences to label representations, and learn the corresponding alignments between them. Such networks offer frame-synchronous decoding and have three components: an audio encoder, a label encoder, and a joint network to combine them and decode. Recurrent neural network transducer (RNN-T) [Graves 12a, He & Sainath$^+$ 19, Tripathi & Lu$^+$ 19] as well as recurrent neural aligner (RNA) [Sak & Shannon$^+$ 17] build the network using recurrent units. However, they can be replaced by self-attentive on a fixed number of past input frames. This eases the computational complexity of the streaming phase. These models consist of 18 and 2 layers of audio and label encoders, respectively trained on 64 TPUs. Shallow fusion is then performed using a 6-layer self-attentive LM [Zhang & Lu$^+$ 20]. Similarly, to make decoding computationally tractable for streaming, CNN-RNN transducers [Han & Zhang$^+$ 20] are proposed that combine the parameter efficiency of CNN and RNN layers in the transducer mode, also called ContextNet. The models have deep and large convolution blocks in the encoder, while the decoder is composed of one LSTM layer. A 3-layer LSTM LM with 4k nodes is applied. Conformer [Gulati & Qin$^+$ 20] is a convolution-augmented self-attentive model where network depth, model dimension, and the number of attention heads are chosen to be smaller. This work uses an additional 3-layer LSTM LM with 4k hidden dimensions for shallow fusion.

Table 3.14 gives the comparison to the other works on LibriSpeech test sets. One reason for reporting the number of training epochs is that we tend to show that the baselines are in the right ballpark, and thus further comparisons under fair conditions make sense. With our restricted hardware, a reasonably good baseline with 7.7% in WER on `test-other` is achieved. Our attention-based model is behind hybrid HMM systems, which are often state-of-the-art ASR models. Compared to the other attention-based models, we are on the correct track considering the number of training epochs. We also observe consistent improvement by increasing the number of iterations (see Table 3.13 from 12.5 to 25 full epochs). Our training takes about one week on a single Nvidia 1080 GPU. This contrasts with the other works in which the best system takes over 600 epochs, which trains for about 24 days on 32 TPUs [Park & Chan$^+$ 19]. There, we deliberately limit the training epochs to 25 for faster development of new models in the next chapters.

3.9 Implementation

The implementation and extensions to the neural network models was started using Blocks [van Merriënboer & Bahdanau$^+$ 15], which is a Python framework on top of Theano [Bergstra & Breuleux$^+$ 10, Bastien & Lamblin$^+$ 12]. In 2017, the development of Theano was stopped, leaving only minimal maintenance of the software. We therefore put an end to our Theano development in Blocks and reimplemented and developed new features in RETURNN[3]. All the experiments in this thesis are based on RETURNN, providing a comprehensive implementation for almost all needed features for running experiments on neural models.

RETURNN is mainly written in Python, and it has two backends, based on open-source software, Theano and TensorFlow [Abadi & Agarwal$^+$ 15]. It supports different neural network architectures like feedforward, recurrent, CNN, and multihead self-attention layers with an easy configuration of experiments. Experiments are configured through a config file determining the building blocks of neural networks and hyperparameters. It constructs the network topology through nested

[3]`https://github.com/rwth-i6/returnn`

dictionaries, where each entry represents a layer specification. All in all, RETURNN provides high flexibility of network architectures, fast training and inference time, and easy access to the individual parts, allowing easy modifications to various experiments. For more details, we refer the reader to its documentation[4].

RETURNN has active development, maintenance, and support based on TensorFlow, as Theano is no longer supported. Similar to Theano, TensorFlow enables two crucial properties for neural network modeling. It provides symbolic features such as automatic differentiation and supports GPU computation. TensorFlow lets us define complicated mathematical expressions symbolically in which the computations can be optimally executed. Symbolic expressions allow TensorFlow to automatically determine derivatives without any extra effort to derive them. Thus, it significantly simplifies the development of new features because only the computations for the forward pass of networks need to be specified, and then TensorFlow automatically carries out the calculation of derivatives. Once symbolic expressions are specified, a representative computation graph is created, and TensorFlow optimizes it, executes it, and takes care of the rest of the computations.

3.10 Conclusion

In this chapter, we mainly covered the background knowledge required to understand the context of this thesis. Specifically, we first reviewed different neural networks as key components of the parameterization utilized in the remaining chapters. After explaining the recurrent language models, we proceeded with the encoder-decoder networks as source-conditioned language models in machine translation and described the state-of-the-art neural architectures. Our contributions were begun by verifying and extending attention refinements in the RNN attention models that lack alignment information from previous positions. Fertility with past alignment information was added to the attention computation for machine translation and further used in speech recognition. This resulted in a marginal gain (Table 3.4) for machine translation, while improving the error rate by 1% absolute on the `Hub5'01` test set for speech recognition (Table 3.11).

Accordingly, we also contributed to building our baselines using the RNN attention and self-attentive models and using them in the rest of this thesis. We experimentally compared them on the publicly available large-scale datasets for both machine translation and automatic speech recognition and showed that our baselines are fairly strong and competitive with results in the literature. The experimental results indicated that the self-attentive model outperformed the RNN attention in machine translation (Tables 3.5 to 3.7) and was on par with it in automatic speech recognition if a moderate amount of training data is available (Table 3.13). Next, we emphasized some of these fundamental models' properties to affirm our design choices and motivate our contributions in the following chapters.

3.11 Related Work

Early attempts in neural sequence-to-sequence models date back to the 1980s and 90s when computational power was a serious obstacle to using deep models [Allen 87, Pollack 90, Chrisman 91, Castano & Casacuberta$^+$ 97, Forcada & Ñeco 97]. Therefore, a small vocabulary with a small model size was the only practical choice. With the development of more powerful machines, deep feedforward neural language models [Bengio & Ducharme$^+$ 03, Bengio & Schwenk$^+$ 06] show promising results, though the context dependency is limited to a few predecessor tokens. To have the entire context and full Markov assumption, recurrent-based language models are introduced [Mikolov & Karafiát$^+$ 10, Mikolov & Kombrink$^+$ 11], with a follow-up extension using LSTM cells

[4]`https://returnn.readthedocs.io/en/latest/`

[Sundermeyer & Schlüter$^+$ 12] that outperform the feedforward and RNN LMs. LSTM-based language models have been the de-facto standard approach in language modeling ever since.

The success of neural networks in language modeling and their application to speech recognition [Schwenk & Gauvain 02, Schwenk & Gauvain 04, Sundermeyer & Oparin$^+$ 13] motivate research in integrating them in machine translation. Deep feedforward neural language models are employed [Schwenk & Déchelotte$^+$ 06, Schwenk & Rousseau$^+$ 12] instead of the count-based models in second-pass decoding to rescore the N-best list [Vaswani & Zhao$^+$ 13] generated by the phrase-based system. Recurrent-based language models are also integrated into the system along with expected Bleu training [Auli & Gao 14]. In addition to target neural language models, there are follow-up works where the source language is also taken into account. In all these methods, neural networks are utilized as one of the components in the phrase-based system in the log-linear model combination. Language models are used to combine with bilingual tuples [Zamora-Martínez & Bleda$^+$ 10], to directly score phrase pairs [Schwenk 12], or to represent the source context in translation modeling with phrases [Le & Allauzen$^+$ 12, Devlin & Zbib$^+$ 14].

There are other extensions in which neural networks are used as translation models, like [Kalchbrenner & Blunsom 13, Auli & Galley$^+$ 13, Hu & Auli$^+$ 14, Sundermeyer & Alkhouli$^+$ 14, Cho & van Merrienboer$^+$ 14b] that shed light on stand-alone neural machine translation trained end-to-end as a single network. The architecture of translation models is based on the encoder-decoder family of networks [Cho & van Merrienboer$^+$ 14a, Sutskever & Vinyals$^+$ 14] and later augmented with the attention layer, additive [Bahdanau & Cho$^+$ 15] or multiplicative [Luong & Pham$^+$ 15] variants. The main building block of the mentioned works is based on recurrent neural networks. However, convolutional encoder-decoder models [Gehring & Auli$^+$ 17b] or multihead self-attention [Vaswani & Shazeer$^+$ 17] are also used to directly transform source sentences into target sentences. As noted before, each of these architectures has its own advantages and disadvantages. The different representation offered by these architectures is a reason to combine them [Gehring & Auli$^+$ 17a, Chen & Firat$^+$ 18, Lin & Sun$^+$ 18, Werlen & Pappas$^+$ 18, Hao & Wang$^+$ 19].

The success of attention sequence-to-sequence modeling has led researchers to introduce extensions to it by equipping the attention layer with additional components or modifying the way it is computed. Motivated by the source length issue in translation and to mitigate over- and under-translation, it is proposed to maintain a coverage vector and keep track of previous attention weights with fertility [Tu & Lu$^+$ 16]. Alternatively, coverage models can be jointly trained with the whole translation model [Mi & Sankaran$^+$ 16]. The idea of using fertility to avoid over- and under-translation is further studied by [Malaviya & Ferreira$^+$ 18]. Similarly, attention-based models are extended by including positional bias, Markov conditioning, fertility, and agreement over translation directions [Cohn & Hoang$^+$ 16], and having a notion of what the attention layer has focused on in previous steps [Sankaran & Mi$^+$ 16]. An implicit distortion model is proposed as the attention mechanism treats the encoder states as a set, not a sequence, while the source orders are crucial for reordering [Feng & Liu$^+$ 16]. Our contribution in Section 3.5.3 lies on the same line of motivation with the verification of results from these papers.

There are some other works where the entire attention layer is replaced with recurrent layers such that left-to-right or bidirectional recurrent units serve as attention by encoding both encoder and decoder states jointly [Zhang & Xiong$^+$ 16, Zhang & Xiong$^+$ 17a, Zhang & Xiong$^+$ 17b]. In this architecture, the previous decoder state, which includes the target histories, serves to initialize the RNN states, and the encoder states serve as the inputs. Similar to this method, in Chapter 4, we describe a generalized extension of recurrent models to multiple dimensions. A dynamic vector based on recurrent units can be attached to the network to keep track of the history of attended positions [Yang & Hu$^+$ 17]. The intuition behind this is how many times a specific word has been attended to and whether the neighboring words are selected at the previous time steps.

Moreover, the reordering knowledge is incorporated into attention [Zhang & Wang$^+$ 17] and enables a word reordering penalty of source words. Similar to IBM models, it includes a probability distribution of relative jump distances between the newly translated source word and to-be-translated ones. The authors propose three different variations of distortion models. A fine-grained attention mechanism is also introduced in which besides each position on the source sequence, each component of the context vector has a separate attention score [Choi & Cho$^+$ 18].

Several works try to improve upon the standard attention model by adding an external memory structure [Wang & Lu$^+$ 16, Meng & Tu$^+$ 18], but none has been widely adopted. In the same line of research, interactive attention between the encoder and the decoder with read and write operations is proposed by [Meng & Lu$^+$ 16].

The advance of such attention models has marked one of the major milestones in the history of machine translation and inspired many researchers in other fields like speech recognition. The performance of early attention models in speech recognition is behind hybrid HMM models [Chorowski & Bahdanau$^+$ 15, Xu & Ba$^+$ 15, Chan & Jaitly$^+$ 16, Bahdanau & Chorowski$^+$ 16]. However, recent end-to-end models close this gap if a moderate amount of training data is provided [Chiu & Sainath$^+$ 18, Park & Chan$^+$ 19, Tüske & Saon$^+$ 20].

Furthermore, there is a group of works in which the main idea is to enable local and hard attention concepts. These types of attention inspire online streaming in speech recognition, simultaneous and document-level machine translation. Enabling a localized monotonic window for soft attention, this group of works entails the monotonicity feature of the speech-to-text alignment [Prabhavalkar & Sainath$^+$ 17, Hou & Zhang$^+$ 17, Chiu & Raffel 18]. In practice, such constraints on the attention distribution over input time steps lead to worse results than the global context [Tjandra & Sakti$^+$ 17, Merboldt & Zeyer$^+$ 19], though a restricted attention mechanism needs less memory. Hard attention [Aharoni & Goldberg 17, Raffel & Luong$^+$ 17, Lawson & Chiu$^+$ 18a], on the other hand, use stochastic sampling for a discrete decision and can be integrated with localized monotonicity [Chiu & Sainath$^+$ 18]. We address related works about such concepts in Chapter 5, where we focus on latent attention models. Besides hard attention models, other models like CTC [Graves & Fernández$^+$ 06], recurrent neural aligner [Sak & Shannon$^+$ 17], and RNN transducer models [Battenberg & Chen$^+$ 17, Prabhavalkar & Rao$^+$ 17] support online streaming in automatic speech recognition.

All neural models discussed in this section fall in the category of attention sequence-to-sequence models. In the next chapter, a model is studied that goes beyond attention models and departs from this overall structure. We will additionally review similar works in which the attention component can be ignored.

4. Two-Dimensional Sequence-to-Sequence Modeling

4.1 Introduction

In the attention models, the input and output sequences are treated as one-dimensional sequences over time, handled separately, and coupled by an attention mechanism. The attention mechanism is generally a linearly weighted sum of the input representations where the attention weights are the result of a similarity measure between the input and output representations. Since the encoder states are invariant to decoder states across output positions, the context vectors can be quite similar for different positions if the attention weights do not give enough relevant information. To obtain sufficiently discriminative contexts, models ideally need to align different target positions to different source positions. As discussed in Section 3.5.3, one way is to adjust future attention weights by including past alignment information as an indicator of what part of the source sentence has already been translated.

Despite well-designed attention refinements that are mainly focused on making attention weights more accurate, the encoder states are still computed only once at the beginning and are left untouched with respect to target histories. In other words, the attention models do not re-encode the source sequence while decoding. In this case, at every decoding step, the same vectors are read repeatedly, which might be partially attributed to the small differences in the context vectors. The context vectors can be insufficient in discriminatively predicting target tokens over time. This problem can be severe for longer sequences as the attention scores contribute less to each position.

This chapter contributes to the advancement of sequence-to-sequence modeling and applies two-dimensional (2D) long short-term memory (2DLSTM) to propose an alternative model for sequence-to-sequence modeling. Our main contribution is to go beyond the current sequence-to-sequence backbone models to a 2D structure in which the input and output sequences are aligned with each other in a 2D grid. The two-dimensional sequence-to-sequence approach is motivated by the fact that a 2DLSTM is able to process data with complex and highly nonlinear interdependencies in a 2D space [Graves 12b], and it therefore seems attractive and inspiring to apply this advantage in sequence-to-sequence modeling.

Instead of using any kind of attention component, a 2DLSTM layer alternatively defines the correspondence between source and target positions and assimilates the context from both sequences for a tighter coupling. The model thus accounts for each possible interaction. One dimension of 2DLSTM processes the source sentence, and another dimension predicts the target tokens. In contrast to the attention model that does not reinterpret the encoder states while decoding, this model enables the computation of the source sentence encoding as a function of the previously generated target tokens. It enhances the degree of variance in the context vectors by refining source representations to be sensitive to the partial translation generated by the decoder. We sidestep the inefficiency computational issue of the 2D structure by processing 2DLSTM states

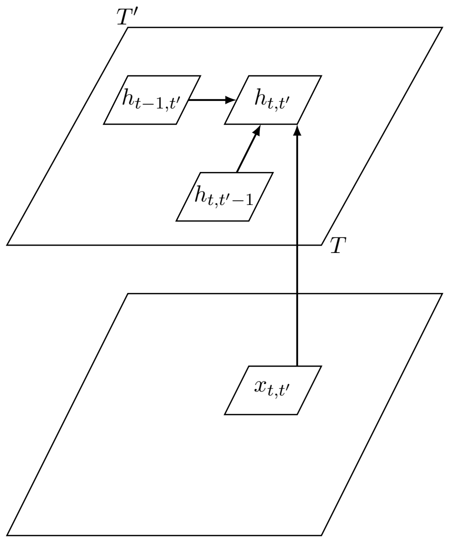


Figure 4.1: The two-dimensional LSTM layer unfolded over both axes for one time step. To compute the activation $h_{t,t'}$ at position (t, t'), the two predecessor hidden states $h_{t-1,t'}$ and $h_{t,t'-1}$ and the current input $x_{t,t'}$ have to be available.

over the diagonal in training. This optimization offers an efficient parallelized operation.

After describing the 2DLSTM layer, its differences from the standard LSTM unit, and corresponding details in Section 4.2, we explain our 2D recurrent sequence model in Section 4.3. In Section 4.3.4, an extension to the bidirectional two-dimensional model is addressed. As a proof of concept, Section 4.4 presents a single model using a 2D grid to serve for two-way translation modeling. Experimental results along with discussions are gathered in Section 4.5, followed by implementation hints in Section 4.6. The chapter is closed with a conclusion and related works in Sections 4.7 and 4.8.

4.2 Two-Dimensional Long Short-Term Memory

As discussed in Section 3.2.2, recurrent neural networks are well suited for sequence modeling where sequences are strongly correlated along a one-dimensional time axis. Handling dynamic length, encoding positional information, using previous contexts, and tracking long-term dependencies by gating strategies are some of the properties that make gated RNNs like LSTMs appropriate for such tasks. Although LSTM is essentially one-dimensional, it can be extended to process multidimensional data. Multidimensional LSTM has been introduced by [Graves 08, Graves & Schmidhuber 08, Leifert & Strauß$^+$ 16] as a generalization of the LSTM gating mechanism for multidimensional data. For instance, a multidimensional LSTM cell with two or three dimensions can be used to process 2D or 3D data like images and videos. In this work, we only consider 2DLSTM for sequence-to-sequence modeling.

A 2DLSTM cell can be seen as a mapping function $g : \mathbb{R}^{T\times T'\times d_x} \to \mathbb{R}^{T\times T'\times d_h}$ that transforms a three-dimensional input tensor into a three-dimensional output tensor. The first two dimensions represent spatial structures (T and T'), while the third corresponds to features. In this transformation, the first two dimensions of tensors keep the same dimension. For example, in machine translation, T corresponds to the source length J, while T' represents the target length I.

Figure 4.1 is an illustration of the dependencies in the 2DLSTM layer. At each position (t, t'), the computation of the hidden state $h_{t,t'}$ depends not only on the horizontal hidden state $h_{t-1,t'}$, but also on the vertical hidden state $h_{t,t'-1}$. $t \in \{1, \ldots, T\}$ is the index on the horizontal axis and $t' \in \{1, \dots, T'\}$ is the index on the vertical axis. Most importantly, they are not of the same length. Initial activations at time steps $t = 0$ and $t' = 0$, i.e., $h_{0,t'}$ and $h_{t,0}$ are usually set to 0.

In general, for an m-dimensional LSTM, there are m previous states, one from each dimension which are used to calculate hidden states of the current position. Like other neural networks, one can also stack 2DLSTM layers to form deep models with l layers. For simplicity of notation, we omit the superscript and consider $h_{t,t'} = h^{(l)}_{t,t'}$.

As for the standard LSTM unit, there are many variations for 2DLSTM. In the original topology proposed by [Graves & Schmidhuber 08], 2DLSTM uses an input gate $g_{i_{t,t'}}$ and an output gate $g_{o_{t,t'}}$ for position (t, t'). Based on the fact that there are two predecessor cell states $c_{t,t'-1}$ and $c_{t-1,t'}$, the 2DLSTM unit has two forget gates instead of one. Both predecessor cell states are multiplied component-wise with their corresponding forget gate, and then both values are summed. The remaining computations are analogous to the standard LSTM computation.

The magnitude of cell states can quickly increase over positions t and t' [Leifert & Strauß$^+$ 16]. The problem comes from the fact that for an m-dimensional LSTM, m predecessor hidden states with one forget gate each are utilized. The output of forget gates is between 0 and 1; thus, the sum of activations of m predecessor states, multiplied by the forget gates, can grow large over positions. This behavior results in unstable training.

Stable 2DLSTM To solve the above problem, a stable variant of 2DLSTM has been introduced [Leifert & Strauß$^+$ 16]. Its architecture and formulation are shown in Figure 4.2 and Equations 4.1-4.7. In this version, besides the input $g_{i_{t,t'}}$, forget $g_{f_{t,t'}}$ and output $g_{o_{t,t'}}$ gates, 2DLSTM employs an additional *lambda gate* $g_{\lambda_{t,t'}}$ at each position. As written in Equation 4.4, its activation is computed exactly like the other gates. The lambda gate is used to weigh the two predecessor cell states $c_{t-1,t'}$ and $c_{t,t'-1}$ and combine them into a single predecessor state by a convex combination before passing them through the forget gate (see Equation 4.6). This decides which information of the old cell state should be kept. The remaining computations are analogous to the standard LSTM. Similar to LSTM, the new cell state $c_{t,t'}$ is multiplied component-wise, denoted by $\odot$, with the output gate to yield the hidden state $h_{t,t'}$. The internal unit formulation is fully expressed by

$$g_{i_{t,t'}} = \sigma(W_i[x_{t,t'}; h_{t-1,t'}; h_{t,t'-1}] + b_i), \tag{4.1}$$

$$g_{f_{t,t'}} = \sigma(W_f[x_{t,t'}; h_{t-1,t'}; h_{t,t'-1}] + b_f), \tag{4.2}$$

$$g_{o_{t,t'}} = \sigma(W_o[x_{t,t'}; h_{t-1,t'}; h_{t,t'-1}] + b_o), \tag{4.3}$$

$$g_{\lambda_{t,t'}} = \sigma(W_\lambda[x_{t,t'}; h_{t-1,t'}; h_{t,t'-1}] + b_\lambda), \tag{4.4}$$

$$\tilde{c}_{t,t'} = \tanh(W_c[x_{t,t'}; h_{t-1,t'}; h_{t,t'-1}] + b_c), \tag{4.5}$$

$$c_{t,t'} = g_{f_{t,t'}} \circ \left[g_{\lambda_{t,t'}} \odot c_{t-1,t'} + (1 - g_{\lambda_{t,t'}}) \circ c_{t,t'-1}\right] + \tilde{c}_{t,t'} \odot g_{i_{t,t'}}, \tag{4.6}$$

$$h_{t,t'} = \tanh(c_{t,t'}) \odot g_{o_{t,t'}}. \tag{4.7}$$

Besides the input-to-recurrent weight matrices and bias terms, 2DLSTM uses two recurrent-to-recurrent weights, vertical and horizontal, such that $W_{\{i,f,o,c,\lambda\}} \in \mathbb{R}^{d_h \times [d_x + 2d_h]}$. $b_{\{i,f,o,c,\lambda\}} \in \mathbb{R}^{d_h}$ denote bias terms.

In general, for m dimensions, multidimensional LSTM makes use of m lambda gates; however, there is only one forget gate. In order to have the weights for a convex combination, the sum of lambda gates' activations is normalized by their sums. The number of lambda gates is reduced to $m - 1$ as the activation of the m-th lambda gate can be computed based on those of the others as the weights are summed to one [Leifert & Strauß$^+$ 16, Voigtlaender 16]. Therefore, 2DLSTM only requires one lambda gate.

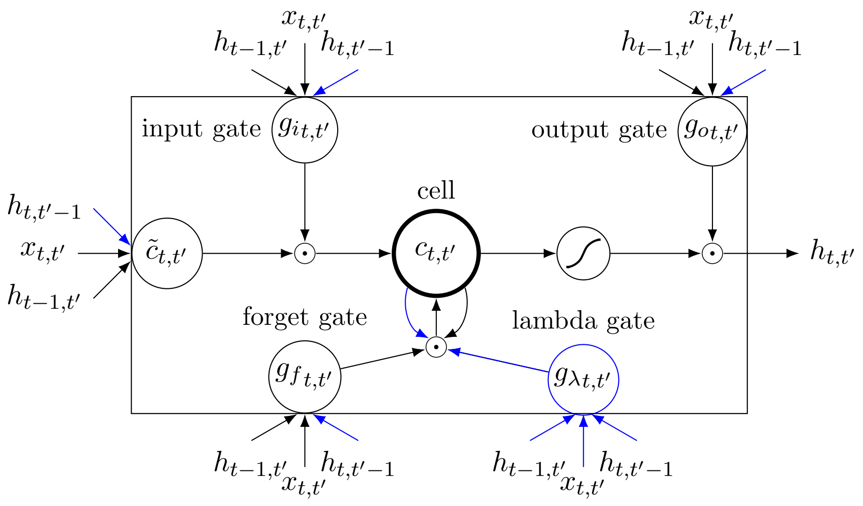


Figure 4.2: The diagram of the 2DLSTM cell for an input. The additional links versus the LSTM unit are marked in blue (cf. Figure 3.1).

Figure 4.2 also summarizes all necessary operations of the 2DLSTM component. The two predecessor hidden states are first combined into one by the lambda gate as a weight. This combined state is then used as in standard one-dimensional LSTM, but on two-dimensional data (compare with Figure 3.1). All the gates use the sigmoid activation function. The candidate cell node uses tanh. The blue arrows exhibit additional links in comparison to LSTM. The 2DLSTM layer is also shortened as a function when only representing inputs and an output given by

$$h_{t,t'} = \text{2DLSTM}\left(x_{t,t'}, h_{t,t'-1}, h_{t-1,t'}\right). \tag{4.8}$$

For simplicity, the trainable parameters are left out.

4.3 Two-Dimensional Model

We apply the 2DLSTM layers to map the input and output sequences into a 2D space. Here, we describe the model in the context of machine translation; however, it can be used in end-to-end automatic speech recognition and any other tasks with two parallel sequences.

We combine the source representation (or temporal frame-wise acoustic information in speech recognition) and a language model on the target side without any attention component. The 2DLSTM reconciles the context from both the source and target sentences and reinterprets the encoder states when predicting a new target token. We call it the *two-dimensional sequence-to-sequence* model, denoted as (*2D RNN*). We note that we use the general naming for recurrent models using the RNN term; however, the model benefits from LSTM units. The model can still be viewed as an encoder-decoder architecture. In contrast to the attention sequence-to-sequence model where the encoder states are not updated, and the model is not able to reinterpret the encoder states while decoding, this model enables the computation of the encoding of the source sequence as a function of the previously generated target tokens. This enhances the degree of variance in the context vectors by refining source representations to be sensitive to the partial translation generated by the decoder. The re-encoded source states contain both the original source and the previously generated target representations.

In the 2D RNN model, as shown in Figure 4.3, one dimension of 2DLSTM (the horizontal axis in the figure) serves as the refined encoder, and the vertical axis plays the role of the decoder. Before the 2DLSTM layer, the bidirectional LSTM-based encoder scans the source words and

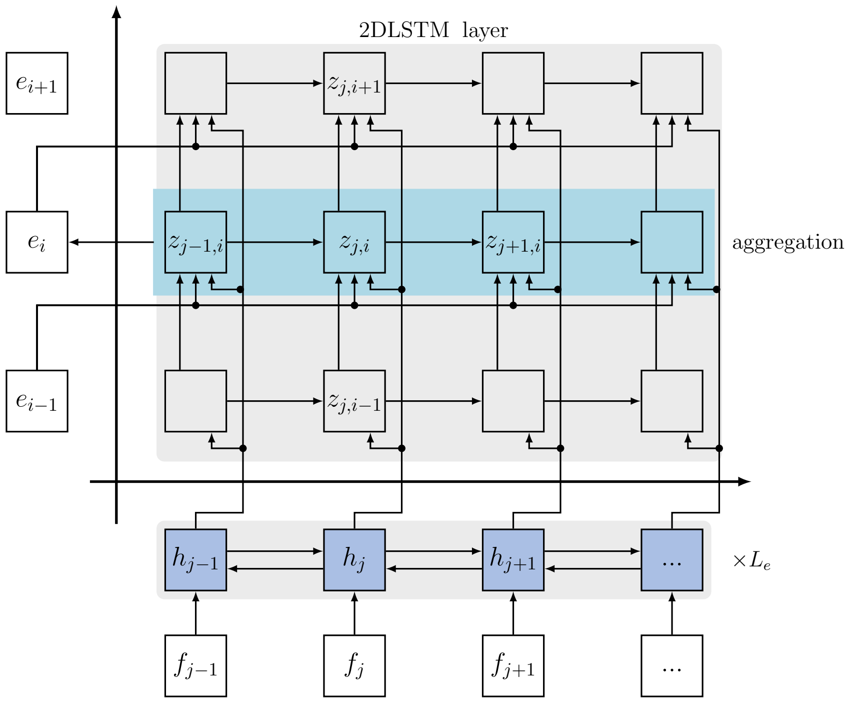


Figure 4.3: The 2D RNN model uses the forward 2DLSTM layer on top of the bidirectional encoder with L_e layers in machine translation. The same architecture is also applied to speech recognition.

obtains h_1^J in order to have the whole source context. At position (j, i), 2DLSTM receives both the encoder state h_j and the last target token embedding $\tilde{e}_{i-1}$ concatenated. The state of the 2DLSTM cell is given by

$$z_{j,i} = \text{2DLSTM}\left(\left[h_j; \tilde{e}_{i-1}\right], z_{j-1,i}, z_{j,i-1}\right) \tag{4.9}$$

where $z_{j,i}$ is the 2DLSTM state at position (j, i). Once the whole source sentence has been processed from 1 to J, the output is collapsed over source positions at each decoder step. This operation can be related to the linearly weighted sum of source positions in the attention models. Multiple operations are proposed in the following section. Once the output of 2DLSTM is condensed, a transformation followed by softmax is used to generate the next target token. Formally, we have

$$p(e_i|e_0^{i-1}, f_1^J) = \text{softmax} \circ \text{linear} \circ \text{maxout} \circ \text{aggregation}(z_{1,i}, \dots, z_{J,i}) \tag{4.10}$$

where the output of the aggregation operation is a compressed vector at the target position i, as described next.

4.3.1 Collapsing the Output for Sequence Prediction

The 2DLSTM layer computes an output with two spatial dimensions along with the feature dimension[1]. The horizontal axis represents the source states, the vertical axis processes the target words, and the 2DLSTM layer shapes a 3D tensor $z_{j,i} \in \mathbb{R}^{J \times I \times d_h}$. At each target position, the two-dimensional sequence needs to be reduced to a one-dimensional vector that can then be processed by higher layers or that computes a distribution over the target vocabulary at the end. Once the given encoded source sequence has been processed, one can compute all $z_{1,1}, \dots, z_{j,i}, \dots, z_{J,I}$

[1]For simplicity, the batch dimension is ignored, as it has to be preserved anyway.

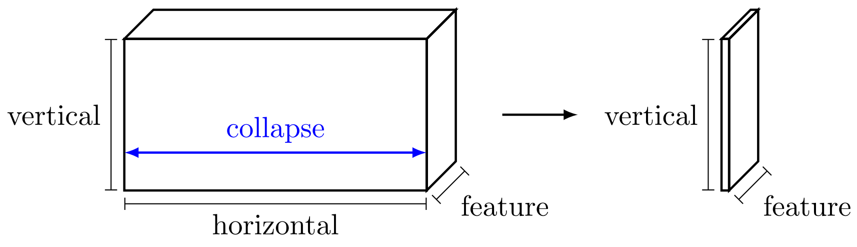


Figure 4.4: Collapsing the output in the 2D RNN model. The two-dimensional sequence of vectors (the vertical or horizontal axis) needs to be reduced to one-dimensional for the next layers. To this end, collapsing over the horizontal dimension is undertaken. In the proposed model, the vertical axis corresponds to the output sequence.

states. To predict a single word, we need to aggregate J d_h-vectors into a fixed-size representation at each output position i. The aggregation function should select the most relevant encoder states of length J. We highlight that the collapsed output can be seen as the context vector as before and denoted with $c_i \in \mathbb{R}^{d_h}$ by which each target token e_i is predicted. Generally speaking, this operation can be expressed as

$$c_i = \text{aggregation}(z_{1,i}^{J,i}). \tag{4.11}$$

This is done by collapsing the output of the 2DLSTM layer horizontally, as shown in Figure 4.4. The collapsing of 2DLSTM output can be conducted over both axes depending on the intended purpose. We will apply aggregation functions over both dimensions in Section 4.4, where the goal is to predict source-to-target and target-to-source translations jointly.

Max-Pooling

A straightforward approach to reduce the length of the source sentence is max-pooling. J vectors are max-pooled, meaning that for every component of a vector $k \in [1, \ldots, d_h]$, we select the component with the maximum value. Hence, we have

$$c_i = \max_{1 \leq j \leq J} z_{j,i,k} \qquad \forall k \in [1, \ldots, h_d], \forall j \in [1, \ldots, J]. \tag{4.12}$$

In fact, for a given target position, the max-pooling operator assigns d_h features to different source positions.

Average-Pooling

Instead of selecting the maximum component, one can take the average of J vectors for each component of the vector. We scale the sum with the inverse of the source length to stabilize the variance of the pooled features and derive

$$c_i = \frac{1}{J} \sum_{1 \leq j \leq J} z_{j,i,k} \qquad \forall k \in [1, \ldots, h_d], \forall j \in [1, \ldots, J]. \tag{4.13}$$

Last Hidden State

We also choose $z_{J,i}$ as the final output of the 2DLSTM layer at each target position i, as it should theoretically contain the entire source sentence information from time step 1 to J and also all past target histories until the position i. Therefore, the output is collapsed to a single vector by

$$c_i = z_{J,i}. \tag{4.14}$$

Weighted Sum

For many applications, it is necessary to have normalized weights, a distribution that can be considered as a notion of alignment between source and target sentences. All the operations described so far yield no alignment scores, though max-pooling leverages the pooled positions to obtain implicit alignments. Max-pooling assigns d_h channels to various input positions. With average-pooling, all positions contribute equally to the final feature. With the last hidden state, information from all positions is implicitly included based on the internal gating mechanism of 2DLSTM, emphasizing the later steps.

As an alternative, we also add an additional component to compute a weighted sum over the horizontal axis and use the obtained vector as the compressed output. Similar to the attention mechanism [Bahdanau & Cho+ 15, Luong & Pham+ 15], we get a distribution over the source positions and provide scores as soft alignments between the source and target positions. These attention weights are used to weigh the J vectors differently. Let us define $\alpha(j|i)$ as the normalized attention weights that focuses on f_j for a given e_i. An implicit alignment between two sentences is expressed by the general multiplicative function of target word embedding from the last position $\tilde{e}_{i-1}$ and the 2DLSTM hidden states. Using softmax, we write

$$\alpha(j|i) = \text{softmax}(\tilde{e}_{i-1}^{\top} W_{att} z_{j,i}) \tag{4.15}$$

where $W_{att} \in \mathbb{R}^{d_e \times d_h}$ is a learnable parameter. Then, the weighted sum of the 2DLSTM hidden states is used as the collapsed output such that

$$c_i = \sum_{j=1}^{J} \alpha(j|i) z_{j,i}. \tag{4.16}$$

4.3.2 Training

To train the 2DLSTM cell, back-propagation through time is performed over two dimensions [Graves 08, Graves 12b]. The order of forward and backward passes is opposite to each other. If the two-dimensional grid is processed from the bottom left to the top right corner in the forward pass, then the gradient is passed backward from time step (J, I) to the origin $(1, 1)$. The derivatives with respect to the trainable parameters are derived by summing over all positions. More details and the derivations of the gradients can be found in [Graves 08].

4.3.3 Order of Processing

One practical concern is the difference between training and decoding. During training, all states of 2DLSTM are computed once at the beginning because the entire source and target sentences are available.

As illustrated in Figure 4.5, the processing is typically done column by column or row by row. Importantly, the 2DLSTM state at position (j, i) can only be processed once two predecessors at positions $(j-1, i)$ and $(j, i-1)$ have been computed. This constraint is fulfilled when processing the 2D grid diagonal-wise. This processing scheme enables us to merge the kernel invocations for all positions on one diagonal, reducing the necessary computation time. As shown in Figure 4.5(a), all cells on each diagonal are computed in parallel. The blue arrows indicate a sequential order of processing. For a 2D grid with J and I words on the source and target sentences, respectively, there are $I \times J$ cells to process. In contrast, the number of diagonals is $I + J - 1$, which in most cases should be significantly lower than the number of cells, leading to faster training. While GPU capacity limits this parallelization, in practice, the training computation can be optimally

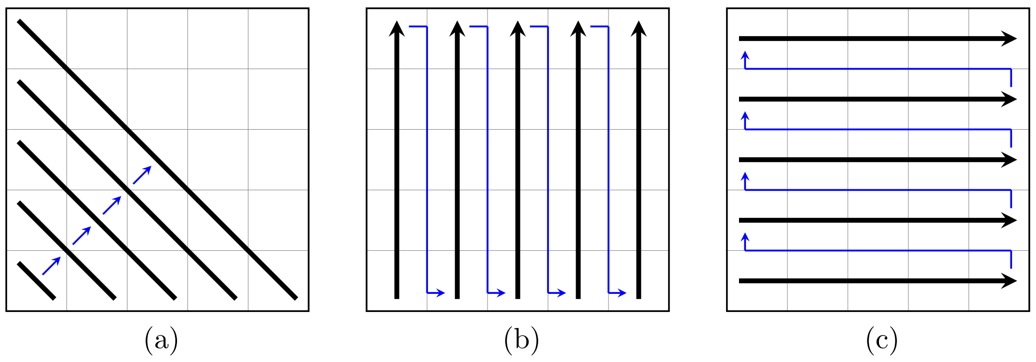


Figure 4.5: Order of processing in 2DLSTM. a) diagonal-wise used in training, where all cells on the same diagonal are computed in parallel. b) column-wise used in target-to-source decoding, where columns are not computed at the same time. c) row-wise used in source-to-target decoding where rows are not computed at the same time. The thick black and thin blue arrows show the direction of the sequential process between cells and columns/rows, respectively.

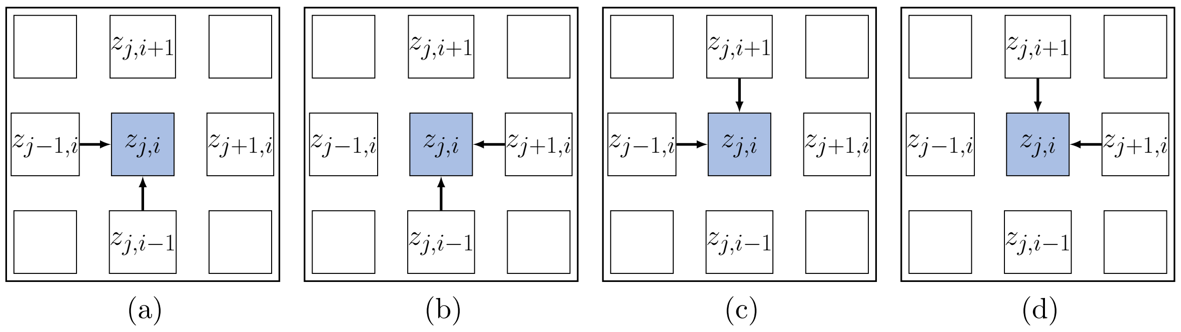


Figure 4.6: Four directions in which the 2DLSTM layer can process the input. For both the vertical and horizontal axes, there are two possible directions, offering a total of four combinations. The arrows indicate which predecessors are taken to compute the activation for position (j, i) colored in blue. a) left-to-right and bottom-to-top, b) right-to-left and bottom-to-top c) left-to-right and top-to-bottom, and d) right-to-left and top-to-bottom.

parallelized to linear time [Voigtlaender & Doetsch$^+$ 16]. In theory, the complexity of training is $\mathcal{O}(JI)$.

In contrast, the computation cannot be done as a single kernel invocation during decoding since the output sequence has to be predicted. This prevents the computation of multiple cells on the same diagonal at once. Thus, the 2DLSTM states have to be computed row-wise for source-to-target, and column-wise for target-to-source translation (see Figure 4.5). Here, for each row or column, the blue arrows imply the order of processing between columns or rows and sequence-style dependency on the one before that.

As in training, the entire source and target sequences are available, this allows us to compute all states of the 2DLSTM before performing any aggregation and prediction step. We process the 2D grid in the forward pass from the bottom left (position $(1, 1)$) to the top right corner (position (J, I)). Analogously, the gradient is passed backward in the opposite direction by parallel computations for all cells on each diagonal.

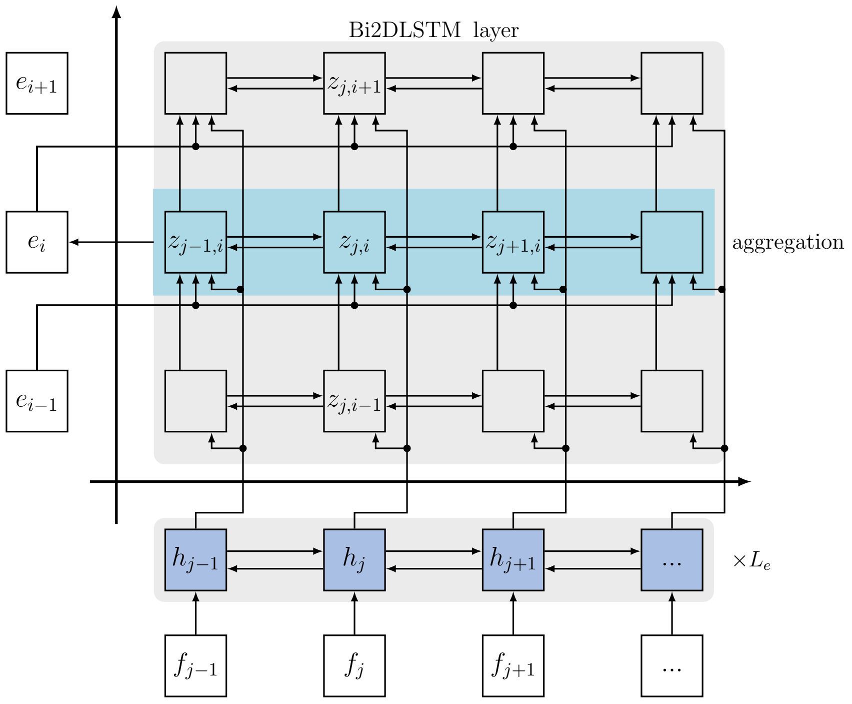


Figure 4.7: The bidirectional 2D RNN model using the concatenation of the forward and backward 2DLSTM layers at position (j, i). The bidirectional arrows represent the bidirectional operations.

4.3.4 Bidirectional Two-Dimensional Model

Similar to the bidirectional LSTMs for one-dimensional sequences, in principle, there are four possible directions in which 2DLSTM can process the input as shown in Figure 4.6. For both the vertical and horizontal axes, there are two possible directions, offering a total of four combinations, left-to-right and bottom-to-top, right-to-left and bottom-to-top, left-to-right and top-to-bottom, and right-to-left and top-to-bottom. Multidirectional 2DLSTM employs four parallel layers that each scans the input in one of the four possible directions. To derive the entire context information of the 2D input, the activations of all four directions are then merged per position by either summation, averaging, or concatenation [Graves 08].

For sequence-to-sequence modeling like machine translation, since the successor tokens in the target sentence are not known and have to be generated, the processing directions are restricted to left-to-right and bottom-to-top (*forward*) and right-to-left and bottom-to-top (*backward*) directions, i.e., Figure 4.6(a) and 4.6(b).

Introducing an additional 2DLSTM hidden layer with the same connections as in the original one but moving backward on the horizontal, yet forward in the vertical axis, we extend it to the *bidirectional 2DLSTM (Bi2DLSTM)* layer. $\overrightarrow{z_{j,i}}$ and $\overleftarrow{z_{j,i}}$ are used to denote the forward and backward hidden states, respectively. As a combination of two 2DLSTMs, the Bi2DLSTM has access to two states at any given position (j, i), i.e., $\overrightarrow{z_{j,i}}$ and $\overleftarrow{z_{j,i}}$, summarizing the past and future input information. In our implementation, the final representation $z_{j,i}$ is simply the concatenation of states from the forward and backward directions at each position (j, i) written as

$$z_{j,i} = [\overrightarrow{z_{j,i}}; \overleftarrow{z_{j,i}}]. \tag{4.17}$$

In this case, the model dependency of Equation 4.9 is modified to

$$z_{j,i} = \text{Bi2DLSTM}\Big(\big[h_j; \tilde{e}_{i-1}\big], z_{j-1,i}, z_{j+1,i}, z_{j,i-1}\Big). \tag{4.18}$$

The bidirectional model can thus create an output that focuses on the inputs right around position j but also depends on input values that are far in the future or past. Once a sequence of $z_{j,i}$ vectors is obtained, Equation 4.10 is applied to generate target tokens. Throughout this work, we name this model the *bidirectional 2D RNN* model, and it is illustrated in Figure 4.7.

4.4 Two-Way Translation Model

The attention models can be seen as unidirectional translation architectures as they translate only from one language to another. Although such models have proven to be effective for high-quality unidirectional translation given a moderate amount of training data, their capacity for two-way translation modeling, i.e., both source-to-target and target-to-source translation using a single model is limited. It is not easy to build a single system that translates reliably from and to two or even more languages. In practice, usually, one model is trained for each direction, and each might only capture partial aspects of the mapping between words [Cheng & Shen$^+$ 16]. The two models seem to be complementary. Therefore, combining them may hopefully improve translation quality in both directions.

An early solution to use a single model for translation between multiple languages is a multilingual system. A one-to-many multilingual system translates from one source language into multiple target languages by adding a separate attention mechanism and decoder for each target language [Dong & Wu$^+$ 15]. A single shared attention mechanism but multiple encoders and decoders for each source and target language can be also used [Firat & Cho$^+$ 16]. Multi-task learning is also proposed for many-to-many tasks [Luong & Le$^+$ 16], where the architecture is extended to have multiple encoders and decoders. A multilingual system can only have a single attention encoder-decoder model [Ha & Niehues$^+$ 16, Johnson & Schuster$^+$ 17]. No change in the network is needed in such systems, but they work with multilingual data to have both source-to-target and target-to-source translations. In practice, these models add an artificial token to the source sentence to mark the target language and to identify the translation direction correctly. Excluding the multilingual systems, one can define a new training loss that combines likelihoods in two directions with a word alignment agreement [Cheng & Shen$^+$ 16].

Inspired by the 2D RNN model, we construct a single *two-way translation* model as a by-product of the 2D grid. It opens up the interaction between source-to-target and target-to-source translation directions jointly trained on the same training data. Our model is similar to the 2D RNN model with small modifications, but it produces translations in both directions. The presented idea is a proof of concept that offers a different and interesting model with initial results, but it has not yet reached parity on translation tasks.

4.4.1 Model Architecture

As shown in Figure 4.8, we apply an encoder with L layers to scan the source sentence from left to right. The encoder layers can be composed of either the unidirectional LSTM or self-attention components. In the latter case, to avoid the future context, we mask out the coming words and only attend to the histories up to time step j. Similar to the source encoder, we use a stack of L unidirectional (or masked) layers on the target side. It yields h_j and s_i as the source and target encoder states, respectively. The source and target sentences do not need to be of the same length. The use of source and target encoders is optional in this architecture; however, initial experiments have shown that they help the performance.

We further equip the network with the 2DLSTM layer to relate the source and target encoder states without any attention component in between. At position (j, i), 2DLSTM receives the concatenation of the last source encoder state h_{j-1} and the previous target encoder state s_{i-1}.

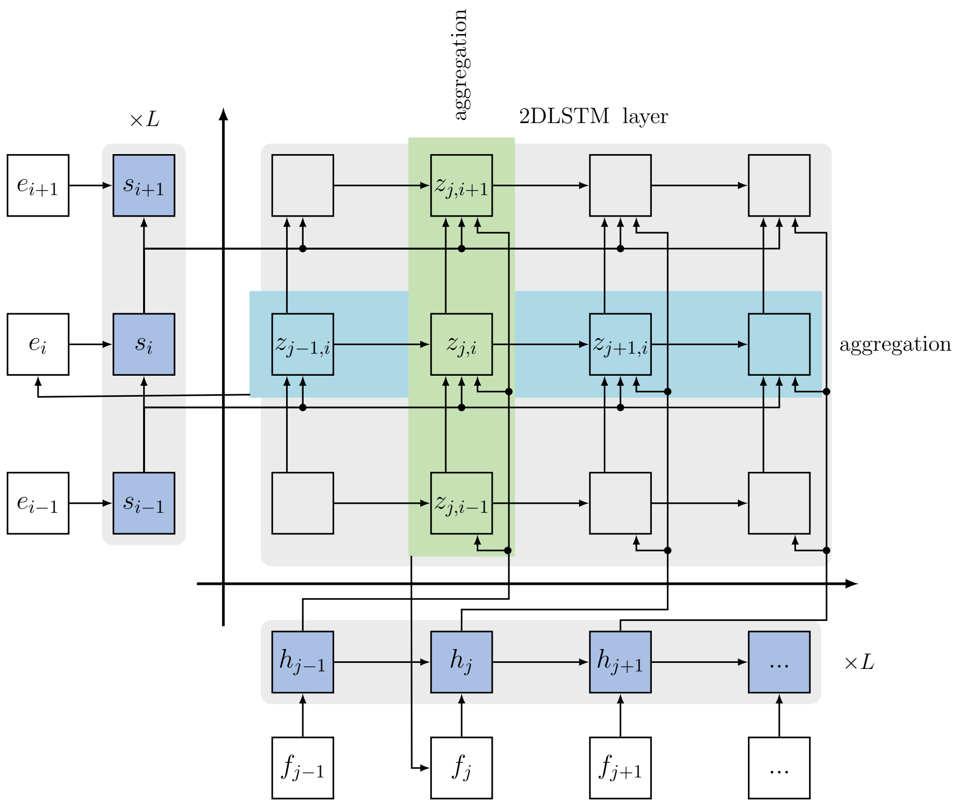


Figure 4.8: The two-way translation model using the 2DLSTM layer on top of the L-layer of the source and target encoders. The light blue and green blocks indicate aggregation over the horizontal and vertical axes.

As before, its recurrence activation relies on both the vertical $z_{j,i-1}$ and horizontal $z_{j-1,i}$ hidden states. The state of 2DLSTM is updated by

$$z_{j,i} = \text{2DLSTM}\Big(\big[h_{j-1}; s_{i-1}\big], z_{j-1,i}, z_{j,i-1}\Big). \tag{4.19}$$

The 2DLSTM state for the token at target position i depends only on the previous tokens e_1^{i-1}. Similarly, the source state at position j only relies on the preceding source tokens f_1^{j-1}. At each decoder step, once the whole source sentence has been processed from 1 to J, the sequence of vectors is aggregated by max-pooling over all horizontal states to obtain the context vector as depicted by light blue in Figure 4.8. In order to generate the next target token e_i, a transformation followed by softmax is applied. The same procedure is employed at the same time for the source token f_j, as shown in light green. Max-pooling aggregates all vertical states to a single context vector to predict the next source token. Therefore, we derive

$$p(e_i|e_0^{i-1}, f_1^J) = \text{softmax} \circ \text{linear} \circ \text{aggregation}(z_{1,i}^{J,i})\big|_{V_e}, \tag{4.20}$$

$$p(f_j|f_0^{j-1}, e_1^I) = \text{softmax} \circ \text{linear} \circ \text{aggregation}(z_{j,1}^{j,I})\big|_{V_f} \tag{4.21}$$

where V_f and V_e are the source and target vocabularies as before. No attention component is used in this model, and inherently we have no future context information, i.e., no bidirectional layers. The reason for this constraint is that including it on one language side would break the network's ability to translate in that direction. All the loss functions are differentiable with respect to the model parameters. It is easy to extend the original training algorithm to implement joint training since two translation models in two directions share the same training data and a set of parameters. The training loss is therefore defined as

$$\mathcal{L} = \log p(e_1^I | f_1^J) + \log p(f_1^J | e_1^I). \tag{4.22}$$

Once we have computed all $I \times J$ states, it enables translation. Afterwards, vertical and horizontal slices can be taken in parallel to compute the individual losses, which are then used for back-propagation. One row of it is taken at each target position, while at each source position, one column is used.

As explained in Section 4.3.3, the 2DLSTM states are processed diagonal-wise in training. This parallelization is not possible during decoding, thus we compute the states of the 2DLSTM grid column-wise for target-to-source and row-wise for source-to-target translation, as shown in Figure 4.5(b) and 4.5(c), respectively.

4.5 Experimental Results

4.5.1 Machine Translation

Setups

All machine translation experimental setups follow those explained in Section 3.8. Like the RNN attention model, 6 bidirectional LSTM layers are used on top of each other; each has 1024 hidden nodes. The decoder is composed of only one forward 2DLSTM layer of size 1024. We also use the concatenation of backward 2DLSTM in the model to have a bidirectional property over encoder states. Again similar to the baseline models, a linear projection and maxout, followed by the softmax function are applied on the output layer. In order to aggregate the sequence of vectors, we try all variations explained in Section 4.3.1.

Most training hyperparameters are identical to the RNN attention model. However, we use a smaller batch size, and the gradients are accumulated to compensate for that. The model is trained with a lower initial learning rate of 0.0005 for better convergence at the end. In the initial experiments, we restrict the maximum sequence length of source sentences to 50 tokens as it speeds up the training phase. Whenever we compare with other baselines, we keep all sequences with a length of up to 100 subwords and ensure that all systems receive the same training data.

A dropout rate of 0.3 is used between recurrent weights of the 2DLSTM layer, unless explicitly stated otherwise.

Results

Aggregation Functions We first examine the mentioned aggregation mechanisms to reduce the 2DLSTM output to a fixed-size representation that is independent of the sequence length. Four 2D RNN models are trained with the same encoder (6 layers of BiLSTM) and one forward 2DLSTM in the decoder. Each model collapses the source length differently. All models have the same number of parameters as the aggregation functions are weightless except for the weighted sum operator with an extra but a negligible number of parameters. Tables 4.1 and 4.2 show the results on the WMT German→English and Chinese→English tasks.

In average-pooling, all source positions are treated equally, and by picking the last state, the final positions contribute to the model the most. Max-pooling and weighted sum have a clear advantage over average-pooling and the last state in terms of translation performance. In either case, the 2D RNN model leverages the pooled positions to obtain an implicit alignment assigning to different positions. These aggregation functions have almost a similar performance on average across the two tasks (compare lines 1 and 4 in the tables).

Table 4.1: The uni- and bidirectional 2D RNN results using different aggregations on WMT 2018 German→English. The maximum sequence length is set to 50 tokens in training as it accelerates the initial experiments. The number of parameters for all aggregation methods is around 220M and 245M if the backward 2DLSTM layer is added.

model			**dev**		**newstest2017**		**newstest2018**	
forward	**backward**	**aggregation**	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]
2DLSTM	no	max-pool	31.2	56.6	32.0	56.6	39.0	48.1
		average-pool	30.8	57.0	31.7	56.7	38.5	48.3
		last state	30.4	57.7	31.1	57.5	37.9	49.2
		weighted sum	31.4	56.4	32.4	56.3	38.8	47.8
	2DLSTM	max-pool	31.2	56.7	31.8	56.8	38.8	48.1
		weighted sum	31.4	56.6	31.9	56.7	39.0	48.3

Table 4.2: The 2D RNN results using different aggregations on WMT 2018 Chinese→English. The maximum sequence length is set to 100 tokens. The number of parameters for all aggregation methods is around 220M.

model			**dev**		**newstest2017**		**newstest2018**	
forward	**backward**	**aggregation**	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]
2DLSTM	no	max-pool	21.1	63.7	23.1	62.6	22.8	66.8
		average-pool	20.8	64.7	22.6	62.9	22.0	67.8
		last state	20.5	64.7	22.3	62.8	22.1	67.7
		weighted sum	21.0	63.6	22.8	62.3	22.4	66.7

Additional Backward Layer We also explore using one backward 2DLSTM layer of size 1024 whose features are concatenated to the forward layer described in Section 4.3.4. That yields a total of 245M trainable parameters for the bidirectional 2D RNN model. The second part of Table 4.1 lists the results for the better aggregation mechanisms from our first set of experiments. Adding the backward layer (lines 5 and 6) gives no gain even with extra parameters, and there is no real benefit in using it. Since the 2D RNN model already contains a strong bidirectional encoder, this can justify the observation that adding an additional right-to-left or backward 2DLSTM layer on top of that is not important. This is in line with the initial experiments in which we replace the bidirectional encoder with the unidirectional one while keeping the backward 2DLSTM layer. This architecture shows worse translation performance. Moreover, adding bidirectional LSTMs in the encoder is cheaper than having several backward 2DLSTM layers in the decoder.

Combination with Self-Attention Encoder The 2D RNN model is built with a deep bidirectional LSTM encoder that takes a lot of trainable parameters. However, it can also be combined with self-attention layers where the number of parameters is smaller and provides powerful modeling on its own, independent of the recurrent-based encoder. To affirm the hypothesis of whether the expressiveness of the 2D RNN model depends on its encoder design choice, we train it with two different encoders: 6 layers of self-attention components and 6 layers of bidirectional LSTM. The former reduces the number of parameters by almost half. The translation results are shown in Table 4.3. We observe that the new combination performs similarly to the recurrent encoder on average. This confirms the ability of 2DLSTM as a stand-alone decoder working without any dependency on the encoder architecture.

Table 4.3: The 2D RNN results using different encoders on WMT 2018 German→English.

model		dev		newstest2017		newstest2018		#
decoder	**encoder**	Bleu$^{[\%]}$	Ter$^{[\%]}$	Bleu$^{[\%]}$	Ter$^{[\%]}$	Bleu$^{[\%]}$	Ter$^{[\%]}$	**parameters**
2DLSTM	LSTM	31.6	56.3	32.2	56.7	39.4	48.0	220M
	self-attention	31.4	56.3	32.5	56.5	38.9	48.3	97M

Table 4.4: Performance comparison between the 2D RNN model and the baselines on WMT 2018 German→English.

model	dev		newstest2017		newstest2018		#
	Bleu$^{[\%]}$	Ter$^{[\%]}$	Bleu$^{[\%]}$	Ter$^{[\%]}$	Bleu$^{[\%]}$	Ter$^{[\%]}$	**parameters**
RNN attention	31.9	55.3	32.9	55.2	39.7	46.8	217M
self-attentive	32.4	55.8	33.2	55.3	40.4	46.8	102M
+ $d = 1024$	33.3	54.7	34.2	54.5	40.6	46.8	291M
2D RNN	31.6	56.3	32.2	56.7	39.4	48.0	220M

Table 4.5: Performance comparison between the 2D RNN model and the baselines on WMT 2018 Chinese→English.

model	dev		newstest2017		newstest2018		#
	Bleu$^{[\%]}$	Ter$^{[\%]}$	Bleu$^{[\%]}$	Ter$^{[\%]}$	Bleu$^{[\%]}$	Ter$^{[\%]}$	**parameters**
RNN attention	21.2	63.1	23.2	61.1	22.7	65.9	217M
self-attentive	22.0	62.8	23.1	61.7	23.0	66.1	101M
+ $d = 1024$	22.7	61.7	23.5	61.0	23.8	65.3	290M
2D RNN	21.1	63.7	23.1	62.6	22.8	66.8	220M

Given these results, the 2D RNN model equipped with one forward 2DLSTM, aggregated by the weightless max-pooling function and composed of 6-layer bidirectional LSTM in the encoder is employed in the rest of the machine translation experiments.

Translation Performance We compare the 2D RNN model to two baselines: the RNN attention and self-attentive models. To ensure a fair assessment, we fix a few design choices, such as the maximum sequence length (100 tokens) and the maximum number of training epochs. Tables 4.4 to 4.6 present results on three WMT tasks. On German→English and Chinese→English, the 2D RNN model achieves comparable results over the RNN attention baseline in terms of Bleu, while the self-attentive model is superior. 2D RNN is behind the baselines on all sets with respect to Ter. Surprisingly, on the English→German `newstest2014` benchmark test set, the model outperforms both the RNN attention and self-attentive baselines by 0.8 and 0.1 Bleu points, 0.5% and 0.2% in Ter, respectively. Although the 2D RNN model has a stable convergence behavior, its 2D structure and complex dependencies can partially explain the slightly lower performance. Its training involves a more complicated optimization problem that makes the entire procedure harder to be trained.

Sequence Length Analysis The 2D RNN model produces translation-sensitive source representations since it refines the encoder states according to what has already been generated. To verify this, we focus on a well-known challenge in attention-based models, and that is how the model

Table 4.6: Performance comparison between the 2D RNN model and the baselines on WMT 2014 English→German.

model	newstest2013		newstest2014		#
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	parameters
RNN attention	25.8	56.8	26.9	56.3	224M
self-attentive	26.3	57.0	27.6	56.0	108M
2D RNN	26.1	56.7	27.7	55.8	227M

performance changes with respect to the source sentence length. To do so, we group sentences of similar lengths and compute BLEU, TER, and the average length of hypotheses for each group.

Following [Rosendahl & Tran+ 19], all test sets `newtest2008-2018` on WMT German→English are concatenated, resulting in a larger test set of 30244 sentence pairs. Consecutive sentence pairs of this set are concatenated to achieve longer sentences with reasonable quality. This concatenation provides a set of 15k sentence pairs with double the average length. Then, we divide the set into 5 bins according to the source sentence length such that each group has a source length difference of 25 subwords. We denote such a bin by $G_{a:b}$ where $a \leq J \leq b$. As a simple example, $G_{0:25}$ contains the group of sentences in which the source length is limited to 0 to 25 tokens. In our setup, the smallest group contains 755 sentences, allowing a more reliable interpretation compared to the groups with few sentences. We note that this setting contrasts with the training mode where the models learn to translate one source sentence without further context beyond the sentence boundary. But the relative performance difference between the models is still reliable as all are equally exposed to this problem.

The length of the observed source sentences is limited to be no longer than 50, 75, and 100 subwords during training. Figure 4.9 demonstrates performance differences between the models for a maximum sequence length of 50 and 100. Table B.1 in Appendix also shows the details of the translation performance of each group in BLEU and TER as well as the size of each bin.

The performance of all models is consistent for sequence lengths observed during training. For instance, when the maximum sequence length is set to 100 tokens, the models' performance is very similar for different lengths up to 100 tokens ($G_{0:25} \rightarrow G_{51:75}$) and drops dramatically in terms of BLEU and TER when source sentences with more than 100 tokens are considered (see the left panel in Figure 4.9). We see the same behavior for models trained on 50 subwords maximum (the right side of the figure). This drop is more severe for the self-attentive model than the RNN-based architecture. Compared to the self-attentive model, RNN attention is more robust in translating long sentences. Moreover, it confirms that the performance of the self-attentive model does not suffer on long sequences while it cannot generalize over unseen sequence lengths.

In comparison, the 2D RNN model is significantly superior if applied to unseen sequence lengths, with the highest performance on longer sentences when the source sentences extend beyond the maximum sequence length observed. In general, the model is capable of translating both short and long sequences, successfully generalizing over longer sequences, and outperforming all the baselines. One main reason for this is that the attention models generate much shorter hypotheses on longer sentences, as shown in Figure 4.9(f). While the 2D RNN model implicitly discourages an early stop of decoding and pushes for untranslated parts of the source sentence.

These results also indicate that the performance drop of the self-attentive model cannot be related to losing training data, as restricting the maximum sequence length in training leads to fewer sentence pairs. The reason is that the recurrent-based models do not suffer from this issue. Our results also confirm the findings of [Rosendahl & Tran+ 19]. In addition, they also show that the performance of the self-attentive model using absolute positional encoding is much worse than

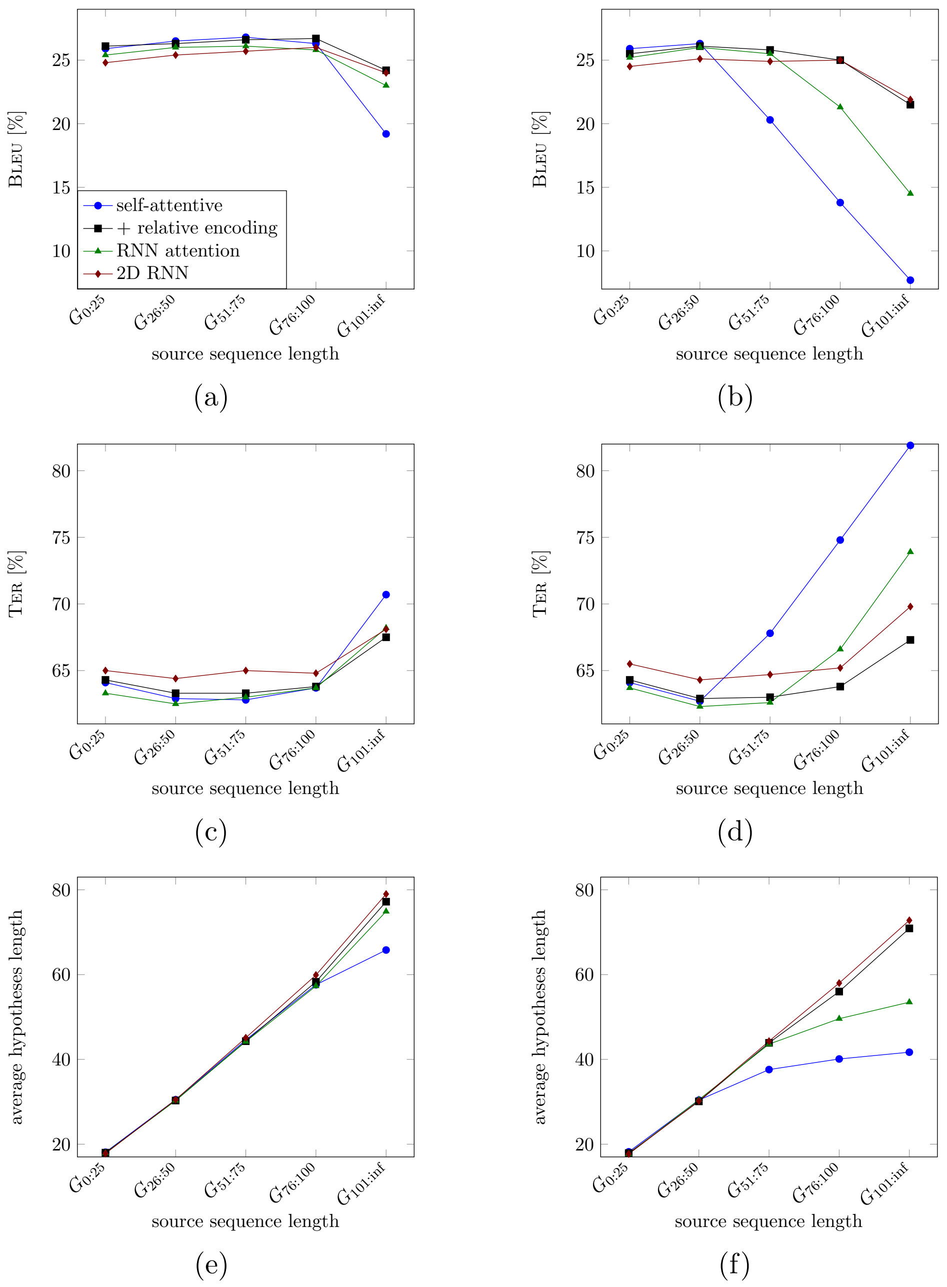


Figure 4.9: Performance of the models versus the source sequence length on WMT German→English. The source length is restricted in training to 100 tokens in a), c), and e), respectively, to 50 tokens in b), d), and f). The size of each group is $|G_{0:25}| = 1070$, $|G_{26:50}| = 6333$, $|G_{51:75}| = 5084$, $|G_{76:100}| = 1880$, and $|G_{101:\text{inf}}| = 755$.

that of relative encoding [Shaw & Uszkoreit+ 18]. A further experiment with relative positional encoding in the self-attentive model is carried out for a fair assessment, presented in black in the figure. The relative positional encoding significantly compensates for the deficiency of the self-attentive model, and its performance does not drop when the length extends beyond the maximum sequence length in training. Although it is superior to the 2D RNN model with respect to BLEU and TER, it slightly generates shorter translations on longer sentences, as seen in Figures 4.9(e) and 4.9(f).

Table 4.7: Performance comparison of the 2D RNN with the 2D CNN model on the IWSLT and WMT German→English translation tasks. Evaluation is done by `multi-bleu.pl`.

method	BLEU [%]			
	IWSLT		**WMT**	
	dev	**test**	**newstest2013**	**newstest2015**
2D CNN [Elbayad & Besacier+ 18]	36.0	34.7	29.0	29.6
RNN attention [Elbayad & Besacier+ 18]	31.7	31.0	-	-
self-attentive [Elbayad & Besacier+ 18]	35.6	34.5	31.6	32.8
2D RNN	36.1	34.6	30.2	31.2

Comparison with Other Works Concurrent with this work, in [Elbayad & Besacier+ 18], convolutional neural networks are used to also motivate a similar two-dimensional layout to jointly encode the source and target sentence. Their model forms pairwise interactions between the source and target tokens. Once the source and target embedding are concatenated as input representation, a set of stacked 2D convolution networks is applied with different filter sizes. The model uses residual-cumulative skip connections between consecutive layers interleaved with feedforward layers. To predict the target token, the final convolution features are mapped to a single fixed-size vector by self-attention. This model is referred to as the *2D CNN* model in our comparison.

To compare the model performance to the 2D CNN model, we replicate the setup in [Edunov & Ott+ 18, Elbayad & Besacier+ 18] for the IWSLT 2014 German→English task. The dataset contains training, dev, and test sets with 160K, 7283, and 6750 sentence pairs, respectively. The bilingual texts on both the source and target sides are tokenized and lower-cased. BPE subword segmentation yields German and English vocabularies of 8.8K and 6.6K tokens.

We also conduct a comparison on the WMT 2015 German→English task where the `Rapid` part of the training data is excluded to use exactly the same data as in [Elbayad & Besacier+ 18]. In this task, `newstest2013` as dev and `newstest2015` as test sets are used with 3000 and 2169 sentence pairs, respectively. Joint BPE with 50K symbols is used. The 2D RNN model follows an identical training setup as for the WMT 2018 German→English task. The model has 174M and 220M free parameters for the IWSLT and WMT tasks. The 2D CNN model uses 14 layers of CNN with a model size of 256 for IWSLT, respectively 16 layers of CNN with a model size of 512 for the WMT task. These configurations lead to 14M and 49M total number of parameters. Peer-to-peer comparison pushes us for case-insensitive and case-sensitive tokenized evaluation for IWSLT and WMT, both using `multi-bleu.pl`.

Table 4.7 gives a comparison between the models. In either task, we achieve competitive results or clearly outperform the 2D CNN model. It is worth mentioning that the 2D CNN model has considerably fewer parameters than the 2D RNN model. The reason is that we have not tuned our model on every single task. Instead, we configure a network to be generalized enough such that it works equally well in all experiments.

We also compare the 2D RNN model to the state-of-the-art systems from the literature on the WMT English→German benchmark task as before in Table 4.8, where our model has no superiority compared to deep models.

4.5.2 Two-Way Machine Translation

Setups

For the German→English task, we use the same self-attentive model as before. The same setup is used to train the reverse English→German direction. We clarify that an independent model is

Table 4.8: Comparison of the 2D RNN model with the recently published works on WMT 2014 English→German. Massive HW refers to massive GPU budgets to train models such that "no" means single-GPU training, while "yes" means multiple GPU access in that the number of nodes/machines can vary.

method	BLEU$^{[\%]}$	**#**	**massive**
	newstest2014	**parameters**	**HW**
DeepL translator (evaluated in 2020)	33.1	-	-
RNN attention (GNMT) [Wu & Schuster^{+} 16]	24.7	-	yes
deep RNN attention (RNMT+)[Chen & Firat^{+} 18]	28.5	378.9M	yes
CNN attention [Gehring & Auli^{+} 17b]	25.2	263.4M	yes
CNN attention (DynamicConv) [Wu & Fan^{+} 19]	29.7	213M	yes
self-attentive [Vaswani & Shazeer^{+} 17]	27.3	93.3M	yes
deep self-attentive [Ott & Edunov^{+} 18]	29.3	210M	yes
deep self-attentive [Liu & Duh^{+} 20]	30.1	256M	no
2D RNN	27.7	227M	no

"-": not available

trained for each direction.

The two-way model is composed of 4 layers of masked multihead self-attention encoders on both the source and target sides with 8 heads each. The encoder must be a unidirectional layer to keep a bounded history restriction. The 2DLSTM layer has 750 hidden nodes. An L_2 norm of 0.05 is also used with a dropout rate of 0.1. Adam updates the model parameters with an initial learning rate of 0.0005 and a dropout of 0.3 on masked self-attention layers. The rest of the configuration is the same as the baseline setup, except for the maximum source sequence length, chosen to be 50 subwords during training.

In addition to WMT German↔English, we conduct experiments on a low-resource task of WMT 2017 Turkish↔English including only the `SETIMES2` corpus with 207k sentence pairs. For Turkish↔English, we use the common preprocessing steps: tokenization, true-casing, and 20k joint BPE symbols. Such segmentation results in 5.9M and 5.8M subwords on the Turkish and English sides, respectively. The English and Turkish vocabulary sizes are 14k and 10k. `newsdev2016` is considered as the development set while `newstest2016`, and `newstest2017` as the test sets.

Similar to German↔English, we utilize the base self-attentive model for both Turkish→English and English→Turkish directions. We only train a single two-way model that requires jointly optimizing the network parameters for both language directions.

Results

Tables 4.9 and 4.10 show the results for the German↔English and Turkish↔English tasks, respectively. In all directions, the two-way model is far behind the self-attentive model. This difference is larger on the high-resource task than on the low-resource scenario. These results indicate that building a single model that translates reliably for both directions is a difficult task. There are some potential reasons for such a huge gap.

The main cause of the performance drop is because the two-way model masks out the future tokens on both the source and target sides and has no access to the full context. Replacing the unbounded encoder layers of the self-attentive architecture with the unidirectional version might explain such performance degradation. Secondly, the joint training of the two-way model requires more sophisticated optimization compared to independent training. For joint training where a new training objective combines likelihoods in two directions, we need to compromise between

Table 4.9: Translation performance of the two-way model WMT 2018 German↔English. The two-way model is a single model for both translation directions. All the model parameters are jointly trained to carry out source-to-target and target-to-source translation. This is in contrast to the self-attentive models that are separately trained for each language direction.

model	**task**	**dev**		**newstest2017**		**newstest2018**	
		Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	Bleu[%]	Ter[%]
unidirectional self-attentive	German→English	32.4	55.8	33.2	55.3	40.4	46.8
	English→German	28.5	62.1	27.2	63.8	40.2	49.6
single bidirectional two-way	German→English	22.7	66.4	23.2	66.7	28.9	59.1
	English→German	18.9	73.3	17.9	74.8	24.5	66.3

Table 4.10: Translation performance of the two-way model on WMT 2018 Turkish↔English. The multilingual setup employs a single self-attentive model where the source-to-target and target-to-source data are combined.

model	**task**	**dev**		**newstest2016**		**newstest2017**	
		Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	Bleu[%]	Ter[%]
unidirectional self-attentive	Turkish→English	21.1	68.6	19.2	71.4	18.7	72.1
	English→Turkish	10.8	90.7	12.0	89.5	12.5	89.6
single bidirectional two-way	Turkish→English	15.6	70.5	14.1	72.4	14.1	72.3
	English→Turkish	8.0	91.2	9.0	90.6	9.5	90.4
single multilingual	Turkish→English	19.7	61.5	18.2	63.7	18.1	63.8
	English→Turkish	10.8	90.7	12.1	89.5	12.6	89.3

two tasks, and the parameters are updated jointly. Therefore, the training is subject to be more complex and might result in a suboptimal solution for the entire optimization problem. This is in contrast to separate training and updating for two independent sets of parameters.

We also set up a multilingual system for the Turkish↔English task shown in the third block of Table 4.10. Proposed by [Johnson & Schuster$^+$ 17], in the data preparation phase, we concatenate the source-to-target and target-to-source variants of the training data with a special token for each direction. To translate from Turkish→English, the token `@en@` is inserted, whereas for the reverse translation from English to Turkish, the token `@tr@` is added to the source side of data. Given that, a single self-attentive model is trained as the multilingual system. As listed in Table 4.10, the performance of the multilingual model is slightly worse than that of the baseline and outperforms the two-way model.

4.5.3 Automatic Speech Recognition

Setups

As in Section 3.8.2, the experiments are done on Switchboard 300h and LibriSpeech 960h with the same baselines.

The 2D RNN model is equipped with 6 stacked BiLSTMs of size 1024 with interleaved max-pooling between the first three layers. The only difference compared to the RNN attention model is that in the 2D RNN setup, a reduction factor of 8 (a factor of 2 between the first three layers) is used instead of 6. The decoder is composed of a single layer of 2DLSTM of size 1024 that processes

Table 4.11: The uni- and bidirectional 2D RNN model performance using different aggregations on Switchboard. No language model is used in decoding. The 2D RNN model has 160M and 155M parameters with the forward and backward layers. AM abbreviates the acoustic model.

AM			WER[%]			
forward	**backward**	**aggregation**	**Hub5'00**			**Hub5'01**
			CH	**SWB**	Σ	
2DLSTM	no	max-pooling	21.5	10.2	15.9	15.5
		average-pooling	23.4	11.3	17.4	16.5
		last state	24.5	11.6	18.1	16.9
		weighted sum	22.2	10.2	16.2	15.5
	2DLSTM	max-pooling	21.6	10.7	16.2	15.6
		average-pooling	23.2	11.4	17.3	16.1
		last state	23.4	10.9	17.2	16.4
		weighted sum	21.7	10.5	16.1	15.4

the encoder states in the forward direction and generates the output label sequence (see Figure 4.3). Similar to the RNN attention baseline, the embedding layers have 512-dimensional vectors. We apply maxout before the softmax layer. Similar to the machine translation experiments, we also explore the extension of the network using the backward 2DLSTM layer and various aggregations in the initial experiments. In the case of an additional backward 2DLSTM layer, we halve the feature dimension such that each direction has a size of 512, a total of 1024.

The main training configurations are similar to those in the RNN attention model as explained in Section 3.8.2 for both Switchboard and LibriSpeech tasks, except for a lower initial learning rate of 0.0005 and a batch size of almost 8k with gradient accumulation of 4 to 8 steps. We note that the batch size can be slightly different from one setup to another, and it is specified to be as big as possible to fit on a single GPU. In the same way, a warm-up of 10 steps is taken at the beginning. In the first set of experiments, dropout between 2DLSTM recurrent weights is not activated, while later on, we add a rate of 0.3.

Results

Aggregation Functions and Backward Layer Similar to machine translation, we first look at the performance of the aggregation mechanisms and the effect of the backward layer. In contrast to the translation experiments, here when we apply the backward layer, half of the parameters are used in the forward layer and the remaining half in the backward layer. The performance of different aggregation functions on Switchboard with and without the backward layer is given in the first and second blocks of Table 4.11. The max-pooling and weighted sum operations show better WER by more than 1% absolute than the other aggregations. This is also true for the 2D RNN model with the backward layer. The results on machine translation are very similar. Interestingly, the weighted sum function works as good as max-pooling on average (compare lines 1 with 4 and lines 5 with 8). Max-pooling selects the most active speech encoder states; however, it does not provide an explicit normalized weighting distribution.

Recurrent Weight Dropout We do not apply dropout on the recurrent nodes of the 2DLSTM cells in the previous set of experiments. Table 4.12 presents the performance of the same models with extra recurrent dropout regularization. By comparing Tables 4.11 and 4.12, it is seen that

Table 4.12: The uni- and bidirectional 2D RNN model performance on Switchboard when dropout with a rate of 0.3 is applied between the recurrent time steps in 2DLSTM. No language model is used.

AM			WER [%]			
forward	**backward**	**aggregation**	**Hub5'00**			**Hub5'01**
			CH	**SWB**	**Σ**	
2DLSTM	no	max-pooling	21.1	9.8	15.5	15.1
		weighted sum	21.0	10.3	15.7	14.8
	yes	max-pooling	21.9	10.5	16.2	15.5
		weighted sum	21.3	10.2	15.8	15.2

Table 4.13: The uni- and bidirectional 2D RNN model performance on LibriSpeech. Dropout with a rate of 0.3 is applied between the recurrent time steps in 2DLSTM. The aggregation operation is chosen to be max-pooling. The effective number of epochs is 25. No language model is used.

AM		WER [%]			
forward	**backward**	**dev**		**test**	
		clean	**other**	**clean**	**other**
2DLSTM	no	4.1	11.4	4.3	11.7
	yes	4.3	11.7	4.3	12.4

recurrent weight dropout helps the 2D RNN model regardless of applying the backward layer or the type of aggregation functions, with more emphasis on the case without a look-ahead.

In the end, it is inconclusive which operator is a better design choice. Here, we select max-pooling as it is weightless. We recommend taking a weighted sum with by-product alignments for those applications in which attention scores or implicit alignments are needed. In the remaining experiments, we use dropout and max-pooling. The backward layer does not contribute to 2D modeling due to the use of bidirectional LSTM in the encoder. This statement is experimentally proven on the LibriSpeech task given in Table 4.13. In this comparison, adding the backward layer leads to a drop of 0.7% absolute WER on `test-other`. Therefore, no backward layer is applied in the rest of the experiments.

Recognition Performance We further compare the performance of the 2D RNN model with the baselines from Chapter 3. Table 4.14 shows the comparison on the Switchboard test sets, respectively Table 4.15 compares the results on LibriSpeech. On Switchboard, the 2D RNN model performs on par with RNN attention when no language model is used. It is slightly better than the self-attentive model. The LM scale λ_{LM} is tuned to be 0.4 for fusing LM with the 2D model for Switchboard. In addition, integrating the RNN and self-attentive-based language models to the 2D RNN model reduces the error rate by 0.6% and 0.9% on the `Hub5'01` test set. Language models compensate for the performance gap of models; however, the 2D RNN model performs slightly worse on average compared to the baselines.

This is slightly different in the case of the second task, as shown in Table 4.15. We observe that the 2D RNN model benefits most from the language model combination. It performs almost equally or somewhat better compared to the baselines if an LM is integrated. Here, the LM scale is 0.2. One associated problem with the 2D RNN model is the decoding speed, which is almost 6

Table 4.14: Performance comparison between the 2D RNN model and baselines on Switchboard with and without the language models in shallow fusion decoding.

AM	**LM**	WER$^{[\%]}$				**#**
		Hub5'00			**Hub5'01**	**parameters**
		CH	**SWB**	Σ		
RNN attention	none	21.1	10.1	15.6	15.0	157M
	RNN	20.6	9.2	14.9	14.1	
	self-attentive	20.2	9.1	14.7	13.9	
self-attentive	none	21.2	10.3	15.8	15.3	99.5M
	RNN	20.7	9.6	15.2	14.3	
	self-attentive	20.7	9.5	15.1	14.0	
2D RNN	none	21.1	9.8	15.5	15.1	160M
	RNN	20.7	9.2	15.0	14.5	
	self-attentive	20.8	9.1	15.0	14.2	

Table 4.15: Performance comparison between the 2D RNN model and baselines on LibriSpeech with and without the language models in shallow fusion decoding. The number of effective epochs is chosen to be 25.

AM	**LM**	WER$^{[\%]}$				**#**
		dev		**test**		**parameters**
		clean	**other**	**clean**	**other**	
RNN attention	none	3.7	10.2	3.8	10.8	188M
	RNN	2.8	7.7	3.2	8.4	
	self-attentive	2.6	7.2	2.7	7.7	
self-attentive	none	4.0	10.2	4.1	10.7	139M
	RNN	2.9	7.6	3.2	8.1	
	self-attentive	2.7	7.3	3.1	7.7	
2D RNN	none	4.0	10.9	4.1	11.3	192M
	RNN	2.7	7.9	2.9	8.1	
	self-attentive	2.6	7.2	2.7	7.6	

times slower than the fastest model, i.e., the RNN attention model. This may limit its usage in production systems.

Comparison with Other Works Similar to the previous chapter, Table 4.16 gives a performance comparison of recently published models. Due to the limited hardware budget of the 2D RNN training, there is no direct comparison to other state-of-the-art methods.

4.6 Implementation

The two-dimensional sequence-to-sequence modeling is based on an efficient GPU implementation of multidimensional RNN made publicly available as part of RETURNN.

The first attempt to use 2DLSTM for sequence-to-sequence modeling returns to our implementation directly in Theano. For 2DLSTM, the input tensor x has dimensions (T, T', B, d_h), where T is the horizontal length, T' is the vertical length, B is the batch size, and d_h is the feature dimension. For 2DLSTM, two nested loops are needed, one over the vertical and one over

Table 4.16: Comparison of the 2D RNN model with the recently published works on LibriSpeech. The models are not necessarily trained with the comparable number of parameters over the same number of epochs.

method	**AM**	**LM**	WER[%] test clean	WER[%] test other	**# parameters (AM†)**	**# epochs (AM†)**
hybrid HMM	RNN [Lüscher & Beck+ 19]	self-attentive	2.3	5.0	160M	12.5
	self-attentive [Wang & Mohamed+ 20]	self-attentive	2.3	4.9	149M	100
CTC	CNN [Kriman & Beliaev+ 20]	self-attentive	2.7	7.3	19M	400
attention	RNN [Park & Chan+ 19]	RNN	2.5	5.8	>200M	600
	RNN [Karita & Wang+ 19]	RNN	3.3	10.8	-	100
	self-attentive [Karita & Wang+ 19]	RNN	2.6	5.7	-	200
	CNN [Hannun & Lee+ 19]	CNN	3.3	9.8	37M	-
	self-attentive [Synnaeve & Xu+ 20]	self-attentive	2.4	5.2	270M	-
transducer	self-attentive [Zhang & Lu+ 20]	self-attentive	2.0	4.6	139M	-
	CNN-RNN [Han & Zhang+ 20]	RNN	1.9	4.1	113M	-
	Conformer [Gulati & Qin+ 20]	RNN	1.9	3.9	119M	-
this work	2D RNN	self-attentive	2.7	7.6	192M	25

"-": not available.

† measures only for the acoustic models.

the horizontal axis, using the scan function, i.e., iteration over the T and T' dimensions. This implementation indeed works but is too slow to train a network in a reasonable amount of time. Therefore, we explored the C++ and CUDA implementation provided in RETURNN developed by Paul Voigtlaender as part of his Master's thesis [Voigtlaender 16]. We first verified the correctness of the gradient computations numerically and compared the results to the slow Theano-based 2DLSTM implementation. The actual kernel executions speed up the training quite a lot. Such observations are also seen by [Voigtlaender 16] on a synthetic task of handwriting recognition.

The implementation is a fast custom C++/CUDA code for both the standard LSTM and 2DLSTM kernels. CUDA, Nvidia parallel computing framework, and its extension libraries like cuBLAS and cuDNN provide an efficient GPU-accelerated implementation of matrix-matrix multiplications.

Besides cuBLAS for all matrix multiplications, the custom 2DLSTM implementation uses the custom CUDA kernels for the activation functions and the LSTM gating mechanism. Another optimization that is widely used in standard LSTM implementations is that the five forward weight matrices (see Equations 4.1-4.5) are stacked into a large weight matrix. The same is also done for the recurrent weights and bias terms. Although this does not reduce the total number of floating-point operations, it diminishes the total number of matrix multiplications. Such stacking can lower the number of CUDA kernel invocations since launching many small kernels can cause a substantial kernel launch overhead [Voigtlaender 16]. Furthermore, the implementation works inplace as much as possible to reuse memory. It also enables mini-batch training where multiple input sequences with different sizes are packed together. Each sequence is padded to the length of the longest sequence in the mini-batch.

In order to integrate the custom CUDA code into Theano, we wrote a custom Theano operation. Later, we also wrapped the CUDA code in a custom TensorFlow operation so that RETURNN can easily access it. Once we have the 2DLSTM CUDA kernel, i.e., a program directly executed on the GPU, we define the output of the function as appropriate for our sequence modeling purposes. We also adapted and implemented multidirectional 2DLSTM and used the batched operations to process one diagonal of two directions while exploiting additional parallelism. Different aggregations were added as features to the corresponding class. The code was further developed for two-way translation modeling in which we implemented one two-directional search algorithm by switching between the vertical and horizontal axes.

4.7 Conclusion

In this chapter, we presented our work on the 2D RNN model, a two-dimensional sequence-to-sequence model that jointly encodes the input and output sequences with the two-dimensional long short-term memory cells. With this joint representation, the input and output sequences meet at the final stage of encoding, allowing each of them to affect the representation of the other. We examined different aggregation methods to collapse the 2D output to a 1D sequence along one axis. Based on Tables 4.1 and 4.2 on machine translation and Table 4.11 on automatic speech recognition, max-pooling and weighted sum led to the best design choice on the final performance. We selected the former as it is weightless. We also investigated the effect of bidirectionality on the 2D RNN models where the forward and backward 2DLSTM units are concatenated to capture look-ahead information. We showed that its influence on top of the bidirectional context was suboptimal. Final experiments on three WMT shared tasks listed in Tables 4.4 to 4.6 indicated that the 2D RNN models were comparable with RNN attention in terms of Bleu, but fall behind self-attentive models. According to Tables 4.14 and 4.15, comparison against attention models on Switchboard and LibriSpeech tasks also proved the competitiveness of the 2D RNN models for speech recognition. Language model integration helped the final performance of the 2D model such that it compensates for its performance gap with the attention baselines. Furthermore, the 2D RNN models showed less sensitivity with respect to the source sequence lengths, as shown in Figure 4.9. This implied a better generalization ability over unseen sequence lengths, especially compared to the self-attentive models without relative positional encoding. Finally, we extended the concept of 2D sequence-to-sequence modeling to two-way translation, where we introduced an interesting capacity of our architecture to translate both source-to-target and target-to-source directions using a single neural network. Although the experimental results were at their initial stages and much poorer than the well-tuned baselines, we believe that the 2D RNN-based models can potentially be applied to other applications where sequence-to-sequence modeling is established as a crucial modeling choice.

4.8 Related Work

Directed acyclic graph recurrent neural networks provide a way of building multidimensional context into recurrent networks, a strategy that is based on networks with tree-structured update graphs [Baldi & Pollastri 03]. There are remotely related works where they scan both vertical and horizontal axes of images interpreted as 2D arrangement of features. The multidimensional LSTM [Graves & Schmidhuber 08, Graves 08] is successfully used in handwriting recognition to automatically extract features from raw images [Strauß & Grüning$^+$ 16] which are inherently two-dimensional. In this application, the 2DLSTM layers show better performance over convolutional neural networks. Larger 2DLSTM layers are explored for deeper and wider architectures using GPU implementation on which our implementation depends [Voigtlaender & Doetsch$^+$ 16].

Different neural networks have been also proposed in automatic speech recognition to model 2D correlations in the input signal. One of these uses 2DLSTM that scans the input over both time and frequency jointly for spatio-temporal modeling [Li & Mohamed$^+$ 16]. Moreover, various architectures to model time-frequency patterns based on deep DNN, CNN, RNN, and 2DLSTM layers are compared for large-vocabulary automatic speech recognition [Sainath & Li 16]. This shows the success of the 2D grid as a feature extractor.

All the above works apply 2D grids on the spectrogram level on input layers to extract features for handwriting or speech recognition. We argue that in either case, it is a two-dimensional interpretation of the input data. In contrast, our work uses the 2D arrangement of points to incorporate the input and output for sequence modeling. Our work differs from existing related architectures that operate along the spatio-temporal dimensions of the data. Our work provides a unified way of using 2DLSTM for sequential computation over two dimensions.

Beyond one-dimensional encoding, a de-facto implementation of a two-dimensional layout is a network of one-dimensional units from parallel sequences which are weaved and arranged in a multidimensional grid. A 2D grid can be created by means of standard LSTM cells communicating not only across the time sequence but also between layers [Kalchbrenner & Danihelka$^+$ 16]. This paper uses a grid of LSTMs in machine translation to re-encode source sentences based on target words generated up to that position. However, their experiments are carried out on a very small dataset, and it is not clear whether the grid has the potential to scale on real-world tasks. Similarly, another 2D-based layout as a translation model is investigated where the source and target sentences are jointly encoded; however, the model is behind the self-attentive baseline [Baquero-Arnal & Iranzo-Sánchez$^+$ 19].

The grid LSTM network is also applied for the endpoint detection task in speech recognition to model both spectral and temporal variations [Chang & Li$^+$ 17]. A 2D attention matrix is also applied in a neural pitch accent recognition model [Bruguier & Zen$^+$ 18] in which graphemes are encoded in one dimension, and audio frames are encoded in the other.

In the same line of motivation to our work, there is also a group of works in which convolutional networks process a 2D grid composed of input and output embedding vectors. Concurrent with us is the pervasive attention model in which a stack of 2D convolutional neural networks is used on top of a joint representation from the source and target embedding. It defines a 2D grid from the positions of the source and target words for translation modeling [Elbayad & Besacier$^+$ 18]. This differs from our architecture with recurrency over two sequences. This model is not superior to the attention model. Similarly, in [Li & Wang$^+$ 20], the source and target embedding can be jointly combined, and instead of convolutions, the self-attention layers are performed on two temporal dimensions separately. This model has competitive results with their self-attentive baselines.

Superficially similar to this work, the idea of active memory based on convolution networks is extended in which the netwpork focuses on all encoder states in parallel in a uniform way [Kaiser & Bengio 16]. Similar to our work, the hidden representations are expanded into a 2D structure by recomputing the encoder states after each output token generation. Another work is a 2D construction of input-output joint representation in question answering by weaving two unidimensional recurrent networks [Raison & Mazaré$^+$ 18]. Form a 2D source-target representation is possible by concatenating the pairwise embedding vectors of words on the source and target sentences [Wu & Xia$^+$ 18]. The authors use this model to distinguish between human and machine translation for adversarial learning. The 2D layout of pairwise input-output sequences is also used for scoring parallel texts or matching between written sentences [Hu & Lu$^+$ 14, He & Lin 16].

The 2D-based model presented in this chapter also motivates sequence-to-sequence models without attention [Kalchbrenner & Blunsom 13, Cho & van Merrienboer$^+$ 14a, Sutskever & Vinyals$^+$ 14] as well as neural models in which the encoder and decoders are not separated.

Having no attention concept, an eager translation model is introduced in [Press & Smith 18] where the decoder starts translating as soon as it receives the first source words. The architecture shows better translation performance on longer sequences.

From the perspective of re-encoding encoder states dynamically while predicting a new target word, the attention mechanism can be also replaced with recurrent layers [Zhang & Xiong$^+$ 16, Zhang & Xiong$^+$ 17a, Zhang & Xiong$^+$ 17b]. Instead of calculating attention weights, RNN layers process the source representations at every target step and internally constructs the context vector that is then used by the decoder. The recurrent layer summarizes the necessary parts of the source sentence directly. The authors argue that the RNN can capture this information to assess which source tokens influence the current target position and which do not. They also show their model's ability on source length generalization. A study of replacing attention components with different recurrent setups, both 1D and 2D recurrency, has also been explored in [Brix 18].

Recurrent attention modeling is closely related to memory networks in which encoder representations can be seen as read-only memory [Weston & Chopra$^+$ 15]. Adding a recurrent attention layer enables both read and write operations. From the view of memory networks, source representations are considered as an interactive memory with the decoder through read and write operations [Meng & Lu$^+$ 16].

In the next chapter, we present a model which modifies the way how the attention mechanism is used as part of the translation network. In the same direction, related works with the concept of neural hidden Markov models are reviewed in which no attention context vector is used; instead an alignment model.

5. Zero-Order Hidden Markov Modeling

5.1 Introduction

Given the source sentence f_1^J, the posterior probability of the target sentence e_1^I is directly defined using the chain rule as written in Equation 3.18. The conditional posterior covers alignments between source and target sentences either implicitly or explicitly. In recent sequence-to-sequence models, the dependency on f_1^J is modeled using an effective attention mechanism as addressed in Section 3.5, where soft attention is used to focus on the most relevant parts of the source sentence while generating target tokens. Attention provides a distribution over the source positions that is considered as an implicit probabilistic notion of alignment. As a generalization of the attention model, we express

$$p(e_1^I|f_1^J) = \prod_{i=1}^{I} p\Big(e_i|e_0^{i-1}, c_i\big(e_0^{i-1}, h_1^J(f_1^J)\big)\Big) \tag{5.1}$$

where the alignment distribution is used as an intermediate step inside the network, and the context vector c_i summarizes it into a single vector given by Equation 3.28. The context vector as a weighted sum of the input states can be seen as a kind of partial decision on the local average observation. As written in Equation 5.1, the source positions and the alignment distribution do not directly influence the final target prediction. Since the number of source positions in machine translation or input frames in speech recognition may vary greatly between target words or labels, respectively, summarizing the input representations as a single vector might not be suitable.

An explicit latent variable might be a potential solution to defining an alignment sequence for such models. This is the same in ASR in that label sequence posterior distribution should cover the alignments between input frames and output labels. The conventional HMM-based ASR models [Robinson 94, Hutter 95] and their variants [Graves & Fernández$^+$ 06, Battenberg & Chen$^+$ 17, Dong & Zhou$^+$ 18, Doetsch & Hegselmann$^+$ 16, Doetsch & Hannemann$^+$ 17] tackle the underlying alignment problem using explicit stochastic latent variables.

It has also been proven that the attention component does not fulfill the same role as the alignment model in conventional statistical machine translation [Koehn & Knowles 17]. Alignments play an essential role in statistical machine translation, both for improving translation quality and for better understanding how the translation is performed. Alignments with an appropriate statistical formulation discussed in Section 1.2.1 allow us to quickly see the important dependencies for a certain translation. In addition, alignments can play a major role where the user desires to influence the translation output [Alkhouli 20] or in general in other downstream natural language processing (NLP) tasks.

Translation model explainability is another crucial concept, particularly for practical machine translation, where emitting explicit word alignment is of great value. Neural machine translation especially has problems in this area. Contrary to conventional alignment models, where a hard alignment is a transparent process, defining the correspondence between source and target words, it is often challenging for the fuzzy attention mechanism to extract a comprehensible alignment. Specifically, the self-attentive models do not guarantee a high-quality alignment, even though alignments can easily be used for explainable machine translation. The internal representations are often in vector spaces with hundreds of dimensions, and the operations are usually nonlinear, further obscuring the decision-making process. Unlike the RNN attention models in which the attention scores can easily be observed, the self-attentive model uses multiple attention heads. Due to this increased complexity, it is not trivial to interpret the results of the self-attentive-based models. On top of that, it is reported that the self-attentive model often concentrates on the end-of-sentence token [Ghader & Monz 17, Alkhouli & Bretschner$^+$ 18, Zenkel & Wuebker$^+$ 19]. On the explainability of attention models, there are some works with the main goal of deriving high-accuracy word alignments [Stahlberg & Saunders$^+$ 18, Zenkel & Wuebker$^+$ 19]. Here, we follow a naïve idea such that the translation of a sentence should be decomposable. This way the correctness of translated words can be seen easier.

Motivated by the fact that statistical word alignments offer pioneering and fundamental concepts in machine translation, and alignment is not a fully stochastic process in attention-based models, this chapter describes a latent variable attention model inspired by HMM formulation as its first contribution. The success of HMM in traditional MT and ASR systems inducts an integration of attention modeling into the HMM concept. Our overarching goal is to improve the performance of current state-of-the-art models and to better understand how translation has been performed.

We introduce a latent variable attention model that can be interpreted as a mixture of multinomials. From the statistical viewpoint, we revisit the target sentence posterior probability formulation by calculating a separate translation score for every target-source word pair. The model assumes a decomposition of the target sequence posterior into lexicon and alignment probabilities. Since marginalization has an exponential number of terms in the alignment order of the model, a zero-order assumption is studied where there is no dependency between subsequent alignments. The zero-order assumption keeps the explicit alignment as part of the stochastic model and combines the ease of the end-to-end training of the attention model with an efficient and simple beam search. This is unlike higher-order models [Alkhouli & Bretschner$^+$ 16, Wang & Alkhouli$^+$ 17, Alkhouli & Bretschner$^+$ 18, Wang & Zhu$^+$ 18]. Our next contribution lies in investigating how the HMM decomposition affects both translation performance and alignment errors. Regarding the interest in alignment-based approaches, we also concentrate on the problem of explainability of alignments, especially for self-attentive models. We follow the same formulation and extend the zero-order HMM model for automatic speech recognition.

After providing a mathematical model in Section 5.2, we relate the model to elements from HMM with discussions on training and search. Section 5.3 explains the model parameterization using both the RNN and self-attention neural network design choices and compares it with the attention mechanism from the statistical and modeling perspective. Before presenting experimental results and the corresponding analyses in Section 5.5, an approximation is addressed that speeds up the model in Section 5.4. To study the model's effect, we also quantitatively and qualitatively analyze the extracted alignments, showing that more focused alignments for better interpretability. Although we introduce the model and its dependency using MT notation, we also apply it to speech recognition, the experimental results of which are discussed in Section 5.5.2. This is followed by an implementation focus in Section 5.6, and the chapter is finished with a conclusion and related works in Sections 5.7 and 5.8.

5.2 Model and Formulation

Together with IBM models [Brown & Pietra$^+$ 93], HMM [Vogel & Ney$^+$ 96] has been used to align source and target words in conventional machine translation. Conceptually similar to

HMM (see statistical word alignment in Section 1.2.1), we introduce alignments as a sequence of stochastic latent variables b_1^I that establishes a mapping from position i on the target sentence to position j on the source sentence. Mathematically, we consider an alignment from each target position i to exactly one source position j, i.e., $i \rightarrow b_i = j$. Here, the mapping is from the output (target) to the input (source). This is different from what is used in the original HMM model by [Vogel & Ney$^+$ 96], where a mapping from the source to the target is considered. Therefore, our model is often categorized as the *direct HMM* model.

Introducing alignments as a sequence of hidden variables, all possible alignments are marginalized out in Equation 5.2, such that b_i aligns the target position i to the source position j. We decompose the joint distribution into the *alignment* and *lexicon* models, obtaining Equation 5.4. Performing the sum over all alignments would lead to a combinatorial problem with exponential complexity in the source sentence's length, i.e., $\mathcal{O}(I^J)$. A high-order model conditioned on the full alignment path b_1^I makes the computation of sum intractable.

Assuming the first-order Markov assumption with respect to the alignment dependencies in Equation 5.5 results in first-order HMM [Wang & Alkhouli$^+$ 17, Wang & Zhu$^+$ 18]. Note that the model has a first-order dependency with respect to the alignment, while the alignment dependency is restricted to the current aligned position $b_i = j$ in the lexicon model. This model is an expressive assumption with feasible complexity and requires a specific training procedure that involves the forward-backward (Baum-Welch) algorithm with EM-like training. This algorithm is based on the principle of dynamic programming where the computation of the current step relies on that of the previous step. The number of alignment paths explored in search is reduced without any approximation; however, the model still has the computational complexity $\mathcal{O}(J^2 \times I)$[1].

The posterior decomposition is thus stated as follows

$$\begin{aligned}
p(e_1^I|f_1^J) &= \sum_{b_1^I} p(e_1^I, b_1^I|f_1^J) && (5.2)\\
&= \sum_{b_1^I} \prod_{i=1}^{I} p(e_i, b_i|e_0^{i-1}, b_0^{i-1}, f_1^J) && (5.3)\\
&= \sum_{b_1^I} \prod_{i=1}^{I} \underbrace{p(e_i|e_0^{i-1}, b_0^{i}, f_1^J)}_{\text{lexicon model}} \underbrace{p(b_i|e_0^{i-1}, b_0^{i-1}, f_1^J)}_{\text{alignment model}} && (5.4)\\
&= \sum_{b_1^I} \prod_{i=1}^{I} p(e_i|e_0^{i-1}, b_i, f_1^J) p(b_i|e_0^{i-1}, b_{i-1}, f_1^J) && (5.5)\\
&= \sum_{j} \prod_{i=1}^{I} p(e_i|e_0^{i-1}, j, f_1^J) p(j|e_0^{i-1}, f_1^J) && (5.6)\\
&= \prod_{i=1}^{I} \sum_{j} p(e_i|e_0^{i-1}, j, f_1^J) p(j|e_0^{i-1}, f_1^J). && (5.7)
\end{aligned}$$

It is not possible to simplify the equation without losing model expressiveness. To allow for efficient marginalization over the alignment variables, we keep the current alignment value independent of previous ones. Assuming a zero-order Markov assumption to simplify the dependencies of the alignment sequences (meaning that the current target word and alignment do not depend on any previous alignment position), we derive Equation 5.6. There is no alignment position dependency on $i - 1$; therefore, we can drop b_i and replace it with j for the zero-order assumption.

[1]In this dissertation, we do not use the forward-backward algorithm to train the models since the focus is on the zero-order dependency.

Due to the zero-order dependency, one can re-arrange the sum and the product in Equation 5.7. This leads to polynomial complexity, i.e., $\mathcal{O}(I \times J)$. We refer the reader to [Brown & Pietra$^+$ 93] for the mathematical proof of swapping between the sum and product. This model is a zero-order model with respect to the alignments, and therefore is also called the *zero-order (direct) HMM* model throughout this work. Although conventional HMM is a first-order model, we can still define the zero-order HMM by removing the dependency on the predecessor alignment. This terminology is equivalent to a mixture model where we average over lexicon probabilities.

The latent variable marginalization becomes simple and efficient for the zero-order model and can be easily applied in both training and decoding. Unlike higher-order dependence models [Alkhouli & Bretschner$^+$ 16, Wang & Alkhouli$^+$ 17, Alkhouli & Bretschner$^+$ 18, Wang & Zhu$^+$ 18], no dynamic programming and no search for the alignment path is required. Hence, a simple beam search decoder is used. The polynomial complexity of the zero-order model is still large to compute efficiently. Therefore, in Section 5.4, we address a speedup technique used in practice.

Relation to IBM-2 Model in Machine Translation The zero-order HMM model is conceptually similar to the IBM-2 model expressed in Section 1.2.1, though with an inverted alignment direction, i.e., from target to source positions. Additionally, the neural model has a dependency on target histories.

5.2.1 Training

The derivative of the cost function in the zero-order HMM model is defined by

$$\frac{\partial}{\partial\theta}\log p_i(e|e_0^{i-1}, f_1^J) = \frac{\partial}{\partial\theta}\log\Big[\sum_{j=1}^{J} p_i(e|e_0^{i-1}, j, f_1^J)p_i(j|e_0^{i-1}, f_1^J)\Big]. \tag{5.8}$$

Analogous to the attention-based models, the cross-entropy is minimized using back-propagation and mini-batch gradient calculations using the Adam update rule (see Section 3.6).

5.2.2 Search

As discussed before, the search procedure finds the most plausible target sentence hypothesis that has the maximum probability score according to the model. Given Bayes' decision rule, the aim is to maximize the argument for the following formula as

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I}{\operatorname{argmax}}\Big\{\frac{1}{I}\prod_{i=1}^{I}\sum_{j=1}^{J} p(e_i|e_0^{i-1}, j, f_1^J)p(j|e_0^{i-1}, f_1^J)\Big\}. \tag{5.9}$$

The translation score is normalized by the target length. Similar to the search procedure for the attention model described in Section 3.7, we define an auxiliary quantity $Q(i, e_0^i)$ at position i for the partial hypothesis e_0^i and obtain the recursive search equation as

$$Q(i, e_0^i) = Q(i-1, e_0^{i-1})\sum_{j=1}^{J} p(e_i|e_0^{i-1}, j, f_1^J)p(j|e_0^{i-1}, f_1^J) \tag{5.10}$$

with the final score of

$$p(e_0^I|f_1^J) = Q(I; e_0^I). \tag{5.11}$$

Given this, the beam search algorithm is used once the alignment and lexicon scores have been computed. It does not require searching over alignment positions due to the zero-order dependency.

5.3 Neural Parameterization

In this section, we describe the individual statistical components of the zero-order model through neural networks. A zero-order HMM has two main quantities: the lexicon and the alignment models needed to be estimated. To parameterize these components, we use both the recurrent and self-attention layers. It is easier to implement the zero-order HMM model in the recurrent network than the self-attentive network with its complicated architecture.

5.3.1 Recurrent Network

Stacked bidirectional LSTM layers are used to compute encoder states h_1^J. In the RNN attention baseline, at each step, the LSTM decoder generates the target token conditioned on the previous one $\tilde{e}_{i-1}$, the last decoder state s_{i-1}, and the context vector c_i. While in this model, we have no context vector.

To compute the lexicon model, we add the last decoder state and the last target token with the encoder states at each position. In other words, it is a target-side language model combined on different encoder representations from time step 1 to J. This can also be interpreted as a hard attention model and expressed as

$$p(e_i|e_0^{i-1}, j, f_1^J) = \text{softmax} \circ \text{linear}(\tilde{e}_{i-1}, s_{i-1}, h_j). \tag{5.12}$$

Contrary to Equation 3.29, the decoder state computation only relies on the last decoder state s_{i-1} and the current target word embedding given by

$$s_i = \text{LSTM}\big(s_{i-1}, \tilde{e}_i\big). \tag{5.13}$$

To form the alignment model which is basically a probability distribution over the source positions, we directly apply the attention weights $\alpha(j|i)$ as computed in Equation 3.34 with the fertility value that is

$$p(j|e_0^{i-1}, f_1^J) = \text{softmax} \circ \text{linear} \circ \tanh\Big(s_{i-1}, h_j, \phi_j \sum_{k=1}^{i-1} \alpha(j|k)\Big). \tag{5.14}$$

Once the lexicon and alignment probabilities have been obtained, we sum over j positions as stated in Equation 5.7. Note that the parameters are shared between the lexicon and alignment models. We highlight the difference between Equations 3.30 and 5.12, namely that there is a direct dependency on temporal positions instead of a summary of all inputs in the single vector c_i. It means we do not separate the influence of each source position on the target positions. To compute the lexicon probability at each target position, we compute softmax over the target vocabulary J times. This is in contrast to the attention-based models where softmax is computed over the target vocabulary only once at each step. This model with the recurrent neural network design is referred to as *RNN zero-order HMM*. Figure 5.1 represents the dependencies within the recurrent encoder-decoder networks.

Combination with Attention Context Vector We can also include the attention context vector in the lexicon part of the zero-order HMM model. The context vector is added to the decoder state as well as the output layer computation. Equations 5.12 and 5.13 are updated to

$$p(e_i|e_0^{i-1}, j, f_1^J) = \text{softmax} \circ \text{linear}(\tilde{e}_{i-1}, s_{i-1}, h_j, c_i), \tag{5.15}$$

$$s_i = \text{LSTM}\big(s_{i-1}, \tilde{e}_i, c_i\big). \tag{5.16}$$

In principle, the attention and zero-order models can have an equal number of parameters. Here, due to the different decoder and encoder sizes, a linear projection layer is used with no maxout layer. Nevertheless, these small changes do not affect the model expressivity.

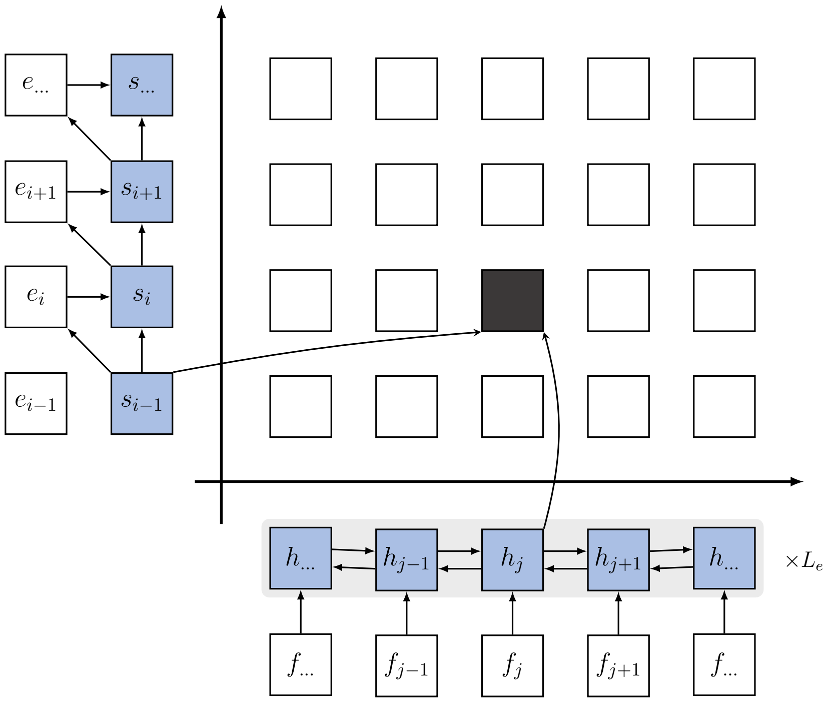


Figure 5.1: Model dependency between the encoder and decoder in the zero-order HMM. The major difference to the RNN attention model is that no context vector is implemented; thus, the computation of the decoder states only depends on the target word histories (cf. Figure 3.5).

5.3.2 Self-Attention Network

As seen in Section 3.5.4, the self-attentive decoder is composed of three main blocks: FFBlock, MaskedMheadSelfAttBlock, and CrossAttBlock. The decoder contains multihead cross-attention which incorporates the representation of the encoder and decoder states. Therefore, every decoder layer is already dependent on the encoder representations and attention information, and thus contains the context vector. The network jointly handles both the lexicon and alignment model at each decoder layer. It is therefore hard to determine a sole lexicon model without any alignment information involved.

A deep stack of L_d decoder layers, functioned by TransDecLayer is used to obtain $c_i^{(L_d)}$ in a similar way to the self-attentive model (see Equation 3.57). Given the last decoder state of the top layer (not all layers) and the last target word for the lexicon model, we combine the encoder states at each time step. In this case, Equation 3.58 is modified to

$$p(e_i|e_0^{i-1}, j, f_1^J) = \mathrm{softmax} \circ \mathrm{linear}(c_i^{(L_d)}, \tilde{e}_{i-1}, h_j^{(L_e)}). \tag{5.17}$$

with $h_j^{(L_e)}$ being the self-attentive encoder of L_e layers as formulated in Equation 3.48. We note that the lexicon model includes attention in such modeling, which means that the context vector information flows into it. Hence, it does not provide a pure lexicon score. We refer to this model as *self-attentive zero-order HMM*; however, it has the context vector.

The self-attentive model contains extra multihead cross-attention layers. To get the attention weights at each layer, several attention heads are available. Given the final encoder representation and decoder states at each layer l, each attention head m computes a normalized distribution over the source positions according to Equation 3.54. To recap, we have

$$\alpha_m^{(l+1)}(j|i) = \mathrm{softmax} \circ \mathrm{DotAtt}(s_{i-1,m}^{(l)}, h_{j,m}^{(L_e)}). \tag{5.18}$$

Table 5.1: Design choices of the zero-order HMM architectures. The RNN attention model (Section 3.5.2), the RNN zero-order HMM model with and without attention context vector (Section 5.3.1), the self-attentive model (Section 3.5.4), and the self-attentive zero-order HMM model (Section 5.3.2).

NN choice	**model**	**encoder**	**decoder**	**attention/alignment**
RNN	attention zero-order HMM	bidirectional RNN	RNN	additive single-head
self-attentive	attention zero-order HMM	self-attentive	masked self-attentive	multiplicative multihead averaged multiplicative multihead

Given multiple self-attention layers with multiple heads, the alignment model is not as trivial to implement. Concerning which head to choose, there are two easy approaches. The first is layer-wise averaging over all heads. The other would be to pick the head with the maximum value for each alignment and finally renormalize the full distribution. We only look at the first approach in this work. Regarding the attention layer, we can average over all attention layers or use a specific layer. Other authors have found that the upper attention layers of the self-attentive model perform longer jumps and generally attend to more different source positions [Irie 20]. Due to this property, we average over the attention heads in the last layer of the decoder at each target position to form the alignment model given by

$$p(j|e_0^{i-1}, f_1^J) = \frac{1}{H}\sum_{m=1}^{H} \alpha_m^{(L_d)}(j|i). \tag{5.19}$$

H is the number of attention heads, and L_d is the last decoder layer. Our preliminary experiments also show that averaging over the last layer performs slightly better than averaging over all layers [Bahar & Makarov$^+$ 20a].

In theory, the zero-order HMM model has more statistical capacity than the attention-based models. The attention model is a deterministic interpolation of deterministic features (encoder states), while zero-order HMM is a mixture model. The network outputs exactly one distribution over the target vocabulary per position with the attention model, while zero-order HMM outputs J distributions, which are mixed probabilistically, yielding a multi-modal marginal distribution. We equip a deterministic soft function with a probabilistic model. Since alignments directly affect the final target sentence posterior, it is worth exploring whether a lexicalized alignment model can assign more appropriate scores.

Tables 5.1 and 5.2 compare the networks' design choices and the dependence of different models.

5.4 Complexity and Speedup Technique

Since the zero-order HMM model combines a mixture of softmaxes, the exact marginalization over all source positions is computationally expensive for a large vocabulary. In other words, to compute the lexicon scores, we need to employ softmax over the target vocabulary $|V_e|$, J times at each time step, in comparison to only once in the case of the attention model. The time complexity of the zero-order HMM model is $\mathcal{O}(I \times J \times |V_e|)$ compared to $\mathcal{O}(I \times |V_e|)$ for the attention model. Thus, the model can become very slow for a large vocabulary setting or long sequences. Using GPU and optimized matrix operations, the computational cost is significantly compensated by parallelization as long as everything fits in memory. Basically, the main efficiency problem of computing the distribution is the linear operation that involves a large matrix multiplication of size $d_h \times |V_e|$ on the output layer. Moreover, the space complexity of the model on the output layer

Table 5.2: A dependency comparison between the RNN attention model (Section 3.5.2), the RNN zero-order HMM model with and without attention context vector (Section 5.3.1), the self-attentive model (Section 3.5.4) and the self-attentive zero-order HMM model (Section 5.3.2) in terms of alignment, decoder, and output layers. The RNN zero-order HMM model has no dependency on the attention context vector; however, this information can be conveyed into the model as listed in line 3. The self-attentive zero-order HMM model, on the other hand, has the attention information due to the inherent topology of the self-attentive architecture. All models use the full source sentence while conditioning on the target history tokens up to position $i-1$.

NN choice	**model**	**NN layers**		
		alignment	**decoder**	**output**
RNN	attention	$\alpha(j\|i)$	$\mathrm{LSTM}(s_{i-1}, \tilde{e}_i, c_i)$	$\mathrm{softmax} \circ \mathrm{linear}(s_{i-1}, \tilde{e}_{i-1}, c_i)$
	zero-order HMM		$\mathrm{LSTM}(s_{i-1}, \tilde{e}_i)$	$\mathrm{softmax} \circ \mathrm{linear}(s_{i-1}, \tilde{e}_{i-1}, h_j)$
	+ context vector		$\mathrm{LSTM}(s_{i-1}, \tilde{e}_i, c_i)$	$\mathrm{softmax} \circ \mathrm{linear}(s_{i-1}, \tilde{e}_{i-1}, h_j, c_i)$
self-attentive	attention	$\alpha_m^{(l)}(j\|i)$	$\mathrm{TrafoDecLayer}(\tilde{e}_{i-1})$	$\mathrm{softmax} \circ \mathrm{linear}(c_i^{L_d})$
	zero-order HMM	$\frac{1}{H}\sum_{m=1}^{H} \alpha_m^{(L_d)}(j\|i)$		$\mathrm{softmax} \circ \mathrm{linear}(c_i^{L_d}, \tilde{e}_{i-1}, h_j^{L_e})$

is $\mathcal{O}(J \times B \times |V_e|)$, rather than $\mathcal{O}(B \times |V_e|)$ for the attention models. This complexity requires decreasing the batch size and confirms the necessity of multiple GPU training or applying some optimization strategies to make the computations more tractable. Both time and space complexities can be either reduced by less computation over source positions or a smaller vocabulary. The output label sequence consists of small-size subwords or characters in speech recognition, thus a small output vocabulary. In contrast, a strong model often relies on a larger vocabulary of 30k to 50k tokens in neural machine translation.

Alternatively, we can conduct less computation in the denominator of softmax normalization. One of the applicable methods is sampled softmax [Jean & Cho$^+$ 15], in which instead of computing the entire matrix multiplication for the target distribution, only the target tokens in the corresponding batch and a random subset of the output vocabulary are considered. All other values are assumed to be zero with a sampling hyperparameter. The samples can be taken from uniform, log uniform, or learned unigram distributions. We refer the reader to [Makarov 19] for a detailed investigation of sample softmax in zero-order HMMs for machine translation. Next, we address an operation by which fewer source positions ($< J$) are counted.

5.4.1 topK Approximation

Motivated by [Shankar & Garg$^+$ 18], we use the topK approximation to select the K most relevant source positions based on the highest probabilities of the alignment model and to compute the corresponding lexicon scores. Formally, Equation 5.7 is approximated as

$$p(e_1^I|f_1^J) \approx \prod_{i=1}^{I} \sum_{j \in \mathrm{topK}} p(e_i|e_0^{i-1}, j, f_1^J)p(j|e_0^{i-1}, f_1^J). \tag{5.20}$$

By doing so, the computational complexity reduces to polynomial complexity $\mathcal{O}(I \times K)$, where $K < J$. This approximation always allows gradients to be computed, leading to stable training for K values larger than 1. Due to the constant nature of the optimization, speed is also greatly improved by more than a factor of two compared to that of the model without this approximation. We have empirically found that $K = 6$ is enough, and we barely get benefits for $K > 6$. Unfortunately, the model still performs about three times slower than its baseline model

Table 5.3: Performance comparison of the zero-order HMM models with ($K = 6$) and without the topK approximation on WMT 2018 German→English. Training speed presents the number of processed tokens per second during training averaged over several epochs.

model	**NN choice**	**topK approx**	**dev** BLEU$^{[\%]}$	TER$^{[\%]}$	**training speed [tokens/sec]**
zero-order HMM	RNN	no	30.5	56.8	289
		yes	30.5	56.9	758
	self-attentive	no	32.3	55.9	496
		yes	32.2	55.5	1402

on a single-GPU setup. An in-depth analysis of different K values with respect to the translation and speed performance of the model will be discussed next.

5.5 Experimental Results

5.5.1 Machine Translation

Setups

The data and baseline models are those explained in Section 3.8.1. In all translation tasks, both the recurrent and self-attention models follow the baseline configurations. Due to the space complexity of the zero-order model, we decrease the batch size. At the same time, the gradient accumulation is increased to 8 and 12 steps for the self-attentive and RNN models, respectively. We also use a lower initial learning rate of 0.0008 for the RNN-based zero-order models. The rest of the training scheme is identical to those of the baselines.

Results

topK Approximation In the topK approximation, zero-order HMM training requires an additional hyperparameter of K. On the WMT German→English task, we explore the effect of changing K on BLEU, based on the `newstest2015` development set. Table 5.3 shows the results with and without this approximation in the zero-order HMM-based models. As shown, the topK approximation with $K = 6$ makes no significant difference in terms of translation performance, though it reduces the time complexity by a factor of almost 2.7. As listed in the table's rightmost column, the model with this approximation processes more tokens. However, it is still slower than the attention baselines as the lexicon scores need to be computed over the target vocabulary $|V_e|$, K times. In theory, the time complexity of the zero-order HMM model is $\mathcal{O}(I \times K \times |V_e|)$ compared to $\mathcal{O}(I \times |V_e|)$ of the baseline. The number of tokens per second is 2.1k and 3.9k for the RNN attention and self-attentive models, respectively. The model is still about 3 times slower than its baseline on a single GPU using this operator. Moreover, the space complexity of the model on the output layer is $\mathcal{O}(K \times |V_e|)$ rather than $\mathcal{O}(|V_e|)$ for the baseline which requires us to decrease the batch size.

Effect of K Figure 5.2 compares more K values concerning BLEU and training speed on the German→English development set using the self-attentive zero-order HMM model. As shown, BLEU goes up from 30.8 to 32.2 by changing K from 2 to 6, and then it levels out without any further significant improvement. After K values exceed 6, performance decreases slightly and plateaus. This implies that small values of K are sufficient for training, and we barely get a better BLEU score for larger values. Intuitively, as K increases, training time goes up. But, it is not

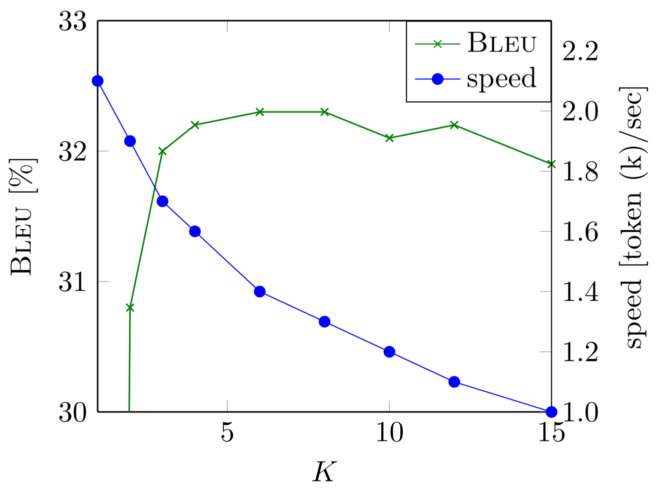


Figure 5.2: Training speed and translation quality versus different values of K in the topK approximation on the development set of WMT 2018 German→English based on the self-attentive zero-order HMM model.

Table 5.4: Comparison of different K values with respect to Bleu score on the development set of WMT 2018 German→English in decoding, while the self-attentive zero-order HMM model is trained using a constant value of $K = 6$.

K	1	2	3	6	10	no approx.
Bleu$^{[\%]}$	30.7	31.9	32.2	32.2	32.2	32.2

linear with respect to K. On the other hand, speed is highest for $K = 1$ and falls logarithmically. Based on this observation, we focus our experiments on applying the topK approximation for all zero-order HMM models with the trade-off value of $K = 6$ and do not tune it further for every task separately.

Similar to [Shankar & Garg$^+$ 18], we also investigate whether the gain is due to the softmax bottleneck and check if larger values of K improve the performance in decoding. To do so, we take the model trained with $K = 6$ and deploy it using different K values in decoding. The results are listed in Table 5.4. The output has only a single softmax vector when $K = 1$. Therefore, the zero-order HMM model encounters the same bottleneck as the self-attentive model. As expected, the performance drops compared to $K = 6$, which means that the model gives some gains behind the ensemble. Performance suffers when K is too small, like $K = 1$, but then plateaus out and does not increase over the value at $K = 6$. The exact marginalization helps neither. Thus, applying the same K value in both training and decoding meets the model requirements.

Translation Performance Tables 5.5 to 5.7 compare the performance of the attention and zero-order HMM models on three translation tasks. On WMT German→English, the RNN-based zero-order HMM model is behind the RNN attention model in both Bleu and Ter, while including the context vector on top leads to an overall average improvement of 0.3% in Bleu and similar in Ter (cf. lines 1 and 3). In principle, the attention and zero-order models can have an identical number of parameters. Here, we use a linear projection layer due to encoder and decoder layer sizes. Another difference is the maxout layer that is omitted in the zero-order models. However, none of the mentioned layers affect the model expressivity while increasing the number of parameters by less than 8%. On the self-attentive modeling where we also have the attention information inside

Table 5.5: Performance comparison between the zero-order HMM models and the baselines on WMT 2018 German→English.

NN choice	model	dev Bleu[%]	dev Ter[%]	newstest2017 Bleu[%]	newstest2017 Ter[%]	newstest2018 Bleu[%]	newstest2018 Ter[%]	# parameters
RNN	attention	31.9	55.3	32.9	55.2	39.7	46.8	217M
	zero-order HMM	30.5	56.9	31.3	56.7	37.4	49.5	225M
	+ context vector	32.2	55.2	33.0	54.9	39.9	46.8	235M
self-attentive	attention	32.4	55.8	33.2	55.3	40.4	46.8	102M
	zero-order HMM	32.2	55.5	33.7	54.9	40.5	46.8	102M

Table 5.6: Performance comparison between the zero-order HMM models and the baselines on WMT 2018 Chinese→English.

NN choice	model	dev Bleu[%]	dev Ter[%]	newstest2017 Bleu[%]	newstest2017 Ter[%]	newstest2018 Bleu[%]	newstest2018 Ter[%]	# parameters
RNN	attention	21.2	63.1	23.2	61.1	22.7	65.9	217M
	zero-order HMM	20.2	64.0	21.5	62.8	21.5	67.0	220M
	+ context vector	21.4	62.4	22.9	60.7	23.0	65.1	234M
self-attentive	attention	22.0	62.8	23.1	61.7	23.0	66.1	101M
	zero-order HMM	22.0	62.7	23.1	61.7	23.6	65.7	101M

the lexicon model, the zero-order HMM model provides 0.3 more Bleu points and decreases Ter by 0.4%. Here, both models have approximately the same number of parameters.

The results on Chinese→English are similar to German→English, but more pronounced in Ter with an average improvement of 0.5%. Adding the context vector to the zero-order HMM model maintains its performance that is not surprising. The results on the English→German task depicted in Table 5.7 are also similar to the opposite direction, such that the average improvement over the RNN baseline is negligible. While we have no gain over the self-attentive model, it did not result in any significant degradation either. In general, it can be concluded that the attention representation as an intermediate step of the network offers sufficient information to produce a reasonable translation.

Alignment Error Rate Besides translation performance as an ultimate goal, we explore the effectiveness of zero-order HMM for alignments. A substantial motivation for the zero-order HMM model is the possibility of improving the attention component of the baselines, especially in the self-attentive architecture, to make it closer to that of traditional alignments generated by IBM and HMM models. To verify this assumption, we use the RWTH German-English golden alignments which provide 505 manually word-aligned sentence pairs extracted from the Europarl corpus [Vilar & Popovic$^+$ 06]. We compare the alignment error rate of the models explained in Section 1.5.4 as a main quantitative metric for the analysis of alignments in a forced decoding scenario in that the target sentence is given. Therefore, we do not accumulate the search errors in the Aer computation. To obtain the alignments for the neural models, we use the maximum of the attention weights as the alignment at each position.

It is plausible that due to different levels of information flow, each layer of the self-attentive architecture provides different attention heads compared to a single layer that has symmetrical attention heads. As observed by [Garg & Peitz$^+$ 19, Voita & Talbot$^+$ 19, Irie & Zeyer$^+$ 19], for the self-attentive model, attention heads seem to fail to learn meaningful alignments in the initial

Table 5.7: Performance comparison between the zero-order HMM models and the baselines on WMT 2014 English→German.

NN choice	model	newstest2013		newstest2014		# parameters
		BLEU [%]	TER [%]	BLEU [%]	TER [%]	
RNN	attention	25.8	56.8	26.9	56.3	224M
	zero-order HMM	24.9	57.4	26.0	57.0	236M
	+ context vector	26.0	56.5	27.4	55.5	247M
self-attentive	attention	26.3	57.0	27.6	56.0	108M
	zero-order HMM	26.3	57.0	27.3	56.4	108M

layers and show a notable improvement in the middle layers. The initial layers serve to learn the representations of the sequence generated by the decoder. The middle layers get the relevant positions from the encoder output via cross-attention, given good contextual representations. It is also reported that the penultimate layer outperforms the final layer in terms of alignment error rate [Garg & Peitz$^+$ 19], indicating that the final layer uses the alignment-based features in the penultimate layer to obtain its representation for the final decision. It also supports our initial observation in which the AER of the self-attentive model significantly improves from 70.5% on the last layer to 55.9% on the fourth layer. This improvement implies that the self-attentive model performs the most interpretable attention not on the last layer, but on the intermediate layers. Our further investigation using entropy in the next section also confirms this. Therefore, we use the average of attention heads of the fourth layer of the self-attentive model to compute AER. Due to the results shown by [Alkhouli & Bretschner$^+$ 18], we compute AER of the zero-order HMM model by averaging over all attention heads of all layers.

It is important to highlight that the manual alignments are given on the word-level calculation, whereas the presented models are trained on the subword-level tokens. To make AER more comparable with traditional systems, we propose two strategies. In the first one, we keep the provided word-level golden alignments as the reference. These results are presented on the left part of Table 5.8. To achieve the alignment of neural models, the BPE subwords are merged together using either the average or maximum attention value of all split tokens to determine which source token corresponds to which target token. This differs from the work by [Sabet & Dufter$^+$ 20] in which subwords are converted to word-level alignments such that two words are aligned whenever all of their subwords are aligned.

We also note that the self-attentive-based models attend many of their heads to the most frequent tokens, e.g., the end-of-sentence token. To mitigate this effect on AER, we ignore this position for the computations. The results are listed in Table 5.8. The average merging performs consistently better than the maximum variant.

In the second technique, in order to have a fair comparison of AER on the models trained on BPEs, we run GIZA++ statistical alignments of the BPE subword level and use them as the reference. GIZA++ performs 5 iterations of the IBM-1, HMM, and IBM-4 models. The models are trained in both translation directions and merged by the grow-diagonal heuristic. In this case, merging subwords to words is no longer required, and we can directly compute the errors on the subword level. The right part of Table 5.8 shows subword-level AER.

Although the best layer is selected that favors better alignments, AER for the self-attentive model is still marginally behind the zero-order HMM model for both word- and subword-level calculations. Comparing AER on word-level and subword-level references, we also observe that combining the subwords for the purpose of alignments is not accurate enough.

Once the most interpretable attention has been extracted for the self-attentive-based models,

Table 5.8: The alignment error rates of the self-attentive-based models on RWTH German→English golden alignments in forced decoding where the target sequence is given. The subwords are merged either by the average or maximum attention values of all split tokens.

Europarl AER$^{[\%]}$					
word-level reference				**subword-level reference**	
attention		**zero-order HMM**		**attention**	**zero-order HMM**
average	**max**	**average**	**max**		
55.9	56.2	54.7	55.1	48.2	47.1

Table 5.9: The alignment error rate of the models on RWTH German→English golden alignments in forced decoding where the target sequence is given. AER of the self-attentive-based models is taken from the best layer listed in Table 5.8.

model	**Europarl** AER$^{[\%]}$	
	word-based reference	**subword-based reference**
GIZA++ (word-based) [Garg & Peitz^{+} 19]	21.4	-
GIZA++ (BPE-based) [Garg & Peitz^{+} 19]	18.9	-
GIZA++ supervised [Garg & Peitz^{+} 19]	16.0	-
[Peter & Nix^{+} 17]	19.0	-
[Zenkel & Wuebker^{+} 19]	21.2	-
[Zenkel & Wuebker^{+} 20]	16.3	-
GIZA++ (word-based)	21.2	-
self-attentive	55.9	48.2
self-attentive zero-order HMM	54.7	47.1
RNN attention	50.3	41.4
RNN zero-order HMM	49.8	40.8
+ context vector	49.0	39.9
combination RNN	49.4	40.3
combination self-attentive	55.6	47.8

"-": not available.

we compare the alignment errors with the statistical alignment model using GIZA++ and the recurrent models. Table 5.9 gives the comparison. AER on the recurrent models is simply taken by the maximum value of the attention distribution. The RNN attention model performs better than both other approaches at 50.3% and 41.4% on word and subword level, respectively, signifying that the self-attentive attention distributions are still far from being easily interpreted as alignments. Applying the zero-order HMM model with a context vector on top of the RNN attention network further improves the alignment error rate by 1.3% and 1.5% absolute.

The alignment quality of the zero-order model is poorer compared to conventional alignment models such as the IBM models or other works in the literature [Peter & Nix^{+} 17, Zenkel & Wuebker^{+} 19, Garg & Peitz^{+} 19, Zenkel & Wuebker^{+} 20]. One potential reason for this is that in the neural models presented in this work, we have no full dependency on the whole target sentence in contrast to the conventional alignment models, and neither we depend on the current target token e_i. Since left-to-right decoding is carried out, we only depend on the predecessor words for an efficient and simple search. The model does not employ additional alignments for supervision, contrary to [Peter & Nix^{+} 17, Zenkel & Wuebker^{+} 19, Garg & Peitz^{+} 19, Zenkel & Wuebker^{+} 20].

Table 5.10: Averaged entropy scores of the attention distributions over the `newstest2018` test set for the self-attentive-based models.

layer	**entropy**	
	attention	**zero-order HMM**
1 (bottom)	1.39	1.28
2	1.20	1.25
3	1.17	1.08
4	1.00	1.20
5	1.10	1.28
6 (top)	1.42	0.18
average	1.21	1.05

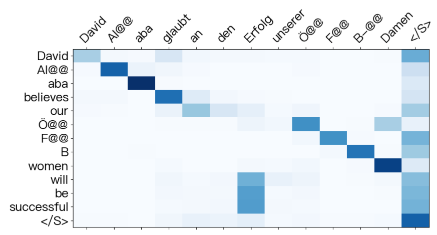
Attention Concentration via Entropy To further understand the attention mechanism's behavior, especially in the self-attentive-based models, we examine attention concentration by calculating the entropy of attention weights proposed by [Ghader & Monz 17]. Contrary to the conventional word alignments, where each target word is assigned to one or more source positions, attention is distributed more freely. The lower the entropy, the sharper attention on a single source token. The exact formula of the entropy is given by

$$E_{att} = -\sum_{j} \alpha(j|i) \log \alpha(j|i). \tag{5.21}$$

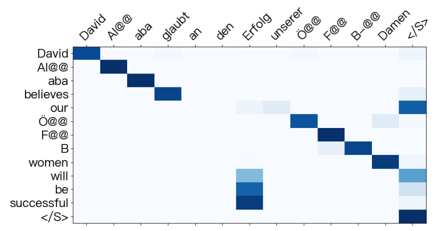
For the experiments, we compute the entropy of the self-attentive-based models again on German→English for the ablation study. The entropy is calculated layer-wise by taking an average across all the heads of the layer. We compute the attention entropy per sentence pair, and then the final reported entropy is the average over the `newstest2018` test set. Table 5.10 shows that for the self-attentive model, attention is most concentrated at the middle layer, with the lowest entropy being on the fourth one. This contrasts with the zero-order HMM that appears at layer 6 with a remarkably lower entropy value than self-attention. Interestingly, these results are in line with our previous observations in terms of Aer and coincide with the findings of [Tang & Sennrich$^+$ 19], saying that the attention weights of layer four have the lowest Aer. We believe that the low entropy of the 6th layer in zero-order HMM comes from using the attention layer in the output, and minimizing cross-entropy also minimizes this quantity more directly than in the standard self-attentive case.

Quantitative Attention Analysis The phenomenon that we observe, which has been also reported by others [Alkhouli & Bretschner$^+$ 18, Zenkel & Wuebker$^+$ 19, Garg & Peitz$^+$ 19], is that the self-attentive network heavily attends to frequent words like the end-of-sentence token, which often has the highest average attention value of the sequence, as seen in the examples in Figures 5.3(a) and 5.3(e). This effect leads to worse interpretation and possibly worse performance since a large portion of the processing capability is spent only on one token. However, the intuition behind the original multihead attention architecture is that it learns different alignment information. If one head learns bad alignments, the other heads may learn good ones. Using an explicit distribution for the alignments, we hope the model will achieve more precise attention weights, solve the end-of-sentence token attendance issue, and lower Aer. What is observed from the quantitative attention analysis is that the zero-order HMM model obtains lower Aer.

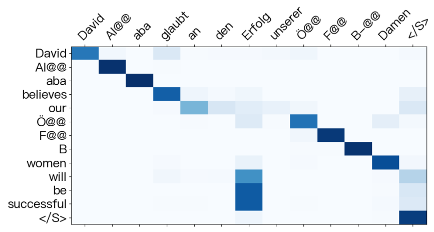
In this part, we qualitatively visualize some randomly selected examples from the test sets of the German→English and Chinese→English tasks demonstrated side by side in Figure 5.3. The



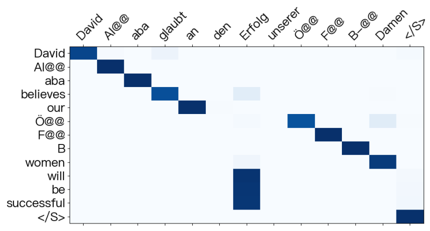
(a) self-attentive



(b) self-attentive zero-order HMM

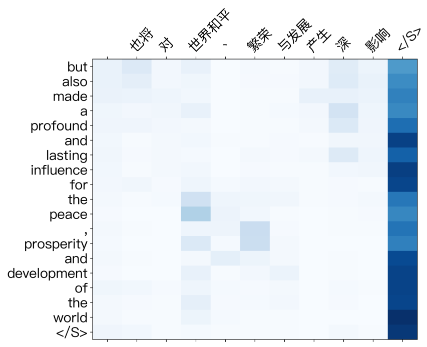


(c) RNN attention

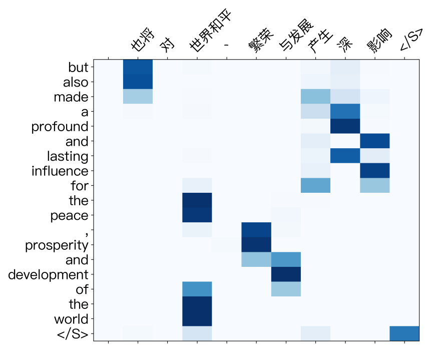


(d) RNN zero-order HMM

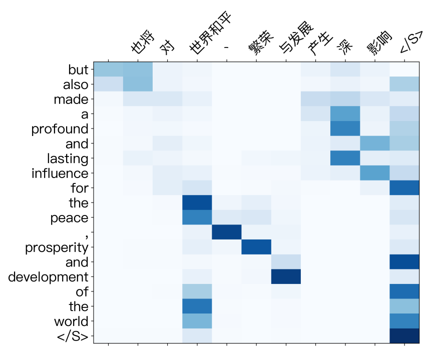
German→English example.



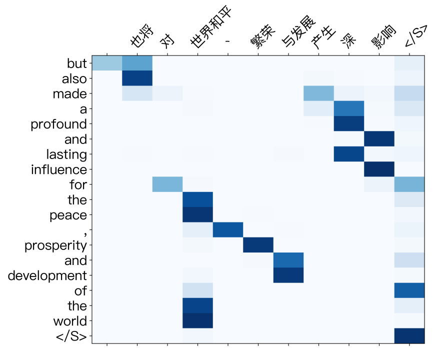
(e) self-attentive



(f) self-attentive zero-order HMM



(g) RNN attention



(h) RNN zero-order HMM

Chinese→English example.

Figure 5.3: Attention distribution of the models in forced decoding where the target sentence is given. The middle layer of the self-attentive model is taken to extract the attention distribution, while in the self-attentive zero-order HMM model, the last layer is used.

horizontal and vertical axes depict the target and source sentences, respectively. To capture the heatmap data, the average is taken over the attention heads of the best layer, i.e. the middle layer in the self-attentive model, the last layer in zero-order HMM. To have a consistent comparison, we run forced decoding experiments in which the reference sentence is given, and we gather the data based on the forward pass. It is, in practice, slightly more focused than the heatmaps generated during decoding.

We also observe relatively bad attention distribution by the zero-order HMM model; however, bad distributions by the standard self-attentive model occur more often. On the German→English example, the zero-order HMM-based models show more focused distributions in comparison to the attention baselines. The Chinese→English example shows an extreme case where the self-attentive attends almost completely to the end-of-sentence token in Figure 5.3(e). At the same time, the zero-order model generates a more understandable attention distribution by attending to other tokens, as shown in Figure 5.3(f).

We also visualize the attention distribution of the RNN attention models to verify that attention issues are specific to the self-attentive model and are not inherent in the soft attention mechanism. As with previous observations, the RNN attention models perform consistently across the datasets, with generally sharper attention than the self-attentive baseline. The RNN zero-order HMM model improves the alignment quality quantitatively, as can be seen in Figures 5.3(d) and 5.3(h), with the attention weights being highly focused.

Although the alignment quality gets better in zero-order HMM modeling, it has no real benefit on the translation performance. The improvement in alignment quality does not carry over to better translation, which is not surprising given prior work [Koehn & Knowles 17]. From the modeling perspective, given a moderate amount of data, we can let the network learn and optimize its parameters based on its objective which is the translation, not the alignment task. In this case, the concept of hard alignment might be unnecessary for the final task if we have enough data.

Model Combination To integrate the attention and zero-order HMM (including the context vector) models on the output layer, we first combine them in decoding by the log-linear combination of model scores equally. The results are shown in Table 5.11 for German→English. Such fusion leads to an improvement both in Bleu and Ter (compare with Table 5.5).

Motivated by the results of the combination in decoding, we additionally integrate the attention model's output with the zero-order HMM model. The model scores are combined on the probability level in training for both the RNN and self-attentive parameterization. Formally, we mix word-level log probabilities given by

$$\mathcal{L} = \lambda \log p(e_i|f_1^J, e_0^{i-1}) + (1-\lambda) \log \sum_{j=1}^{J} p(j|f_1^J, e_0^{i-1}) p(e_i|f_1^J, j, e_0^{i-1}) \tag{5.22}$$

where λ is chosen to be 0.5.

In this combination, all parameters but the last layer are shared and trained jointly; however, the models are independent of being recurrent or self-attention. Unlike the results of the model combination in decoding, the combination of the two models and training them from scratch leads to a performance drop when comparing the numbers with Table 5.5. The log-linear combination also leads to in-between results for the alignments. As shown in the last block of Table 5.9, the alignment error rate is marginally improved over the attention-based models, while it decreases in comparison to zero-order HMM.

Comparison with Other Works We finally compare the zero-order HMM models to similar works by [Wang & Zhu+ 18, Wang & Yang+ 21] where they apply first-order models. Their model can also be parameterized using both RNN and self-attentive networks. Table 5.12 compares

Table 5.11: Log-linear combination of the attention and zero-order HMM models on WMT 2018 German→English in both decoding and training. The attention and zero-order HMM (including the context vector) models are log-linearly combined both in decoding and training with equal weights. Compare to Table 5.5 for relative performance difference from the baselines.

NN choice	**combination in**	**dev**		**newstest2017**		**newstest2018**		**# parameters**
		BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]	
RNN	decoding	32.8	54.4	34.0	54.0	41.1	45.6	452M
	training	31.7	55.8	32.8	55.4	39.7	47.2	257M
self-attentive	decoding	33.1	54.9	34.3	54.3	41.3	45.9	204M
	training	32.0	56.2	33.1	55.8	39.7	47.7	119M

Table 5.12: Performance comparison of zero-order and first-order HMMs ([Wang & Zhu+ 18, Wang & Yang+ 21]) on three translation tasks.

NN choice	**model**	**German→English newstest2017**		**Chinese→English newstest2018**		**English→German newstest2014**		**# parameters**
		BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]	
RNN	first-order HMM	32.4	55.9	22.0	66.6	26.1	56.8	220M
	attention	32.9	55.2	22.7	65.9	26.9	56.3	219M
	zero-order HMM	31.3	56.7	21.5	67.0	26.0	57.0	227M
	+ context vector	33.0	54.9	23.0	65.1	27.4	55.5	238M
self-attentive	first-order HMM	33.8	54.4	23.8	64.7	27.9	55.3	90M
	attention	33.2	55.3	23.0	66.1	27.6	56.0	103M
	zero-order HMM	33.7	54.9	23.6	65.7	27.3	56.4	103M

the performance. RNN first-order HMM has separate lexicon and alignment models, resulting in two neural networks that coincide on the output layer. Including explicit first-order dependency on the alignment (line 1), the RNN-based first-order model outperforms the zero-order model performance in BLEU and TER. However, its performance is still behind the attention models. Adding the context vector to the zero-order HMM model performs best when the RNN design choice is applied. Given the self-attentive network, the first-order model (line 5) is superior to the zero-order HMM and self-attentive baseline. It is important to mention that first-order HMMs also depends on the encoder representations and attention information and includes the context vector. These models also generate more focused alignments closer to GIZA++ [Wang & Zhu+ 18].

As before, we also compare the best obtained model with other works from the literature on the WMT 2014 English→German benchmark task in Table 5.13. None of the zero-order HMM models really outperforms the state-of-the-art models.

5.5.2 Automatic Speech Recognition

Setups

As in Section 3.8.2, we carry out experiments on Switchboard 300h and LibriSpeech 960h with the same RNN and self-attentive baselines. For RNN zero-order HMM, we mainly follow the network structure of the baseline RNN attention model regarding the neural networks' depth and width, with the exception that we do not use the maxout layer on the output, as explained in Equation 5.12. The CTC loss is used during training. We train the model with a smaller batch size and a lower learning rate ($\eta = 0.0008$) while keeping the effective number of epochs to almost

Table 5.13: Comparison of zero-order HMMs with the recently published works on WMT 2014 English→German. Massive hardware refers to massive GPU budgets to train models such that "no" means single-GPU training, while "yes" means multiple GPU access in that the number of nodes/machines can vary.

method	BLEU[%]	**#**	**massive**
	newstest2014	**parameters**	**HW**
DeepL translator (evaluated in 2020)	33.1	-	-
RNN attention (GNMT) [Wu & Schuster+ 16]	24.7	-	yes
deep RNN attention (RNMT+)[Chen & Firat+ 18]	28.5	378.9M	yes
CNN attention [Gehring & Auli+ 17b]	25.2	263.4M	yes
CNN attention (DynamicConv) [Wu & Fan+ 19]	29.7	213M	yes
self-attentive [Vaswani & Shazeer+ 17]	27.3	93.3M	yes
deep self-attentive [Ott & Edunov+ 18]	29.3	210M	yes
deep self-attentive [Liu & Duh+ 20]	30.1	256M	no
RNN zero-order HMM	27.4	247M	no
self-attentive zero-order HMM	27.3	108M	no

"-": not available

33. We use no label smoothing and apply a finer policy of SpecAugment where we mask fewer successive time steps and frequency channels. Observing from the initial experiments, for the self-attentive zero-order HMM model, we wait a bit longer for a maximum of 60 epochs to have a reliable convergence. This is unlike the self-attentive model, which is converged by a maximum of 50 epochs. Unlike the baseline model, we have positional encoding on the output embedding. The batch size is chosen to be 10k tokens with a gradient accumulation of 8 steps. On LibriSpeech, the self-attentive and RNN models have the same configuration as used for the baselines. We train all models for 25 epochs with smaller batch sizes, lower learning rates, and finer SpecAugment.

In general, we observe that convergence of the zero-order HMM model for ASR is harder than that in MT. We additionally have a linear projection on the output layer for the lexicon model, leading to a few more free parameters. After confirming the sufficiency of $K = 6$, as will be addressed later, all models have been trained with the topK optimization.

Results

Effect of K in topK Approximation Similar to the translation experiments, we first investigate the effect of K applied in the topK approximation versus WER. The WER goes down when K changes from 2 to 6, and then it saturates without any further improvement. This implies that small values of K are good enough for training [Bahar & Makarov+ 20b]. Given a trained model using $K = 6$, we also check whether larger K values improve the performance in decoding and confirm that the model does not need a larger K value during decoding.

According to Section 5.4, the time complexity of the zero-order model is K times slower compared to the attention models. However, in end-to-end ASR systems in which the label sequence consists of subwords or characters with small vocabularies, the computational cost is significantly amortized by parallelization, and the zero-order model leads to a negligible slowdown. Compared to MT with larger vocabularies, the ASR experiments are faster. For instance, for Switchboard, the average training speeds over 20 epochs of the RNN attention and zero-order HMM models are 2.7k and 2.4k characters per second, respectively. This means that the model is only 0.2 times slower than its baseline. The same proportion is also observed on the self-attentive models.

Table 5.14: Performance comparison between the zero-order HMMs and baselines on Switchboard with and without the language models in shallow fusion decoding.

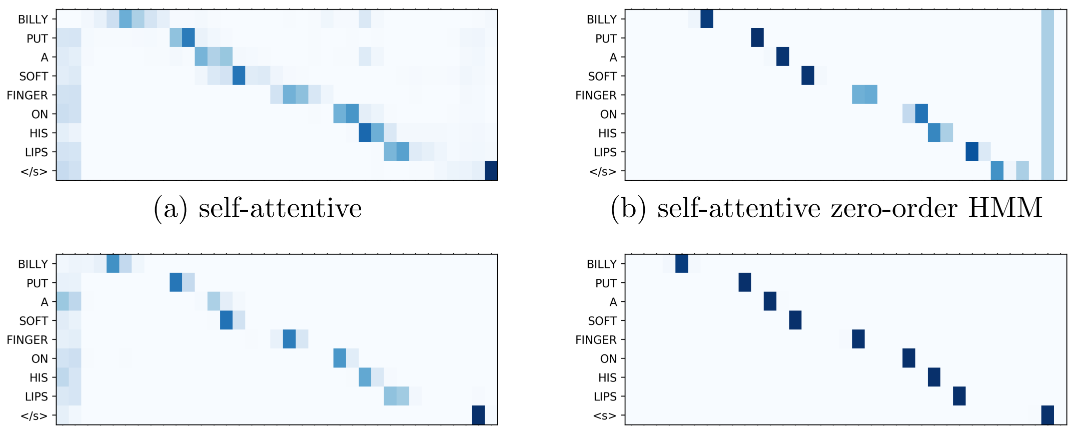
AM		LM	WER[%]				#
NN choice	**model**		**Hub5'00**			**Hub5'01**	**parameters**
			CH	**SWB**	**Σ**		
RNN	attention	none	21.1	10.1	15.6	15.0	157M
		RNN	20.6	9.2	14.9	14.1	
		self-attentive	20.2	9.1	14.7	13.9	
	zero-order HMM	none	21.8	10.4	16.1	15.6	150M
	+ context vector	none	21.1	9.9	15.5	14.9	160M
		RNN	21.2	9.5	15.4	14.3	
		self-attentive	21.0	9.4	15.2	14.1	
self-attentive	attention	none	21.2	10.3	15.8	15.3	99.5M
		RNN	20.7	9.6	15.2	14.3	
		self-attentive	20.7	9.5	15.1	14.0	
	zero-order HMM	none	22.1	10.1	16.1	14.9	101M
		RNN	21.8	9.6	15.7	14.3	
		self-attentive	21.6	9.5	15.6	14.1	

Table 5.15: Performance comparison between the zero-order HMMs and baselines on LibriSpeech with and without the language models in shallow fusion decoding. The effective number of epochs for these experiments is 25.

AM		LM	WER[%]				#
NN choice	**model**		**dev**		**test**		**parameters**
			clean	**other**	**clean**	**other**	
RNN	attention	none	3.7	10.2	3.8	10.8	188M
		RNN	2.8	7.7	3.2	8.4	
		self-attentive	2.3	7.2	2.7	7.7	
	zero-order HMM†	none	4.4	11.7	4.5	12.2	193M
		RNN	3.5	9.8	3.7	10.4	
		self-attentive	3.3	9.4	3.4	9.8	
self-attentive	attention	none	4.0	10.2	4.1	10.7	139M
		RNN	2.9	7.6	3.2	8.1	
		self-attentive	2.7	7.3	3.1	7.7	
	zero-order HMM	none	4.2	11.5	4.4	11.7	139M
		RNN	3.2	9.0	3.5	9.4	
		self-attentive	3.0	8.4	3.3	9.0	

† includes the attention context vector.

Recognition Performance Table 5.14 shows the results of the experiments on Switchboard. On the blind test set `Hub5'01`, the RNN zero-order HMM model is behind the baseline by 0.6% absolute WER when no language model is used. Adding the attention context vector information leads to small but consistent improvements over the RNN attention baseline. Both the RNN and self-attentive language models help the final performance on top, resulting in 14.1% WER. However, after language model integration the RNN attention model performs better. Using the self-attention layers to parameterize the model is weaker than the baseline on all sets with and without the language models.



(a) self-attentive

(b) self-attentive zero-order HMM

(c) RNN attention

(d) RNN zero-order HMM

Figure 5.4: Attention distribution of the models on LibriSpeech `dev-other`. The attention distribution of the self-attentive baseline and zero-order HMM model are taken by averaging across all heads of the 4th and 12th layer, respectively.

The results on LibriSpeech are presented in Table 5.15. Similar to the previous task, the zero-order HMM models including the context vectors are weaker than the attention ones. Here, the error rate difference on `dev-` and `test-other` is a bit larger. Using an external language model also has a big influence on both RNN and self-attentive zero-order HMM models, as intended. The zero-order HMM models do not yet reach the performance of the baselines, therefore more work has to be done to examine possible improvements.

Attention Analysis To support the previous justification, we qualitatively examine the alignment behavior and compare the models' attention distribution. Figure 5.4 illustrates a forced decoding example on one utterance from LibriSpeech `dev-other`. The attention used in the zero-order HMM model shows highly concentrated monotonic alignments compared to the attention models. We have often observed this characteristic over all sets for many utterances independent of the sequence length. The difference in the distributions might be attributed to the way that the alignment is explicitly used in the HMM model and thus predicts the output label differently. Similar to machine translation, for the self-attentive-based models, many attention heads do not perform any different operation and always attend to the same input position throughout an entire sequence. Here, we have taken the best attention heads of all self-attentive-based models, which are the fourth and twelfth layers for the standard and zero-order HMM models.

Comparison with Other Works Given the best models, we compare their performance to those given in the works in the literature. As shown in Table 5.16, the zero-order HMM models perform far behind the hybrid HMM ASR systems, although they are inspired by such models. In either case, we achieve results that are clearly behind the results found in the literature.

5.6 Implementation

This section shortly describes the implementation of the zero-order HMM model. We note that symbolic expressions allow TensorFlow to employ symbolic differentiation. This means that derivatives can be automatically determined without any extra effort to derive them.

Table 5.16: Comparison of the zero-order HMM with the recently published works on LibriSpeech. The models are not necessarily trained with a comparable number of parameters over the same number of epochs.

method	**AM**	**LM**	WER$^{[\%]}$		**# parameters**	**# epochs**
			test		**(AM†)**	**(AM†)**
			clean	**other**		
hybrid HMM	RNN [Lüscher & Beck^{+} 19]	self-attentive	2.3	5.0	160M	12.5
	self-attentive [Wang & Mohamed^{+} 20]	self-attentive	2.3	4.9	149M	100
CTC	CNN [Kriman & Beliaev^{+} 20]	self-attentive	2.7	7.3	19M	400
attention	RNN [Park & Chan^{+} 19]	RNN	2.5	5.8	>200M	600
	RNN [Karita & Wang^{+} 19]	RNN	3.3	10.8	-	100
	self-attentive [Karita & Wang^{+} 19]	RNN	2.6	5.7	-	200
	CNN [Hannun & Lee^{+} 19]	CNN	3.3	9.8	37M	-
	self-attentive [Synnaeve & Xu^{+} 20]	self-attentive	2.4	5.2	270M	-
transducer	self-attentive [Zhang & Lu^{+} 20]	self-attentive	2.0	4.6	139M	-
	CNN-RNN [Han & Zhang^{+} 20]	RNN	1.9	4.1	113M	-
	Conformer [Gulati & Qin^{+} 20]	RNN	1.9	3.9	119M	-
this work	RNN zero-order HMM	self-attentive	3.4	9.8	193M	25
	self-attentive zero-order HMM	self-attentive	3.3	9.0	139M	25

"-": not available.

† measures only for the acoustic models.

Like the 2D RNN model in Chapter 4, the first implementation was done in our Theano-based software with further implementation in RETURNN. As RETURNN simplifies various configurations via a config file, the model was initially built up of individual components of RETURNN without the topK optimization in the configuration file. As each layer of the model is differentiable, the entire model can be trained jointly. Unfortunately, this version cannot use the unrolling technique whereby the recurrency is unfolded along the output axis to process the entire output in parallel. Unrolling is especially important for the self-attentive-based architecture and gives a performance increase of around twofold in our experiments.

To overcome this obstacle, we implemented the model with all its extensions as its own layer written in `TFNetworkHMMFactorization.py` file. The corresponding class can be imported directly into the configuration file. Having the model as a separate layer also allows finer control of the individual tensors, which proved to be important for some experiments.

As mentioned, the zero-order model is inherently slower than attention models due to the lexicon score computation. We found empirically that the computation of the output distribution takes around one-third of all computational time. Therefore, we added some speedup techniques. The first and the most effective one is the topK approximation, as explained in Section 5.4.1, where we presume the top K-scored choices for the alignment model. Its optimization always allows gradients to be computed and thus results in stable training for K values larger than 1. We also implemented a threshold-based approach in which all input positions are set to zero if the alignment model for that position is under a certain value. This only marginally improved speed since TensorFlow currently has a very limited implementation of sparse matrix operations. Moreover, the models are not stably converged with this setup, as the gradients are too noisy in the initial stages of training. The third integration is the well-known sampled softmax that speeds up training by quite a lot while reducing the performance, based on our initial experiments. Commonly used in language model training, the method has not yet reached parity either on MT or ASR tasks. Initial experimental results on WMT 2018 German→English can be found in [Makarov 19].

5.7 Conclusion

In this chapter, zero-order HMM was presented in which we incorporated the attention model into the direct HMM formulation by introducing a latent variable as an alignment sequence. After simplifying the alignment dependency order of the model, we only explored the zero-order assumption. Marginalization over the latent variable becomes simple and efficient for the zero-order models and can be easily applied in both training and decoding without dynamic programming. As speed is a crucial aspect, we investigated a speedup technique that increased both training and decoding speeds approximately by a factor of 3 without loss of translation quality (Table 5.3).

We compared the zero-order HMM model, which is completely independent of attention models, with the RNN attention and self-attentive models, and demonstrated that its performance was behind the attention-based models on translation tasks. We observed that adding the attention context vector made the model as strong as the state-of-the-art methods for machine translation (Tables 5.5 to 5.7). However, the model showed a weaker performance on speech recognition and has not yet reached parity there.

We also studied the alignment quality and showed that zero-order HMM provided more focused alignments in comparison to the attention models, especially the self-attentive model in both text translation and automatic speech recognition, leading to better explainability. The model achieved remarkable improvements in terms of the alignment error rate (Table 5.9). Although the alignment quality got better, its improvement did not transfer to the final translation or recognition performance.

5.8 Related Work

Word alignment as discussed in Section 1.2.1 is one of the key components in traditional phrase-based systems. It can be considered as an explanation for generating a reasonable translation. From the alignment perspective, unsupervised neural alignment models outperform their count-based counterparts in terms of Aer, and the number of unaligned words decreases [Ho & Yvon 19]. Using variational auto-encoders, neural word alignment with latent representations achieves competitive results compared to count-based models [Ho & Yvon 20].

However, neural translation models do not have a notion of hard alignment, even though the soft attention mechanism can be seen as an alignment distribution. It is proven that attention is a poor alternative for this distribution and serves a different purpose [Koehn & Knowles 17]. Since word alignments are of much importance in practical machine translation such as terminology constraints [Hasler & de Gispert$^+$ 18, Alkhouli & Bretschner$^+$ 18], an explicit alignment distribution for neural machine translation is still an open research problem.

We already discussed approaches in which implicit alignment concepts are used within attention-based models as an extension to the attention layer. There are different ways to use precomputed word alignments obtained from the GIZA++ or fast align [Dyer & Chahuneau$^+$ 13] tools. Statistical alignments can be used to supervise the attention model by adding an extra loss term to cross-entropy during training as a supervision signal for the attention weights. Such modeling biases the attention weights towards additional alignment information [Chen & Matusov$^+$ 16, Liu & Utiyama$^+$ 16, Mi & Wang$^+$ 16, Garg & Peitz$^+$ 19].

Including the current target word in the computation in training can reduce the alignment error rate [Peter & Nix$^+$ 17, Li & Liu$^+$ 18]. To obtain high-quality alignments, one recomputes the attention heads after each target word prediction [Zenkel & Wuebker$^+$ 19]. An additional alignment layer is added to the self-attentive model and trained in an unsupervised fashion. This is possible by taking the attention score with which the model can, as closely as possible, predict the correct target word using only the attended source word. The resulting alignments are of higher quality than those from the self-attentive baseline with regard to errors on golden alignments. Translation performance also has minor improvements. In the same line of motivation, instead of an objective function, external count-based alignments are added in the computation of attention weights [Alkhouli & Ney 17]. While this information is given in training, the alignments need to be hypothesized in decoding.

Alignment-based neural machine translation is another category of models inspired by the conventional HMM framework. These models introduce alignment as a hidden stochastic variable and decompose the translation model into alignment and lexicon submodels. In contrast to conventional HMM with simple dependencies, they use neural networks to capture long, even unbounded source and target contexts. Feedforward alignment models with no dependency on lexical context with non-probabilistic scores are utilized to create alignments along with training a separate lexicon model based on the conventional phrase-based system. The model outperforms the HMM and IBM-4 models with respect to AER [Yang & Liu$^+$ 13]. An extension to this paper is by [Tamura & Watanabe$^+$ 14], which uses recurrent networks to generate word alignments with longer dependencies on the alignment history for the phrase-based translation system. Similarly, they use unnormalized scores and report an improvement compared to feedforward models. one can derive a noisy channel framework of inverse translation using alignment-based neural models and a language model [Yu & Blunsom$^+$ 17]. The work constructs a monotonic alignment model, and training is done by marginalizing over the alignment hidden variables. A small translation dataset is used due to expensive computation.

Other neural alignment models employ a feedforward alignment model to score relative jumps over the source positions [Alkhouli & Bretschner$^+$ 16]. The model takes a long-context dependency on the alignment path. Hence, they utilize precomputed Viterbi alignments to generate translation instead of the forward-backward algorithm. Since they use separate alignment and lexical models, it is feasible to hypothesize explicit alignment links during decoding by modifying the beam search algorithm. A similar idea is pursued using recurrent networks in [Alkhouli & Ney 17], and self-attention networks with an additional alignment head in [Alkhouli & Bretschner$^+$ 18]. These models train two separate networks and combine them in decoding with the maximum approximation. Instead, the sum over all possible alignment paths can be computed. A non-monotonic first-order dependence assumption is made that makes the computation of the forward-backward algorithm tractable [Wang & Alkhouli$^+$ 17, Wang & Zhu$^+$ 18]. These first-order models use the posterior probabilities as true labels and jointly train the alignment and lexicon models batch-wise in EM-style training. This significantly slows down both training and decoding. A simliar work by [Wu & Cotterell 19] uses a first-order model, but monotonic for a character-level transducer that makes both training and decoding faster due to a much smaller vocabulary.

A latent alignment can depend on its predecessors (first-order or higher-order alignment models), as discussed or have no dependency on its history (zero-order) like ours. There are also several recent works in which attention with a zero-order property is used as a latent variable by decomposing the joint distribution. Similar to our work, the model capacity is limited to a zero-order lexicalized alignment model. In [Wu & Shapiro$^+$ 18], the authors use an RNN-based zero-order model with character-level output vocabulary for the transduction task. Similarly, in [Shankar & Garg$^+$ 18], a zero-order latent attention model is explored in machine translation. They run the experiments on small datasets where the vocabularies appear to be small and apply topK approximation to ease computation. While their models are based on recurrent networks, our work is an extension of their paper as we also use the self-attentive zero-order models with larger vocabularies.

Posterior attention distribution is another point of view that calculates a second attention score after predicting the next token [Shankar & Sarawagi 19]. To decrease the computational power needed, they use approximation, but the model leads to negligible improvements on top of the zero-order model. The model proposed by [Kim & Denton$^+$ 17] extends beyond the basic attention structure by seeing it as a graphical model over a set of latent variables to enable partial selection of input positions. In [Deng & Kim$^+$ 18], the authors also propose a non-differentiable approach where they apply hard attention that explicitly uses a single input for each output.

Hard attention [Xu & Ba$^+$ 15] is a discrete variant of soft attention that takes the best source position at each target step; thus, it can be interpreted similarly to a hard alignment. However, the effectiveness of hard attention for offline translation is often limited as it can only represent monotonic alignments, which motivates its application in simultaneous translation [Arivazhagan & Cherry$^+$ 19] and speech recognition [Raffel & Luong$^+$ 17, Lawson & Chiu$^+$ 18b].

6. End-to-End Speech-to-Text Translation

6.1 Introduction

Sequence-to-sequence models can be used for plenty of applications regardless of their inputs and outputs. One of the applications of sequence modeling that has recently received increasing attention is speech-to-text translation or simply speech translation. Previous chapters have discussed the flexibility of sequence-to-sequence modeling and shown experimental results on automatic speech recognition and machine translation tasks. This chapter sheds light on speech translation that takes audio signals of speech as input and outputs written text translation. Moving from the spoken into the written domain requires handling both ASR and MT challenges at the same time, making the whole task even more difficult than the sum of its parts.

As discussed in Section 1.4, in speech translation, we define a speech input observation x_1^T, a source sequence f_1^J, and a target sequence e_1^I. Figure 6.1 shows a general paradigm of the speech translation task. Given Bayes' decision rule in Equation 1.17, the posterior distribution $Pr(e_1^I|x_1^T)$ can be modeled either by explicit use of the source sequence f_1^J as a pivot step stated in Equation 1.22 or by implicitly learning the intermediate representation. The former can be the traditional speech translation system that relies on the coupling between ASR and MT models, the cascade system discussed in Section 1.4.1, and the latter is *direct modeling* in which source speech signals are directly translated to target texts.

The cascade model divides the task into two independent recognition and translation steps. An ASR system trained on speech-to-source data first transcribes a spoken language utterance. The transcribed word sequence is then translated by an MT system trained on source-to-target corpora. Sentence segmentation and punctuation prediction can additionally be integrated into the pipeline. The whole system final performance depends on the accuracy of each individual part and how all the components are integrated across the cascade system. In practice, a nearly zero recognition error cannot be expected. There, the cascade system suffers from an inherent problem known as *error propagation.* Since natural languages involve a high degree of ambiguity, every component produces a certain number of errors which are then propagated through the cascade pipeline. This causes compounding follow-up errors.

A tighter integration of these two models can arguably prevent a number of errors [Casacuberta & Ney$^+$ 04, Matusov & Kanthak$^+$ 05, Casacuberta & Federico$^+$ 08]. It has also been proposed to provide multiple recognition hypotheses to pass on information on recognition uncertainty, thereby avoiding some of the early decisions that occur in the cascade methods. However, an explicit discrete intermediate transcription needs to be first generated in all such systems, thus cascading involves decoding twice.

In contrast, direct modeling has the potential to tackle the error propagation issue of the cascade method. Unlike the cascade systems, each and every component of the model is trained jointly to maximize translation performance, and it eliminates the two-pass decoding property. However, the transformation from source speech inputs to target texts can be a more complex function compared

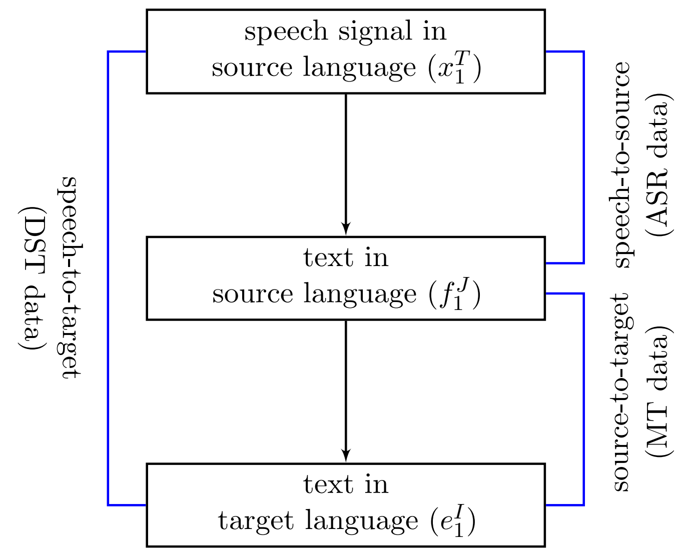

speech signal in source language (x_1^T)
text in source language (f_1^J)
text in target language (e_1^I)
speech-to-target (DST data)
speech-to-source (ASR data)
source-to-target (MT data)

Figure 6.1: An abstract paradigm of speech translation. Assuming a realistic setup, speech translation models can be trained using three types of data; speech-to-source or ASR, source-to-target or MT, and speech-to-target or DST data. The models can rely on explicit use of the source sequence f_1^J as a pivot step or directly translate the sequence of frame-wise speech x_1^T to the sequence of foreign target words e_1^I.

to that of the ASR or MT tasks individually. This might lead to harder optimization or require a moderate amount of paired translated speech-to-target or direct speech translation (DST) data to perform reasonably well. In the DST data, spoken utterances are paired with corresponding target sequences. This is in contrast to the cascade system trained on separate ASR and MT corpora (see Figure 6.1). A realistic data scenario gives us relatively large speech-to-source, ASR data and source-to-target, MT data, but only limited amount of speech-to-target, DST data. The size of common DST corpora is often an order of magnitude smaller than the ASR datasets and several orders of magnitude smaller than the MT ones.

The flexibility of neural modeling and its ability to train all sub-components of a network jointly inspire us to reconsider the traditional cascade system in this chapter and to develop new approaches with the ultimate goal of improving the performance of speech translation systems. To develop neural speech translation models, previous chapters show how to solve simpler tasks, ASR and MT, with pure single and stand-alone neural models. Given that, in this chapter, we exclusively employ neural sequence-to-sequence models to tighten the speech translation pipeline and train models such that all parameters are jointly optimized in part through end-to-end training. In the extreme, we replace the cascade system with the direct model that does not divide the speech translation task into two parts. It does not depend on any supervision of the source text, i.e., transcriptions, and directly translates a speech utterance into a target sequence.

Our first contribution, in Section 6.2, is therefore to study the direct model, which is based on the attention models. As a matter of fact, neural ASR models are extended to output translations instead of transcriptions. Sections 6.2.1 and 6.2.2 cover the CTC loss for faster convergence and data augmentation for reducing the data scarcity problem of direct models. Our next contribution lies in transfer learning as an effective potential solution for exploiting all available data, i.e., ASR, MT, and DST data. We conduct a closer investigation of sequential transfer learning or simply pretraining model parameters and multi-task learning in Sections 6.3.1 and 6.3.2, respectively. We address a better coupling of automatic speech recognition and machine translation models as a pretraining stage and study the effect of transfer learning by initializing different model components. Assuming a realistic setup and not ignoring other available ASR and MT corpora where cascade and direct models are trained on non-equal amounts of data, the direct model performance is often behind the cascade system. Such a scenario puts the direct model at a severe disadvantage in practical situations. Therefore, a new remedy proposed in Section 6.4 that naturally relies on two modeling steps akin to the cascade model but is end-to-end trainable and potentially diminishes the error propagation problem. We experimentally explore the performance of all methods in Section 6.5. We close the chapter with a conclusion and related works in Sections 6.7 and 6.8, respectively.

6.2 Direct Model

An alternative approach to the cascade system is direct modeling, which is both simple and appealing. The idea is to use recent advances in neural networks to directly translate the speech utterance into the target sentence [Berard & Pietquin+ 16]. Mathematically, we directly model

$$p(e_1^I|x_1^T) = \prod_{i=1}^{I} p(e_i|e_0^{i-1}, x_1^T) \tag{6.1}$$

where $T > I$. In fact, any type of neural network architecture can be used to represent the posterior probability. Let us assume the recurrent attention-based network as introduced before. Given L_e layers of bidirectional LSTM, max-pooling operations are interleaved at multiple steps on the time dimension to reduce the number of frames. Formally, we write

$$h_t = \text{BiLSTM}^{(L_e)} \circ \ldots \circ \text{max-pool}^{(1)} \circ \text{BiLSTM}^{(1)}(\tilde{x}_t, h_{t-1}, h_{t+1}). \tag{6.2}$$

where the obtained sequence of input states is shorter than the original one by a time reduction factor. For the sake of simplicity, we use time index t in both cases. Here, $\tilde{x}_t$ represents the frame-wise feature vector at time step t, and h_t represents the encoder state as before. Then, the LSTM decoder generates the target sequence conditioned on the encoder representations. While computing the target word e_i at each time step, an additive attention function (with or without fertility) is used to obtain the context vector as

$$c_i = \text{attention}(s_{i-1}, h_t). \tag{6.3}$$

The decoder appends the context vector to the LSTM state and the previously generated target token for the final prediction given by

$$s_i = \text{LSTM}\left(s_{i-1}, \tilde{e}_i, c_i\right), \tag{6.4}$$

$$p(e_i|e_0^{i-1}, h_1^T) = \text{softmax} \circ \text{linear} \circ \text{maxout}\left(s_{i-1}, \tilde{e}_{i-1}, c_i\right). \tag{6.5}$$

Figure 6.2 shows the transformation from the source speech to the target sentence. The direct model has some appealing advantages compared to the cascade model. It avoids early decisions and thus is not subject to the error propagation problem. It enables joint optimization of all model components and removes the need for explicit intermediate representations. Therefore, it results in faster decoding and less computational complexity in total. Generally speaking, avoiding the decomposition of the target sequence posterior in Equation 1.19 can prevent the aforementioned issues. In addition, direct modeling allows the exploitation of speech-to-target training data that might be available in some data conditions but cannot be used in building a cascade system directly. However, it requires a moderate amount of paired translated speech-to-target data, which is not easy to acquire. Despite these properties, the independence between the target sentence and the speech utterance in the cascade model might be a bad assumption because prosody can contain key information for a correct translation. The direct model makes this information accessible for output translation. Throughout this chapter, the direct method is based on the RNN attention model unless otherwise specified.

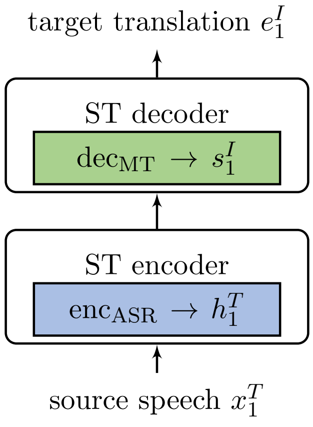

target translation e_1^I
ST decoder
$\text{dec}_{\text{MT}} \rightarrow s_1^I$
ST encoder
$\text{enc}_{\text{ASR}} \rightarrow h_1^T$
source speech x_1^T

Figure 6.2: The direct speech translation model based on the encoder-decoder architecture uses no source transcripts. The model directly translates the sequence of frame-wise speech input x_1^T to the sequence of target tokens e_1^I. The blue and green blocks show parameters which can be initialized by a pretrained ASR encoder and a pretrained MT decoder.

6.2.1 CTC Loss

As stated earlier, the attention mechanism is flexible for different word reordering as it freely pays attention to any input position. But such a non-sequential alignment is not always desired, especially for applications with a monotonic input-output relation. Similar to attention-based approaches, the CTC model is trained in an end-to-end fashion where it uses the Markov assumption and dynamic programming [Graves & Fernández+ 06].

In contrast to the attention model, CTC makes a conditional independence assumption to obtain the transcription label sequence probabilities. To benefit from the two models, one can combine them as joint CTC-attention end-to-end ASR [Watanabe & Hori+ 17] to improve robustness, yield faster initial convergence in rescoring, training [Kim & Hori+ 17, Zeyer & Irie+ 18], or one-pass decoding [Hori & Watanabe+ 17]. It is shown that the multi-objective learning of a hybrid of CTC and attention models is effective where an auxiliary CTC loss helps convergence in the training of ASR attention models.

While it is commonly believed that CTC cannot help speech translation since the monotonic alignment of speech frames and target sentences is no longer required in translation, there is no reason to assume that it cannot be done in the speech part of the network. Therefore, we still apply the same idea to force the ASR alignments to be monotonic. The forward-backward algorithm of CTC encourages monotonic alignment between the speech frames and source label sequences. Precisely speaking, the CTC loss is used on top of the speech encoder of the attention model only during training.

The idea of CTC is to monotonically map a frame-wise speech sequence to a label sequence, which is usually shorter. Assuming the source label sequence of f_1^J, CTC defines the frame-wise source label sequence $z_1^T = \left\{ z_t \in V_f \cup \text{<}b\text{>} \mid t = 1, \ldots, T \right\}$ with an additional blank symbol <b>. The blank symbol solves the problem of unknown segmentation of the input sequence and handles repetitions. It can appear at any time (between and within words) and represents the fact that no token is recognized at the current time step. Therefore, the augmented label sequence can have a size of $2J + 1$. The blank state differs from silence and is modeled as a token in its own right; thus, the final classification involves a vocabulary size of $|V_f| + 1$.

The posterior distribution is decomposed under the conditional independence assumption and given by

$$p(f_1^J|x_1^T) \approx \underbrace{\sum_{z_1^T} \prod_{t=1}^{T} p(z_t|z_{t-1}, f_1^J) p(z_t|x_1^T)\, p(f_1^J)}_{\triangleq\, p_{ctc}}. \tag{6.6}$$

Equation 6.6 includes three quantities: the transition probability $p(z_t|z_{t-1}, f_1^J)$, assumed to be a uniform distribution, the frame-wise posterior distribution $p(z_t|x_1^T)$, and the language model $p(f_1^J)$ as a constant. The sum over all possible z_1^T is efficiently computed using dynamic programming. The frame-wise posterior distribution is conditioned on the input sequence and modeled by

$$p(z_t|x_1^T) = \text{softmax} \circ \text{linear}(h_t) \tag{6.7}$$

where h_t is the output of bidirectional LSTM as written in Equation 6.2.

During training, the ST decoder predicts the frame-wise posterior distribution of the target sequence e_1^I, while the CTC function predicts the posterior distribution of f_1^J given the corresponding input, referred to p_{s2s} and p_{ctc}, respectively. We simply use the sum of their log-likelihood values with equal contributions as

$$\mathcal{L} = \log p_{s2s}(e_1^I|x_1^T) + \log p_{ctc}(f_1^J|x_1^T). \tag{6.8}$$

This combination shares the same encoder between the CTC and the attention networks.

6.2.2 Spectrogram Augmentation

Since speech translation is usually a low-resource task, we explore the effect of the data augmentation technique known as SpecAugment, which is implemented slightly differently from the main proposed approach [Park & Chan$^+$ 19]. In speech translation experiments, we apply SpecAugment on the MFCC feature, which is different from the original paper, applied it on the log mel spectrum of the input audio. In either case, the technique can be seen as feature dropout.

We randomly apply masking in consecutive frames on the time axis as well as successive dimensions on the feature axis. Both maskings are restricted to a fixed maximum size. As a minor effect of time warping is reported in [Park & Chan$^+$ 19], we do not apply it in this work.

Time Masking

Δt successive time steps $[t, t + \Delta t)$ are masked by setting them to zero, $(x_t, \ldots, x_{t+\Delta t}) = 0$. Δt is the masking window selected from a uniform distribution from 0 to a predefined maximum time mask parameter, i.e., in an interval of $[0, \Delta t_{max}]$. The time position t is also randomly selected from another uniform distribution between $[0, T)$. We choose the time position differently from the original paper where they select t in the interval of $[0, T - \Delta t)$. We never exceed the maximum sequence length T. This means that if $t + \Delta t > T$, we set it to T.

Time masking is applied $m_t \in \mathbb{N}$ times where m_t is randomly selected from $[1, M_t]$ with a predefined maximum iteration number M_t. If $m_t > 1$, the same time position t is not selected more than once, i.e., without replacement[1].

[1]It is not clear whether the original paper allows replacement or not.

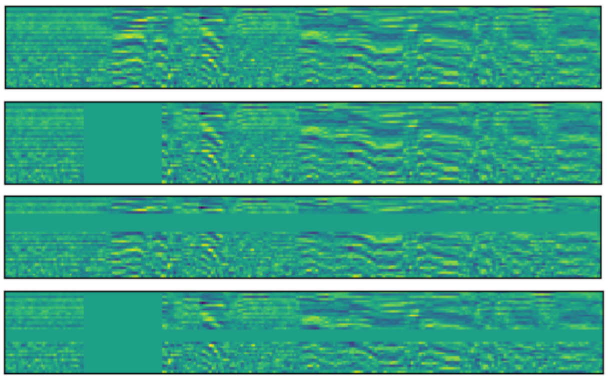


Figure 6.3: From top to bottom, the figure indicates the spectrogram of the input with no augmentation, time masking, frequency masking, and both maskings used.

Frequency Masking

Frequency masking can additionally be employed such that Δd consecutive frequency channels $[d, d + \Delta d)$ are masked, where Δd is chosen from a uniform distribution from 0 to a predefined maximum frequency mask parameter Δd_{max}, and d is randomly selected from $[0, D)$. Again, we note the difference between our implementation and the original paper, where the selection interval is $[0, D - \Delta d)$. D represents the input feature dimension, i.e., the number of MFCC features, which is 80 in our experiments. Similar to time masking, we do not allow for an already selected frequency index d and check if $d + \Delta d > D$, in which case we set it to D. This procedure is again repeated $m_d \in \mathbb{N}$ times, where m_d is randomly selected from $[1, M_d]$ with $M_d \in \mathbb{N}$ referring to the number of times we apply the frequency masks. As we standardize the features to a mean of zero and a variance of one, masking to zero is equivalent to setting it to the mean value. SpecAugment is only applied during training. Figure 6.3 shows examples of the individual augmentations applied to an input. Multiple frequency and time masks might overlap.

6.3 Transfer Learning from Speech Recognition and Text Translation

Supervised learning will break down when there is not enough training data for the desired task to train a reliable model. Transfer learning refers to a general strategy in which we study how models can be transferred and adapted to another set of training data on which they have not been trained so far [Pan & Yang 10, Ruder 19]. The transfer can be across different domains, languages, and even tasks. Leveraging the data of similar tasks or domains helps us to handle low-resource data conditions. In principle, the aim is to transfer as much knowledge as possible. One can broadly categorize transfer learning into transductive and inductive.

In transductive transfer learning, the source and target tasks are the same. Domain adaptation and cross-lingual learning are its two main applications. In contrast, inductive transfer learning represents a setting where the source and target tasks are different. *Sequential transfer learning* (tasks are learned sequentially) and joint learning, also known as *multi-task learning* (tasks are learned simultaneously), are grouped in this category.

Speech translation can be considered a low-resource scenario compared to machine translation and automatic speech recognition. Here, inductive transfer learning can be used to leverage more labeled data. This section leaves the transductive methods out of its scope and discusses the two approaches mentioned above in the following sections.

6.3.1 Sequential Transfer Learning

The most frequently used transfer learning technique is sequential, where the training of sereval tasks using different data is carried out in sequence [Wang & Zheng 15]. The main idea is to transfer information from the source task to the target one. Hence, sequential transfer learning has two phases: *pretraining* and *fine-tuning*, which are performed separately. In the pretraining phase, the model is trained on the source task. Then the knowledge of it is transferred to the target task in fine-tuning. The parameters of the pretrained model are used, and learned representations can be utilized in the second task.

Usually, the pretraining phase is trained using more training data compared to the fine-tuning step. This sequential setting requires more computational time in total than that of multi-task learning with joint optimization. However, the pretraining stage only needs to be performed once, and the fine-tuning phase is often efficient in practice. Sequential transfer learning has been shown to be useful in three main scenarios [Ruder 19]:

1. There is no available data for all tasks at the same time.
2. The target task has a low-resource data condition compared to the source task.
3. Adaptation to the target task is required.

Making use of a sufficient amount of training data on the source tasks is the main reason to benefit from pretraining. Selecting a suitable pretraining task is important in facilitating learning representation for the target task. ASR and MT tasks are inherently the most obvious natural source tasks for an integrated speech translation pipeline. However, one could argue that an easier way to benefit from sequential transfer learning is to take as much unlabeled data as possible because obtaining it does not require labels. We believe the current state-of-the-art speech translation models work ineffectively without any weakly or fully supervised data, as some initial results are reported in [Chung & Weng$^+$ 19]. Therefore, that direction of research is left out of the scope of this work.

Speech translation model parameters can be initialized by ASR and MT parameters already trained on a larger amount of ASR and MT data. The common method is to use the encoder parameters of an ASR and the decoder parameters of an MT model to initialize an ST model's encoder and decoder, respectively [Bahar & Bieschke$^+$ 19]. As shown in Figure 6.2, the pretrained ASR encoder representation (the blue block) and the pretrained MT decoder representation (the green block) are transferred to the ST encoder and decoder. According to the literature, pretraining helps learning convergence and final translation quality by choosing a better starting point for training than random initialization [Bansal & Kamper$^+$ 18, Bansal & Kamper$^+$ 19, Stoian & Bansal$^+$ 20]. It is also shown that the ASR model itself can be used to initialize all parameters of the ST model, where instead of producing transcripts, the model is fine-tuned to generate translations [Bansal & Kamper$^+$ 19, Stoian & Bansal$^+$ 20]. We verify some of these findings in the experimental section and extend the pretraining scheme with the adaptation of components.

Adaptation of Components When using ASR and MT models as the source tasks in pretraining, the ASR encoder contains speech information, whereas the MT decoder expects to get text-related information. In order to familiarize the pretrained text decoder with the output of the pretrained speech encoder, we insert an additional *adapter* layer between the encoder and decoder of the ST model as shown in Figure 6.4. This is done to smooth the connection between these two representations that have been trained individually. The adapter is used as an extra degree of freedom by which the network can reconcile the output of the speech encoder into the input of the text decoder. The adapter layer can be any neural network and is applicable to all architectures.

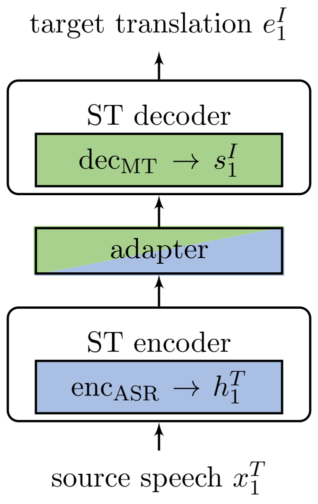

target translation e_1^I
ST decoder
$\text{dec}_{\text{MT}} \rightarrow s_1^I$
adapter
ST encoder
$\text{enc}_{\text{ASR}} \rightarrow h_1^T$
source speech x_1^T

Figure 6.4: The adapter layer between the two pretrained components.

We train it jointly without freezing the parameters in fine-tuning. This contrasts with [Kim & Petrov$^+$ 19], where a linear mapping between encoder and decoder representations is carried out.

6.3.2 Multi-Task Learning

Typical machine learning methods involve training a single task; however, we ignore information that other related tasks can give. Multi-task learning has become an important technique to improve the generalization performance of a task using other related ones by leveraging the domain-specific information, exploiting additional training data, or learning features that are not easy to be learned using just the main task [Caruana 93, Ruder 19]. In the multi-task scheme, one or several auxiliary tasks are employed to co-train and improve the performance of the main task while sharing parts of their parameters. An auxiliary task provides an inductive bias for the benefit of more tasks, thus improving generalization. Therefore, the optimization problem deals with more than one loss function, i.e., the prediction of multiple tasks at once. Unlike sequential learning, the models are optimized jointly, not separately. We need to compromise between multiple tasks. Although an inductive bias seems to work intuitively, in practice, it is important firstly to find suitable auxiliary tasks and secondly to determine how they interact with the main task.

Selecting an auxiliary task depends mainly on its similarity to the main task, i.e., having the same inductive bias [Caruana 98, Baxter 00]. This argument then allows tasks where different resources collect data for the same prediction problem. There are two main ways to combine these components into a complete sequence-to-sequence model, namely *one-to-two* and *two-to-one* [Luong & Le$^+$ 16, Weiss & Chorowski$^+$ 17][2]. In the context of speech translation, we determine four components: a source speech encoder (ASR encoder), a source text encoder (MT encoder), a source text decoder (ASR decoder), and a target text decoder (MT decoder). Generally speaking, the first and fourth components can be identical to the ST encoder and decoder, respectively. We do not distinguish the attention component as it is considered to be a layer in the decoder. Like the choice of an auxiliary task in sequential learning, ASR and MT tasks are inherently the most closely related co-trainers coming with a moderate amount of training data.

It is important to highlight that multi-task learning is an end-to-end approach that can be used with the purpose of co-training the main task by sharing relevant information as well as utilizing

[2]Overall, there are four possibilities for combining them. In this thesis, we ignore auto-encoder as a one-to-one approach requiring monolingual data and the two-to-two option.

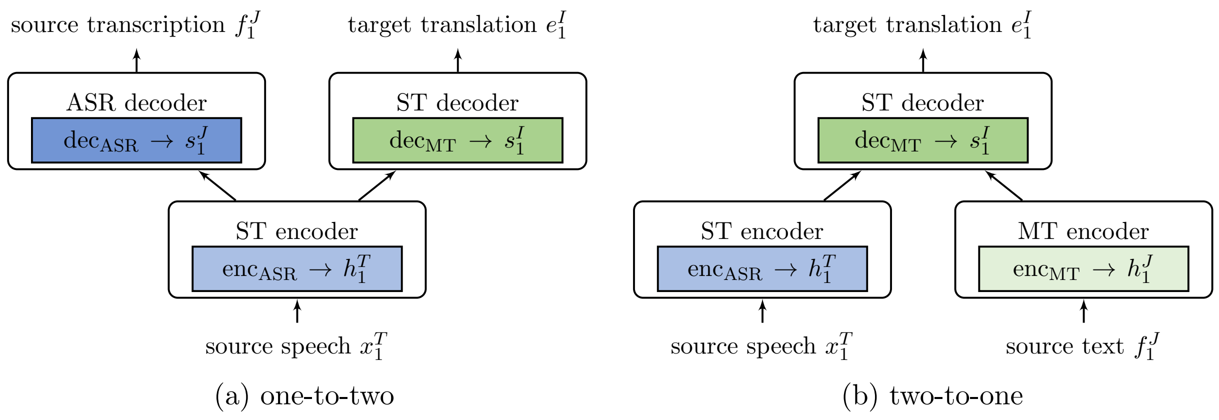

source transcription f_1^J
target translation e_1^I
ASR decoder
$\mathrm{dec}_{\mathrm{ASR}} \rightarrow s_1^J$
ST decoder
$\mathrm{dec}_{\mathrm{MT}} \rightarrow s_1^I$
ST encoder
$\mathrm{enc}_{\mathrm{ASR}} \rightarrow h_1^T$
source speech x_1^T
(a) one-to-two
target translation e_1^I
ST decoder
$\mathrm{dec}_{\mathrm{MT}} \rightarrow s_1^I$
ST encoder
$\mathrm{enc}_{\mathrm{ASR}} \rightarrow h_1^T$
MT encoder
$\mathrm{enc}_{\mathrm{MT}} \rightarrow h_1^J$
source speech x_1^T
source text f_1^J
(b) two-to-one

Figure 6.5: Multi-task end-to-end speech translation. The colored blocks correspond to the components in pretraining. The blue and green blocks resemble ASR and MT pretrained modules. a) The ASR decoder is removed during decoding, resulting in the direct model. b) The MT encoder is removed during decoding, resulting in the direct model.

additional data. By sharing parts of the network, ST might benefit from the relevant knowledge obtained through the training of ASR, MT, or both tasks [Weiss & Chorowski$^+$ 17, Berard & Besacier$^+$ 18]. In addition to the point above, we aim to use multi-task learning to also leverage additional ASR and MT data. Multi-task learning can also be combined with CTC loss. However, it seems to be a good practice only to apply one of them [Bahar & Bieschke$^+$ 19].

One-to-Two

As shown in Figure 6.5(a), in the one-to-two setup, the ASR model is used as the auxiliary co-trainer in which a speech encoder is shared between both tasks, while two independent decoders correspond to transcription and translation texts. To make matters precise, let us assume a speech encoder that encodes a sequence of speech frames x_1^T with internal representation h_t at position t as expressed in Equation 6.2. Taking the RNN-based network, we specify two sets of equations that correspond to the ASR and ST decoders. Therefore, we have

$$\begin{aligned} c_j &= \mathrm{attention}(s_{j-1}, h_t), && (6.9)\\ s_j &= \mathrm{LSTM}(s_{j-1}, \tilde{f}_j, c_j), && (6.10)\\ p(f_j|f_0^{j-1}, x_1^T) &= \mathrm{softmax} \circ \mathrm{linear} \circ \mathrm{maxout}(s_{j-1}, \tilde{f}_{j-1}, c_j), && (6.11) \end{aligned}$$

and

$$\begin{aligned} c_i &= \mathrm{attention}(s_{i-1}, h_t), && (6.12)\\ s_i &= \mathrm{LSTM}(s_{i-1}, \tilde{e}_i, c_i), && (6.13)\\ p(e_i|e_0^{i-1}, x_1^T) &= \mathrm{softmax} \circ \mathrm{linear} \circ \mathrm{maxout}(s_{i-1}, \tilde{e}_{i-1}, c_i). && (6.14) \end{aligned}$$

In these equations, indices j and i refer to source and target word positions, respectively. The error is therefore back-propagated via two decoders into the input, and the final loss is computed as a weighted sum of the two losses given by

$$\mathcal{L} = \lambda \log p(e_1^I|x_1^T) + (1-\lambda) \log p(f_1^J|x_1^T). \qquad (6.15)$$

Here, $0 < \lambda < 1$ is the ST weight loss reducing the number of its updates.

There are two ways to train such a one-to-two network. The first option is similar to the approach described by [Weiss & Chorowski+ 17] in which we only use triple data, including the source speech, source text, and target text. The encoder contains speech frames encoded. Each decoder attends to the encoder states and generates a sequence of tokens in either the source or target language. In this case, the data needs to be parallel, and no additional ASR data can be exploited; however, the ST task can learn from its ASR co-trainer. Another way is to use more ASR data where both ASR and ST models are trained in parallel but on different speech input data. In this case, each batch consists of pairs of speech-to-source and speech-to-target data. Both models run on their own respective training data and only share the encoder parameters (see Figure 6.5(a)). During decoding, the ASR part of the network is discarded, again resulting in a simple direct model.

Two-to-One

A text decoder is shared in the two-to-one architecture to generate a target translation by attending to two independent encoders, a speech and a text encoder. Here, MT is co-trained along with the ST model (see Figure 6.5(b)). Assuming bidirectional LSTM layers, we state

$$h_j = \text{BiLSTM}^{(L_e)} \circ \ldots \circ \text{BiLSTM}^{(1)}(\tilde{f}_j, h_{j-1}, h_{j+1}), \tag{6.16}$$

$$h_t = \text{BiLSTM}^{(L_e)} \circ \ldots \circ \text{max-pool}^{(1)} \circ \text{BiLSTM}^{(1)}(\tilde{x}_t, h_{t-1}, h_{t+1}) \tag{6.17}$$

with h_j and h_t being the text and speech encoders. They do not necessarily have the same number of layers. The text decoder interchangeably selects one of the encoders to collaborate on both text and speech inputs. Depending on whether the decoder is used for speech or text translation, it uses one of the attention components (either c_i^{MT} or c_i^{ST}) to attend to the respective encoder. This means

$$c_i^{\text{MT}} = \text{attention}(s_{i-1}, h_j), \tag{6.18}$$

$$c_i^{\text{ST}} = \text{attention}(s_{i-1}, h_t). \tag{6.19}$$

During training, each batch contains both pairs of speech-to-target and source-to-target data. Whether the source speech or source text is present, a switching layer decides which corresponding context vectors are fed to the decoder. The text translation part of the model is ignored during decoding, resulting in a simple direct model.

The ST training data is typically several orders of magnitude smaller than the available MT data. To avoid the domination of one task with more data, every batch contains a specific percentage of the ST and MT data. The fact that the losses are combined by summing over the entire batch can also interfere in that the losses are combined by a weight batch-wise. Formally, we have

$$\mathcal{L} = \lambda \log p(e_1^I | x_1^T) + (1 - \lambda) \log p(e_1^I | f_1^J). \tag{6.20}$$

where $0 < \lambda < 1$ is the ST weight loss that determines the number of its updates.

Generally speaking, multi-task and sequential transfer learning can complement each other. In this case, the multi-task network components can be initialized by the pretrained models trained on the selected auxiliary tasks, as will be discussed in the experimental results later. The colored blocks in Figure 6.5 also represent the complementarity of pretraining modules in the multi-task scenarios. The blue and green colors correspond to the ASR and MT pretraining units.

6.4 Tight Integration of Cascade Model

Cascaded speech translation systems rely on discrete transcriptions. The transcription generated by the ASR model provides supervision signals from the source side and helps the transformation between the source speech and target text. However, they are not differentiable. Two-step models [Anastasopoulos & Chiang 18, Sperber & Neubig$^+$ 19a] are approaches to using an intermediate representation and preserving end-to-end trainability of speech translation models. Such models pass the recognizer's decoder hidden states or attention context vectors into the MT decoder while ignoring the MT encoder completely.

This section investigates the feasibility of collapsing the entire cascade components into a single end-to-end trainable model where all ASR and MT model parameters are jointly optimized without ignoring any learned parameter, for instance, the MT encoder.

While many recent works investigate direct end-to-end approaches, this part of this thesis proposes a tight integration of ASR and MT modules which is closer to the cascade method. Still, it maintains the end-to-end trainability of the overall architecture. The method is effective yet simple to apply because we do not modify the architecture or training criterion. It also allows us to exploit speech-to-target training data that might be available in some situations but cannot be used for the training of the cascade model. Before the era of neural models, such a tight integration was done using word lattices or word confusion networks [Mangu & Brill$^+$ 00, Bertoldi 05, Bertoldi & Federico 05, Matusov & Kanthak$^+$ 05, Bertoldi & Zens$^+$ 07, Matusov & Ney$^+$ 05, Matusov & Hoffmeister$^+$ 08] at the interface between ASR and MT. This work revisits a similar idea of confusion networks by passing on renormalized source word posteriors from ASR to MT. The core steps of the method that we call *tight integration cascade* are as follows.

Figure 6.6 illustrates an overview of the tight integration cascade model. We train the ASR model with speech-to-source data and the MT model with bilingual translation data and then concatenate them as used in the cascade setup. But instead of providing the 1-best output of the most probable word sequence to the translation system, the posterior probabilities $p(f_j|f_0^{j-1}, x_1^T)$ are used at every position.

These posteriors may simulate a degree of the ambiguities of the ASR candidates with probabilities, thus conveying more information with higher uncertainty. In other words, we pass on the posterior distribution as a soft decision instead of the one-hot vector of the most probable word that requires a hard decision. This process differs from the non-differentiable property of argmax. Since the MT model has been trained on the one-hot representations, it likely performs worse on smooth probability distributions. Therefore, the word distributions are sharpened by an exponent γ at each position and then renormalized the probabilities, which are rewritten by

$$p(f_j|f_0^{j-1}, x_1^T) = \frac{p^\gamma(f_j|f_0^{j-1}, x_1^T)}{\sum_{f' \in V_f} p^\gamma(f'_j|f_0^{j-1}, x_1^T)} \tag{6.21}$$

where V_f is the source vocabulary.

Next is the fine-tuning stage, where training is continued with speech-to-target parallel data. If we skip this step and perform translation directly, it corresponds to a cascade model.

In training, $\gamma = 1$ is used to simulate the degree of ambiguities, whereas in decoding, we vary γ to obtain the best translation performance. $\gamma = 0$ leads to a uniform distribution over the vocabulary. If $\gamma = 1$, we get the actual posterior distribution, and for $\gamma > 1$, the highest probability word is emphasized, resulting in almost one-hot representation if γ is chosen large enough.

Since this sharpening emphasizes the most probable word of the vocabulary in each distribution regardless of the actual decision that has been originally made in the ASR decoder, it is possible that in some cases, the wrong words are picked due to reranking the words. This is because beam search might choose sentences that have the highest final probability but may have lower probabilities during intermediate steps.

In practice, the source transcriptions are often provided in speech translation. Therefore, we use the true transcriptions during training to compute the ASR loss and apply beam search to generate the best ASR hypothesis during decoding and trace back the decisions at each position

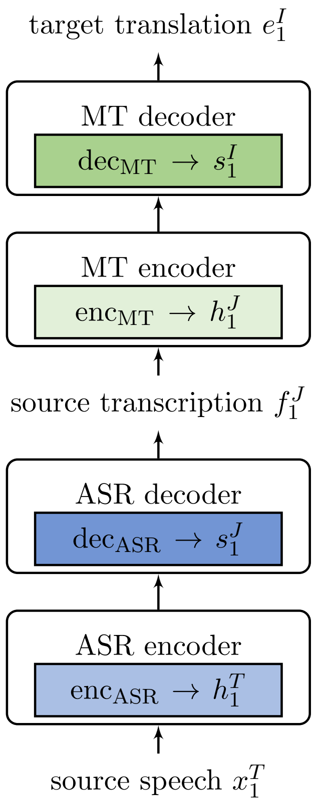

target translation e_1^I
MT decoder
$\mathrm{dec}_{\mathrm{MT}} \to s_1^I$
MT encoder
$\mathrm{enc}_{\mathrm{MT}} \to h_1^J$
source transcription f_1^J
ASR decoder
$\mathrm{dec}_{\mathrm{ASR}} \to s_1^J$
ASR encoder
$\mathrm{enc}_{\mathrm{ASR}} \to h_1^T$
source speech x_1^T

Figure 6.6: The tight integration cascade model. All cascade model parameters are kept. Here, the source transcription can be chosen from the best candidate (the standard cascade model) or from the word posteriors (the tight integration cascade model).

after the best sequence has been chosen. The sharpening is applied to the sequence of posterior probabilities once the beam search has finished while tracing back over the time steps. Then, the respective posterior probability distributions are passed to the MT model at each position. We believe the tight integration cascade model offers five main advantages compared to other models:

- It assumes a more realistic data condition and employs all types of training data, i.e., ASR, MT, and DST corpora. This differs from the data conditions of both cascade and direct models.
- The end-to-end methods either conduct translation without transcribing or suffer from inconsistency between transcriptions and translations [Sperber & Setiawan$^+$ 20]. Some examples that stand with inconsistency are multilingual systems [Inaguma & Duh$^+$ 19, Gangi & Negri$^+$ 19d] and multi-task learning [Weiss & Chorowski$^+$ 17]. Transcriptions are essential in many applications and might need to be displayed together with translations to users. The tight integration cascade model provides both transcription and translation with a strong consistency between them. This consistency is an inherent property of the model, as the generation of translation depends on transcription from the previous layers of the network.
- In comparison to the two-step models, it works on the probability level instead of the hidden representation, which does not represent the word that has to be generated, rather a mix or a weighted representation of all words in the vocabulary. The representation of a discrete target word as a real-valued vector where its dimension is usually much smaller than the vocabulary size ($d_h \ll |V_f|$) can be a blurry representation to determine the correct word.

- It benefits from all components of pretrained models. In contrast to two-step and direct models, we keep all parameters of submodels in the fine-tuning phase. Therefore, it is probably guaranteed to be at least on par with the cascade system.
- This model makes better use of ASR and MT data in comparison to multi-task learning. The main task in multi-task learning (Section 6.3.2) shares half of its parameters with an auxiliary task, either the encoder with the MT or the decoder with the ASR task. Additional data thus only have a rather indirect impact on the main task. Contrary to that, in tight integration cascade, all parameters of auxiliary tasks are involved in the training of the main task. Hence, they have a more direct impact that leads to potentially better data efficiency.

Source Language Model Combination Analogously to speech recognition, we integrate a language model score with that of the acoustic model in a log-linear combination. This is a simple way to include a source language model in decoding. Therefore, Equation 6.21 is rewritten as

$$p(f_j|f_0^{j-1}, x_1^T) = \frac{\left[p_{\text{AM}}(f_j|f_0^{j-1}, x_1^T)p_{\text{LM}}^{\lambda}(f_j|f_0^{j-1})\right]^{\gamma}}{\sum_{f' \in V_f}\left[p_{\text{AM}}(f'_j|f_0^{j-1}, x_1^T)p_{\text{LM}}^{\lambda}(f'_j|f_0^{j-1})\right]^{\gamma}} \tag{6.22}$$

with λ being the language model scale. For $\gamma = 1$, the formula represents the shallow fusion technique.

6.5 Experimental Results

This section includes the results of speech translation neural sequence-to-sequence models. We carry out all experiments using RETURNN with an integrated interface to RASR, similar to the ASR experiments.

6.5.1 Setups

The experiments are conducted on IWSLT 2018 TED talks English→German due to its long history in internal and external evaluations. After ablation experiments mainly on this task, we determine the best models on MuST-C English→Spanish and LibriSpeech audio books English→French commonly used as a benchmark task for speech translation. The data statistics are given in Appendices A.3.1, A.3.2, and A.3.3, respectively. We use ASR, MT, and DST datasets except for LibriSpeech, where we assume a limited data condition and only use the available DST corpus. Since the DST corpus often comes as a triple dataset, we always include it in ASR and MT model training. Besides that, we choose some of the largest publicly available ASR and MT datasets to have the ASR and MT models in an unconstrained-like setting and make the comparison between the end-to-end and cascade models more realistic.

For all tasks, 80-dimensional MFCC features are extracted with overlapping windows of 25 milliseconds. A small number of utterances longer than 6000 frames is excluded from the training data to avoid running out of memory. Casing and punctuation are kept on the target side. The ASR evaluations are always lower-cased and without punctuation computed by `sclite`. Unless explicitly stated, the models are trained under the same data condition for a fairer comparison. However, the way that the data are used can be different from one technique to another.

IWSLT 2018 TED English→German To train ASR models, a total of 390 hours of transcribed speech is used [Matusov & Wilken+ 18]. More details on the data statistics and preprocessing are given in Appendix A.3.1. The DST data statistics are given in Table A.4. A part of our segments is randomly selected as our cross-validation set, and `dev2010` and `test2015` are chosen to be the development and unseen test sets, respectively. The training data is divided into 4 subepochs, and we select our checkpoints based on the development set. Either 10k or 20k BPE subword merge symbols are used such that the latter case is also used on the target side of the MT data.

The ASR recurrent attention model is adapted from the Switchboard configuration with the fertility concept. The embedding size is slightly larger, with 620 dimensions. A lower initial learning rate of 0.0008 and a decay factor of 0.8 are used. Maximum sequence length is limited to 75 subwords during training. We also use CTC loss on top of the speech encoder and clip the audio features to be between -3 and 3 before applying SpecAugment. As stated before, max-pooling between bidirectional LSTM layers reduces the number of audio frames. The RNN attention model is used not only for building the cascade system but also for other approaches mentioned in the previous sections. Besides the RNN attention model, we use a hybrid HMM acoustic model as the baseline for the speech recognition part of the cascade system taken from [Matusov & Wilken$^+$ 18].

The recognition results can be found in Table B.2 in Appendix. According to these results, a carefully trained attention model can substitute the traditional hybrid HMM model. As hybrid HMM models are out of the scope of this work, this chapter continues only with the attention-based ASR models in the rest of the experiments, unless stated otherwise.

Given 38M bilingual sentence pairs, we train the MT RNN attention model to use in the cascade system and other methods requiring a pretrained MT model. The setup configuration is almost identical to that of machine translation described in Section 3.8.1. However, we use a larger embedding size of 620 nodes, a batch size of 4k without gradient accumulation, and a maximum sequence length of 75 tokens in training. 20k merge BPE subword symbols are used.

The direct model follows the structure of the ASR encoder and MT decoder to enable sequential transfer learning. It has 6 layers of bidirectional LSTM of size 1k with interleaved max-pooling in between that are iteratively trained via the layer-wise construction scheme. Its decoder is a shallow 1-layer LSTM of size 1k. It uses the same target vocabulary as the MT model. The training parameters are mainly based on the ASR setup.

MuST-C English→Spanish Considering a more realistic data condition, we use more auxiliary ASR and MT training data than the end-to-end data on MuST-C English→Spanish. Following [Bahar & Bieschke$^+$ 21], a total of approximately 2300 hours of transcribed speech is employed to train the speech recognition system (see Appendix A.3.2 and Table A.5 for more details.)

We follow the same end-to-end attention models and training setup as those of the IWSLT task. The vocabularies are obtained by 5k subword BPE symbols in English (both ASR and MT) and 32k in Spanish. Due to the larger training data, the epoch is divided by a factor of 20. In the experiments, MuST-C English→Spanish is considered as a development task where we also show results on self-attentive models. However, the recurrent networks are the main architecture of speech translation models in this work.

The self-attentive model is equipped with 6 layers of size 512 in both the encoder and decoder. The ReLu hidden dimension is chosen to be 2048. We have no CTC loss, add positional encoding to the output embedding, and wait longer for more epochs. Instead of absolute, the relative positional encoding is included, which does not really help the final performance.

For this task, two language models are also trained: one LSTM with 4 layers of size 2048 each and one 24-layer 8-head self-attentive LM with 1024 nodes each. The details of the LM data (including 2.1B tokens) are discussed in Appendix A.3.2. The language model scores are then integrated with those of the acoustic attention model through shallow fusion. Table B.4 indicates that the deep self-attentive model has lower perplexity. The use of the external language models as presented in this table directly affects the performance of the ASR models.

We also train both the RNN and self-attentive MT models using 47.7M sentence pairs that are split into 40 subepochs for training updates. Data statistics can be found in Appendix A.3.2. The RNN attention model is configured similarly to that of the MT experiments with no dropout applied. The self-attentive model has the same configuration as the big transformer network in [Vaswani & Shazeer$^+$ 17], composed of 6 layers, 16 heads, and a model size of 1024. Again, the direct model follows the structure of the ASR encoder and MT decoder for each architecture.

LibriSpeech English→French As shown in Table A.6, we employ 100h of clean speech corresponding to 47.2k segments. This amount of training data is also utilized for ASR and MT training. The data condition is not realistic as no additional ASR or MT corpora are taken. The reason for this is firstly that we tend to keep this task as a benchmark task for comparison with other research groups, and secondly that we wish to keep the data condition low-resource for further analyses. 5k joint BPE units are applied on both English and French texts. The same source BPE units are used on the ASR data, hence the source vocabulary is shared between ASR and MT models. We follow the same attention models and training setup as those of the IWSLT English→German task, but a dropout rate of 0.4 is exercised. Evaluation is case-insensitive computed by the Moses `multi-blue.pl` script on tokenized references to be comparable with other works in the literature.

6.5.2 Results

Cascade Models

This section shows the results of the cascade systems as the baselines.

Source Text Processing As discussed before, coupling the ASR and MT models in the cascade pipeline is a well-known challenge in speech translation. As a general rule, recognition and translation qualities are highly correlated [Ruiz & Gangi$^+$ 17]. If the recognizer outputs low-quality sentences with ASR noise, the translation quality significantly drops due to sensitivity to both natural and synthetic noise [Belinkov & Bisk 18].

One method of increasing the robustness of translation models against recognition errors is introducing artificial noise to the source side of the MT training data [Sperber & Niehues$^+$ 17] or to train the MT model on both clean and noisy ASR transcripts collectively [Gangi & Enyedi$^+$ 19]. In practice, the recognizer's output is not only noisy in the choice of words but also contains no punctuation, and it is case-insensitive. One way is to augment the source transcriptions with punctuation. However, this approach is exposed to error compounding and requires a separate punctuation predictor [Peitz & Wiesler$^+$ 12].

Instead, the source side of the text translation system can be adapted to look like the output of the speech recognizer. The MT model can be trained on a punctuation-free source text to resemble the test condition while keeping all punctuation on the target side. Following these strategies, we experimentally explore two options on the English side of the MT data in comparison to the *standard processing* in which we tokenize punctuation and lowercase the English text.

The first technique is done with the goal of converting the source text into a speech transcript similar to the one produced by the ASR system. We add some level of asr-like noise according to the most common recognition errors. After tokenization and lower-casing, all punctuation marks and spliced back contractions (e.g., `do n't` → `don't`) are ignored. All entities expressed with digits (e.g., numbers, dates, etc.) are mapped to their spoken form. Abbreviations of measurement units (e.g., `km` → `kilo meter`) are also expanded.

Because of this processing, the translation model implicitly learns to insert punctuation marks, restores word casing, and converts spoken forms to digits as part of the translation process since the target side of the training corpus contains punctuation marks, cased tokens, and digits. We refer

Table 6.1: Performance comparison of the MT and cascade ST systems using different intermediate processing on IWSLT English→German. All MT models are based on the RNN attention model, while the ASR model is a hybrid HMM model.

task	**processing**		**dev2010**		**test2015**	
	MT input	**ASR output**	BLEU [%]	TER [%]	BLEU [%]	TER [%]
text translation	standard	-	31.3	49.6	32.4	50.5
	asr-like	-	29.8	50.7	31.7	51.7
cascade speech translation	standard	w/o punc.	25.7	56.8	25.2	60.3
		w punc.	24.4	58.4	24.7	60.7
	asr-like	w/o punc.	26.3	55.8	26.7	57.9

to this technique as *asr-like processing*. The same processing is used for the English monolingual data as in ASR language modeling.

Table 6.1 presents the translation performance of these two processing techniques. All MT models are RNN attention-based, while the ASR model is a hybrid HMM model. As expected, in the pure text translation, the standard processing outperforms the asr-like one (cf. lines 1 and 2) because the asr-like input contains a bit of noise and not the ground-truth source text. Applying asr-like processing decreases the performance by 1.1 points in both BLEU and 1.1% in TER on average. In contrast to this, feeding the ASR output, in which there is no punctuation, to the MT model trained using the standard processing (line 3) is behind the model trained with the asr-like scheme (line 5). This can be partly attributed to the fact that in the asr-like processing, punctuation prediction is made implicitly by the translation model. Such a scheme helps the coupling of the components and is able to predict outputs of proper length for noisy inputs, even a small amount of noise. This is also observed by [Sperber & Niehues$^+$ 17]. The MT system expects a well-formed sentence with punctuation marks. In order to match the ASR transcripts with the input of the MT system trained with the standard processing, expecting well-formed sentences, we also insert the punctuation marks into the ASR output as the second alternative.

The punctuation prediction model is a simple attention model proposed by [Tilk & Alumäe 16] and taken from [Matusov & Wilken$^+$ 18]. The model is applied explicitly to the ASR output to enrich it with punctuation before feeding it to the MT model. The result is shown in the 4th line of Table 6.1. This leads to worse performance than the setting with no punctuation but already seen asr-like examples during training. One reason for the degradation is that the punctuation predictor is not error-free, and punctuation marks in wrong positions can change the meaning of a sentence and mislead the MT model, which has seen correct punctuation. The second reason can be related to the punctuation model in which no acoustic features are used; only the words in a sentence are considered to predict punctuation. These symbols are limited to periods, commas, and question marks. In both cases, the degradation is mainly due to recognition errors, while the asr-like processing improves the robustness of the MT model to the same kind of noise.

This work assumes word and punctuation errors compound and thus applies the asr-like processing. We also suppose that the input to the translation system is a properly-formed sentence. In the following sections, the text translation is referred to as the model with asr-like processing given that its performance is lower than the standard processing but is the best-obtained setup.

Cascade vs. Text Translation Once we have developed the asr-like text processing, we couple the end-to-end attention-based ASR and MT models as the cascade system. We first compare the performance of the cascade speech translation systems against that of text translation.

Table 6.2 again indicates that cascade speech translation is behind text translation on IWSLT

Table 6.2: Performance comparison of the text MT and cascade ST systems on IWSLT English→German. Both ASR and MT models are based on RNN attention.

task	dev2010		test2015		#
	Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	parameters
text translation	29.8	50.7	31.7	51.7	200M
cascade speech translation	26.2	56.1	27.3	57.5	427M

Table 6.3: Performance comparison of the text MT and cascade ST systems on MuST-C English→Spanish using both the RNN attention and self-attentive models. The rightmost column of the bottom part of the table counts all trainable parameters of the ASR, MT, and source-side LM in the cascade system.

task	model	testHE		testCOMMON		#
		Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	parameters
text translation	RNN attention	41.9	41.8	34.1	50.0	202M
	self-attentive	42.7	41.8	35.4	49.0	260M
cascade speech translation	RNN attention	38.5	45.8	31.2	54.3	494M
	self-attentive	39.0	46.0	31.5	55.7	653M

Table 6.4: Performance comparison of the text MT and cascade ST systems on LibriSpeech English→French using the RNN attention model.

task	dev		test		#
	Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	parameters
text translation	25.2	62.6	22.9	64.4	166M
cascade speech translation	23.3	65.2	21.4	66.4	334M

English→German with the total number of parameters on the rightmost column. This is an inevitable phenomenon because of the compounding follow-up errors produced by each cascading component, specifically the ASR model. Here we note the difference between line 2 of Table 6.2 and line 5 in Table 6.1. The former case is based on the end-to-end RNN attention model, whereas the latter uses the hybrid ASR model (see Table B.2 for Wer comparison.)

On MuST-C English→Spanish in Table 6.3, we examine both the RNN and self-attentive models. The cascade model of each model group (either RNN or self-attentive model) is composed of the ASR, source-side LM, and MT models. All components of each group are based on the same architecture, either RNN or self-attentive. More details and the combination of these components are presented in Table B.5.

In addition to the inherent performance degradation of speech translation compared to text translation, we see that the self-attentive models work slightly better on both tasks. The source-side language models reduce the word error rate of the ASR models as reported in Appendix B.3 as well as increasing the performance of the cascade systems as intended (see Table B.5 in Appendix).

Table 6.4 reconfirms the previous observation on LibriSpeech English→French. In addition to the error propagation problem as seen in these tables, the ASR and MT models treat the source language transcript differently, and we have to find a solution for better coupling of these two modules such as those proposed via asr-like processing. Such a two-step property also leads to

Table 6.5: Direct ST results with CTC-attention loss on IWSLT English→German.

method	CTC	reduction factor	dev2010 Bleu[%]	dev2010 Ter[%]	test2015 Bleu[%]	test2015 Ter[%]	# parameters
direct	no	8	14.8	69.8	14.9	73.5	181M
		6	16.0	69.2	15.3	73.2	
	yes	8	17.9	66.3	16.5	70.4	192M
		6	19.4	64.1	18.6	67.6	

losing speech information at the final step due to the independence assumption between the target sentence and the speech utterance. Further, the system is inherently sequential due to the two-step modeling. Therefore, it requires a two-pass search, resulting in a higher decoding time. In general, such systems also need more computational power to train a given number of models, in most cases two. As seen in the rightmost column of the tables, the cascade model have more than twice or as many parameters as the text models. Given these reasons, we explore the direct models next.

Direct Speech Translation

CTC Loss in Speech Translation The frame-wise input sequence is much longer than the source sequence, i.e., $J \ll T$. According to what is reported in [Zeyer & Irie$^+$ 18], the input sequence is 30 times longer on Switchboard if audio feature frames of every 10ms and subword segmentation of 10k are applied. The max-pooling time reduction is a common remedy to compress any necessary information. On the one hand, the lower the reduction factor is, the more memory and computation for the attention layer are needed. On the other hand, a higher time reduction factor leads to faster and simpler training; however, the model does not converge if a very high factor is selected, as it has also been observed for speech recognition [Chan & Jaitly$^+$ 16, Zeyer & Irie$^+$ 18].

In contrast, CTC defines frame-wise label sequence z_1^T, and its final classification involves prediction per frame. Therefore, the number of final frames after reduction may play an important role in CTC performance. It requires us to find a compromise between the performance and the time reduction factor. Along with applying CTC, we change the time reduction factor in Table 6.5. A lower time reduction of 6 instead of 8 helps the direct model performance with and without CTC loss applied; however, the batch size needs to be adjusted. As expected, the performance difference of the time reduction factors is more pronounced if CTC is activated. It helps 2.1 points in Bleu and 2.8% in Ter on `test2015`. This gain is much larger than 0.4 points and 0.3% when no CTC is used.

In addition, comparing lines 2 and 4, we gain a large improvement of 3.3 Bleu points and 5.6% absolute in Ter on the IWSLT test set. We highlight that we apply a different layer-wise construction when the reduction factor is set to 6. In this scheme, the hidden dimension is gradually increased in addition to the number of encoder layers. Therefore, the improvements can be partially attributed to better optimization when comparing the reduction factor of 8 and 6. For higher downsampling, the batch size has to be limited and further hinders model optimization.

Table 6.6 also verifies gains on MuST-C English→Spanish, though the improvements are smaller. These results are obtained with a total time reduction factor of 6 as the best-observed setup.

Apart from the final translation improvements, the CTC auxiliary objective function slightly accelerates the learning curve and converges faster, as shown in Figure 6.7. This behavior is observed even when the two losses are equally weighted. From speech recognition training, the convergence without CTC might be too challenging. One potential reason for such gains might be that the CTC function improves the lack of left-to-right constraints or monotonic alignments in the speech part of the network, especially for long sequences. Faster convergence with CTC also

Table 6.6: Direct ST results with CTC-attention loss on MuST-C English→Spanish. The time reduction factor is chosen to be 6.

method	**CTC**	**testHE**		**testCOMMON**	
		BLEU[%]	TER[%]	BLEU[%]	TER[%]
direct	no	29.0	55.5	23.6	62.9
	yes	30.5	53.1	25.1	61.0

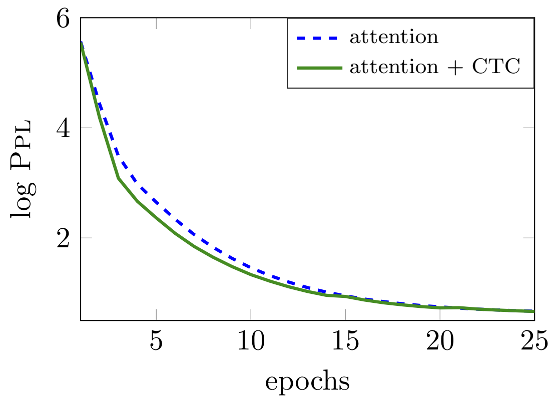


Figure 6.7: The learning curve of the direct ST attention model based on the logarithm of training perplexity with and without the CTC auxiliary loss on IWSLT English→German.

indicates easier training of the attention-based models from scratch due to better alignments on longer sequences. Although speech translation needs some reordering, we think there are local monotonicities among consecutive frames. In fact, the CTC loss is similar to adding an additional output layer to predict the transcripts, thus it can be seen as multi-task learning. The multi-task property of CTC might also contribute to the performance difference. It may also be partly due to better optimization. The CTC loss adds $d_f \times (|V_f| + 1)$ extra parameters to the model, as stated in the rightmost column of Table 6.5. d_f represents the hidden dimension before the output layer.

SpecAugment in Speech Translation In the first set of the SpecAugment experiments, we deactivate either time or frequency masking in a batch. We exercise different values of Δd_{max}, between 2 and 40, and increase m_d gradually while the time masking is disabled. We found that, on the one hand, a very large value of frequency masking ($\Delta d_{max} = 40$) can hurt the performance. On the other hand, even a small value ($\Delta d_{max} = 2$) leads to some gains. In order to keep a reasonable ratio of frequency masking, M_d has to be limited. By repeating the frequency masking of big windows more than 5 times, augmentation is less beneficial than anticipated. It is important to highlight that since we randomly select the masking window between zero and the maximum value of Δd_{max}, the results are close to each other.

We also disable the frequency masking by setting Δd_{max} and m_d to zero and vary the time mask parameter Δt_{max} and the number of times it is called. Again, there is a limit to how much data augmentation can be applied. Enlarging the time masking window Δt_{max} to 100 leads to lower BLEU and TER scores. Furthermore, we drastically increase the time masking window Δt_{max} into 400 steps and apply it 5 times, which fails. Too short time maskings are also less effective, indicating less augmentation [Bahar & Zeyer+ 19b].

In general, if the initial convergence is stable, in all cases, adding some data augmentation improves the setup. However, at some point, the performance degrades with more augmentation.

Table 6.7: Direct ST results with SpecAugment on IWSLT English→German.

method	SpecAugment	dev2010		test2015	
		BLEU[%]	TER[%]	BLEU[%]	TER[%]
direct	no	19.4	64.1	18.6	67.6
	yes	20.5	62.9	19.7	66.5

Table 6.8: Direct ST results with SpecAugment on MuST-C English→Spanish.

method	SpecAugment	testHE		testCOMMON	
		BLEU[%]	TER[%]	BLEU[%]	TER[%]
direct	no	30.5	53.1	25.1	61.0
	yes	31.3	52.7	25.7	60.2

Moreover, it is observed that in many cases applying a small window several times results in a better augmentation policy compared to a large window applied once [Bahar & Zeyer$^+$ 19b]. In the end, one optimum value of maximum frequency masking that works reliably for our setup with the 80-dimensional MFCC features is Δd_{max} to be 20% of the feature dimension, i.e., 16, and respectively the maximum time masking window of size 20. The combination of two maskings gives further boosts. The predefined M_t and M_d are halved in the first 2000 batches for having stable training at the beginning. Adding some data augmentation improves the performance without increasing model size but with a slightly longer convergence time. SpecAugment performs quite well regardless of the features and their dimensions but needs a bit of tuning. SpecAugment boosts the performance of the direct model by an improvement up to 1.1 points in BLEU and 1.1% in TER on average on the IWSLT task, as shown in Table 6.7. Table 6.8 presents further results on the MuST-C test sets with less gain up to 0.7 points in BLEU and 0.6% in TER. This observation may be due to the fact that the direct speech translation data used in MuST-C is larger than IWSLT, and SpecAugment as an augmentation approach is more beneficial on lower-resource tasks.

Importance of SpecAugment on Overfitting We also investigate the effect of SpecAugment on overfitting. Figure 6.8 plots the logarithm of the perplexities of the training and development sets with and without augmentation. From the perplexity difference, it is conclusive that SpecAugment leads to better generalization. The model trained with SpecAugment still has training likelihood that is higher than the baseline system. Thus, it can be confirmed that the method reduces overfitting. In this case, the augmented model has to be trained for a few more epochs. Comparing Tables 6.7 and 6.8 also shows that SpecAugment helps translation performance of the direct model more on the low-data condition (considering IWSLT with a lower amount of training data); however, it avoids overfitting to some extent irrespective of the amount of training data. A study on the impact of a different portion of training data has been reported in [Bahar & Zeyer$^+$ 19b]. The augmentation policy compensates for the lack of data when the size of the training data is halved. Being successful, the SpecAugment method is included in the rest of the experiments.

Sequential Transfer Learning

This section explores sequential transfer learning or simply pretraining for direct models in which parts of the network are initialized with parameters from other models: initially trained for different, though similar tasks. In theory, initializing the parameters with pretrained models should not hurt the fine-tuning optimization, but in practice, it often does because the pretrained

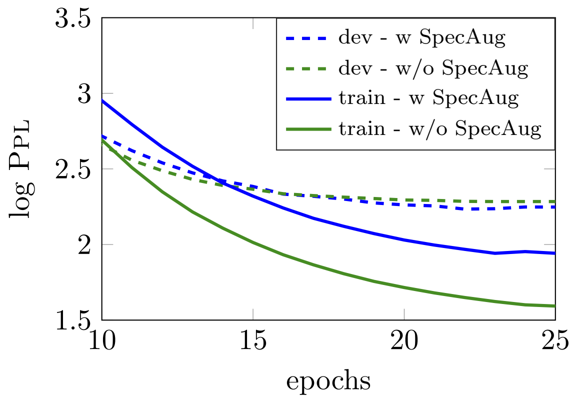


Figure 6.8: Average of the logarithm of the perplexity of the training and development sets across epochs on IWSLT English→German.

components are not incorporated properly together. In this case, it might lead to an optimization problem in which the training is not robust to new training data variations.

To enable pretraining, for the ST models, we use the same architecture as the ASR encoder and the same architecture as the MT decoder. Here, the ASR and MT models described for the cascade system are used. In the initial experiments, the CTC loss with pretraining hurts convergence. This might be due to the fact that CTC mostly helps convergence on the speech side of the model, and with pretraining, we use an already-converged model that does not suffer from convergence issues. Therefore, all pretraining experiments exclude the use of CTC.

We conduct different pretraining schemes, as shown in Table 6.9. Pretraining strategies can be used on different models with different recipes. The first and second columns of the table indicate what component is employed to initialize either the direct model encoder or its decoder. For example, the second row refers to the case in which we initialize the network encoder with the ASR encoder, while the decoder is randomly initialized. In Figure 6.2, the blue block is activated. Similar to [Bansal & Kamper^{+} 19], we also initialize the German text decoder using the pretrained English ASR decoder, and the results are similar to the second line of the table.

Pretraining on the encoder gives a gain of 0.9 points and 2.2% in terms of Bleu and Ter on `test2015`. It is interesting and surprising that transferring knowledge only from the speech encoder using the ASR model seems sufficient to improve over the baseline.

Then, the MT decoder is taken to initialize the text decoder parameters (the green block in Figure 6.2). We observe that pretraining the text decoder seems not to be useful, as the model diverges, shown by "-". This can be attributed to a larger impact of the decoder on the final performance compared to the encoder. One reason for this might be that error is propagated through the decoder down to the speech encoder. Naturally, this leads to the decoder receiving larger updates than the encoder, ultimately resulting in a decoder which translates the output of a still terrible encoder. These observations also mean that the pretrained MT decoder expects to be jointly trained with a text encoder, not a speech encoder.

In the next step, both the pretrained speech encoder and text decoder are used at the same time (both green and blue blocks in Figure 6.2). Again, as shown in line 4, coupling the pretrained ASR encoder and MT decoder fails. This verifies our previous assumption about a proper adaptation of both parts.

Adapter Layer Based on this, we add an additional layer, one bidirectional LSTM, as the adapter component to familiarize the input of the pretrained decoder with the output of the pretrained encoder. This is added on top of the encoder and randomly initialized. The adapter component is

Table 6.9: Comparison of the different pretraining schemes used in the direct ST model on IWSLT English→German corresponding to the blue and green blocks of Figure 6.2. The first line is the direct model without pretraining copied from Table 6.7.

pretraining		adapter	dev2010		test2015	
encoder	decoder		Bleu[%]	Ter[%]	Bleu[%]	Ter[%]
none	none	no	20.5	62.9	19.7	66.5
ASR	none		21.4	60.6	20.6	64.3
none	MT		-	-	-	-
ASR	MT		-	-	-	-
ASR	MT	yes	22.9	59.9	22.3	63.5

"-": failed results.

Table 6.10: Direct ST results with pretraining on MuST-C English→Spanish.

method	pretraining	testHE		testCOMMON	
		Bleu[%]	Ter[%]	Bleu[%]	Ter[%]
direct	no	31.3	52.7	25.7	60.2
	yes	35.9	48.0	28.7	56.3

Table 6.11: Direct ST results with pretraining on LibriSpeech English→French.

method	pretraining	dev		test	
		Bleu[%]	Ter[%]	Bleu[%]	Ter[%]
direct	no	17.3	72.1	18.1	70.9
	yes	20.6	66.9	21.3	66.0

jointly trained with all other parameters without freezing in fine-tuning. Adding the adapter layer achieves an improvement over the randomly initialized direct model by 2.6 Bleu points and 3.0% Ter on `test2015`. It allows the model to learn a mapping between the learned representations. Overall, pretraining seems to help a lot when it comes to training direct models. Pretraining using the ASR and MT models in combination with the adapter component yields the best performance with 22.3% and 63.5% in Bleu and Ter on the test set.

We also state that if the decoder attention layer is not initialized, the entire network still transfers information from both pretrained components that provide decent improvements. This non-pretrained layer between the encoder and decoder can justify a smoother coupling. The justification for using non-pretrained components in between can be extended to other layers of the network. Arguably, instead of inserting an adapter layer, one could keep the same number of layers and avoid initializing the last layer of the bidirectional encoder. Initial experiments have shown the effectiveness of this.

The same pretraining scheme with the adapter component (one BiLSTM layer) is therefore carried out on MuST-C English→Spanish and LibriSpeech English→French, whose results of which are shown in Tables 6.10 and 6.11, respectively. The results are similar to the observation on the English→German task. Pretraining helps the MuST-C task by an average of 3.8 Bleu points and 4.3% absolute Ter on two test sets. It also gives a similar boost of 3.4 points in Bleu and 4.9% in Ter on the English→French test set. This performance difference is larger than anticipated. Although using more training data is the main reason, it is worth mentioning that

Table 6.12: Multi-task learning results on IWSLT English→German. The models have the same number of parameters in decoding.

method	pretraining	dev2010		test2015	
		Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
direct	no	14.8	69.8	14.9	73.5
two-to-one		18.2	67.2	18.1	69.8
one-to-two		17.4	67.0	16.2	72.7
direct	yes	22.9	59.9	22.3	63.5
two-to-one		20.1	63.8	19.9	66.8
one-to-two		21.1	61.8	20.3	66.1

the method also works for the low-resource LibriSpeech task. In this task, the direct ST model has an equal amount of parallel training data compared to the ASR and MT tasks. Selecting suitable source tasks might be the reason for the gain. The same amount of data is used but for different purposes. Moreover, the ST training starts from a point with optimum parameters.

Pretraining is closing the gap between direct and cascade modeling by transferring knowledge from a higher resource, yet similar tasks. However, it lags behind the cascade models. Such a concept can also be applied in other NLP tasks. A simple example is the translation of distant language pairs, where English is used as a pivot in between. Similarly, source→English and English→target models can be trained and then exploited in a transfer learning with an adapter in between to familiarize the source encoder and target decoder [Kim & Petrov$^+$ 19].

Multi-Task Learning

We compare the one-to-two and two-to-one multi-task methods with the ASR and MT tasks as the co-trainers. Since the CTC loss is a multi-task setup, we exclude it as initial experiments indicate that it lowers the performance of multi-task learning. For each batch, a certain number of sequences is chosen at random such that the average batch contains that portion of the DST, MT, or ASR data. For the one-to-two and two-to-one methods, the best setup is achieved by performing 60% and 70%, respectively, of training steps on the core speech translation task and the remaining updates on the auxiliary task.

Table 6.12 illustrates that all models benefit from the additional data through pretraining. The methods listed in the first block of the table apply neither CTC nor SpecAugment. The models have the same number of parameters in decoding, while the direct, two-to-one, and one-to-two models equipped with adapted pretraining have 206M, 363M, and 252M parameters, respectively. Having studied the relative gains, we observe that the direct model takes the most advantage of the pretrained components and surpasses the multi-task approaches with pretraining by almost 2% and 2.6% absolute difference in Bleu and Ter on the test set.

By comparing the multi-task strategies, pretraining helps the one-to-two more than the two-to-one method; however, without pretraining, the one-to-two performance is ahead. This observation may be related to the amount and quality of ASR and MT data. When the additional MT data (27M sentence pairs) is utilized in the two-to-one setup (line 2), it outperforms the one-to-two method with a smaller amount of ASR data (260K segments). As long as all auxiliary data are used via pretraining, multi-task learning plays a role in guiding the main task. In this case, the ASR auxiliary task helps more than MT, and this is similar to the results reported in [Weiss & Chorowski$^+$ 17, Berard & Besacier$^+$ 18]. As stated, multi-task learning is applied to supervise speech translation, use additional data, or both. Our contribution differs from these papers as we aim to use more training data rather than only guiding the ST task. In general, multi-task

Table 6.13: Comparison of different sharpening factors against performance scores on the cascade of ASR and MT models in decoding. The ASR model renormalized posteriors with γ exponent are passed on to the MT encoder.

beam size	γ	**dev2010**		
		BLEU$^{[\%]}$	TER$^{[\%]}$	WER$^{[\%]}$
1	-	26.0	56.8	12.2
12	-	26.2	56.1	11.2
	0.5	-	-	-
	0.9	12.9	79.7	11.3
	1.0	21.9	64.6	11.2
	1.5	26.0	56.6	11.6
	2.0	26.0	56.7	11.7
	4.0	26.0	56.7	12.0
	32	25.9	56.8	12.2
	128	25.9	56.8	12.2

learning achieves no real benefits on top of direct modeling if pretraining is applied. This is similar to what is observed by [Sperber & Neubig^{+} 19a].

Tight Integration of Cascade Model

Effect of γ in Passing Posteriors After obtaining the best hypothesis for each utterance via the ASR model, we pass it to the MT model. To this end, we concatenate the ASR and MT networks, and as a sanity check, we pass the one-hot vectors as the ASR outputs to MT. This leads to the exact same result as in the standard cascade system. Unlike the cascade system, this step needs to have a shared source vocabulary between ASR transcripts and MT source sentences. Otherwise, coupling requires a linear mapping between the two components that have to be trained first.

This part sheds light on the question of which sharpening factors in Equation 6.21 make the distribution more peaked and suitable for the MT model. By varying γ, we adjust the distribution to be closer to the one-hot representations expected by the MT embedding layer. Table 6.13 lists the translation performance of the cascade models on `dev2010` for IWSLT English$\rightarrow$German by passing renormalized posterior probabilities to the MT model while applying different sharpening factors in decoding. No fine-tuning using direct speech translation data is utilized in this table, and an identical set of parameters to the cascade model is involved. Results are compared with the cascade system where the one-hot vectors of the subwords in the transcript hypothesis are passed to the MT model with beam sizes of 1 and 12 on the first and second lines of the table.

As expected, for a small value of $\gamma = 0.5$, no reasonable translation can be achieved, as the probability of source words is uniformly distributed over the vocabulary. For values below 1, a degradation in performance is seen. For larger values, the performance shows a small gain until it settles at around 26.0% BLEU. We have found that $\gamma = 2$ is on par with the results of the performance of the cascade system. Beyond that, the performance barely changes as the distribution is already sharp enough, and a further increase of γ does not alter the distribution.

The MT encoder is trained to see the one-hot vectors and does not know how to deal with the probability distributions like posteriors. In fact, this performance is similar to greedy search on the ASR decoder of the cascade model (cf. line 1 of the table). This is due to the nature of the sharpening not representing the proper choices of beam search. Sharpening emphasizes the most probable token at each position. Nevertheless, in decoding, it is only the case for the previous token in the sequence, and here we deal with the sequence whose total probability is

Table 6.14: Performance comparison of the tight integration cascade model with other methods on IWSLT English→German. All models are trained using equal amounts of training data; i.e., ASR, MT, and DST corpora, except the text translation method.

method	dev2010		test2015	
	BLEU[%]	TER[%]	BLEU[%]	TER[%]
text translation	29.8	50.7	31.7	51.7
cascade	26.2	56.1	27.3	57.5
tight integration cascade	26.8	55.1	28.1	56.7
direct†	22.9	59.9	22.3	63.5
two-to-one†	20.1	63.8	19.9	66.8
one-to-two†	21.1	61.8	20.3	66.1

† includes adapted pretraining.

highest. Intermediate tokens might not have the best probability scores at their position, resulting in reranking of the tokens.

To prove this, the ASR model is evaluated using sharpening on the posterior probabilities, and the results are listed in the table. The word error rates also show a similar pattern, where for higher γ factors, the performance decreases until it settles to the performance of recognition with simple greedy search with 12.2% WER. To convey more information and pass more ambiguities of the recognition system, the experiments take advantage of $\gamma = 1$ in training and $\gamma = 1.5$ or 2 in decoding. However, the choice of γ needs tuning from one task to another.

Translation Performance Table 6.14 presents the IWSLT results when we tighten the ASR and MT models and fine-tune the coupled model using speech-to-target data in an end-to-end fashion. We also compare it to all the models studied so far. All models are trained using equal amounts of training data; i.e., ASR, MT, and DST corpora, except the text translation method trained on the bilingual MT corpora. As expected, due to error propagation in the cascade system, its performance is significantly lower than that of the pure text translation model. The direct model performance is far behind the cascade model. In multi-task learning, auxiliary data helps only to some degree, and with pretraining it does not help at all. The optimization of multi-task models is extremely dependent on the DST data. Our results also indicate that the tight integration cascade outperforms the direct model by 5.8 points in BLEU and is more pronounced in TER on `test2015`. It also beats the results of the cascade model by 0.8 points in BLEU, and reduces TER by 0.8%.

The results on other tasks are very similar. They are shown on the MuST-C and LibriSpeech tasks in Tables 6.15 and 6.16, respectively. For MuST-C, we use 2300h extra ASR data, which is almost eight times more than in IWSLT. The amount of DST data is also larger compared to the IWSLT task. All the methods discussed in this chapter are architecture-independent and can be constructed using any neural network. To provide a comprehensive overview, we also examine the performance of well-established methods using the self-attentive-based models on the MuST-C task. With this data condition and pretraining, we close the gap between the cascade and direct models and get a difference of 1.5 points in BLEU and 0.7% in TER on `testCOMMON` using the RNN attention models, while there is less improvement using the self-attentive architecture (Table 6.15). Using the self-attentive models, a leap in performance is observed except for direct models on `testCOMMON`. Again, the tight integration model outperforms the cascade system as well as the direct model on all test sets.

On LibriSpeech, we only use the DST data for model training limited to 47.2k segments. As presented in Table 6.16, the direct model reaches the cascade model on the test set and even yields

Table 6.15: Performance comparison of the tight integration cascade model with other methods on MuST-C English→Spanish using the RNN and self-attentive modeling. All models are trained using equal amounts of training data; i.e., ASR, MT, and DST corpora, except the text translation.

method	**RNN modeling**				**self-attentive modeling**			
	testHE		**testCOMMON**		**testHE**		**testCOMMON**	
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]
text translation	41.9	41.8	34.1	50.0	42.7	41.8	35.4	49.0
cascade	37.1	47.1	30.2	55.4	37.5	47.4	31.2	54.7
+ source LM	38.5	45.8	31.2	54.3	39.0	46.0	31.5	55.7
tight integration cascade	38.0	46.4	30.6	55.0	38.8	46.3	31.1	54.5
+ source LM	38.1	46.1	30.8	55.0	39.2	45.6	31.4	54.7
direct†	35.2	49.1	28.7	56.1	35.5	48.7	27.5	60.6

† includes adapted pretraining.

Table 6.16: Performance comparison of the tight integration cascade model with other methods on LibriSpeech English→French. All models are trained using equal amounts of training data, only the DST corpus.

method	**dev**		**test**	
	BLEU[%]	TER[%]	BLEU[%]	TER[%]
text translation	25.2	62.6	22.9	64.4
cascade	23.3	65.2	21.4	66.4
tight integration cascade	23.2	64.7	21.4	65.8
direct†	20.6	66.9	21.3	66.0

† includes adapted pretraining.

a better TER score. This is clearly due to data constraint in the cascade model training. There, the tight model is also comparable to the cascade system since the model parameters are trained using already seen samples, and fine-tuning does not bring improvements in terms of BLEU, though it helps TER by 0.5%. A larger improvement is achieved on the dev set. On a low-resource data condition, the tight integration model guarantees at least the performance of the cascade system.

Source Side Language Model Combination As shown by the results in Table 6.15, we train the source-side RNN and self-attentive language models. The perplexities of the language models are reported in Table B.4 in Appendix. Using shallow fusion, we integrate LMs into the cascade and tight integration models in decoding. For the latter, the sharpening factor γ is set to 1. The LM scale is found to be in the range of 0.15-0.20, which shows its small contribution to the final performance. Language models usually help the final translation performance due to better WER on the ASR side. However, their contribution is more pronounced in the cascade model compared to its combination with the tight integration model. The LM improvements over the tight integration models are small but consistent. Comparing the RNN and self-attentive LMs, the self-attentive LM combination leads to more gain due to having lower perplexity.

Experimental results show that the tight integration cascade model successfully improves the data efficiency issue of end-to-end models; however, it still suffers from the error propagation problem, as the second part of the model relies on the first decoder and its decisions. Similar to the cascade models, the decisions are not error-free due to non-zero error rates.

Table 6.17: Impact of freezing ASR in the tight integration cascade model training.

ASR training	**IWSLT**			**MuST-C**			**LibriSpeech**		
	test2015			**testCOMMON**			**test**		
	BLEU [%]	TER [%]	WER [%]	BLEU [%]	TER [%]	WER [%]	BLEU [%]	TER [%]	WER [%]
yes	28.1	56.7	13.0	30.6	55.0	11.2	21.4	65.4	16.0
no	28.4	56.3	10.6	30.9	54.8	10.4	21.2	65.7	16.2

Effect of Subtask Training The tight integration training can be considered as a multi-task setup of recognition and translation jointly, where the goal is to improve the generalization performance of the ST task. In this case, we need to compromise between multiple tasks, and this might lead to a suboptimal solution for the entire optimization problem. Therefore, we further study the freezing of the ASR part of the network and only fine-tune the MT parameters. This setup guarantees the initial WER obtained by the ASR model, and further training does not affect it. The tight model can generally take the recognition quality into account in addition to translation performance. This is particularly important for those applications where we need to generate both transcription and translation. For example, when both have to be displayed to users.

Both translation and recognition results are listed in Table 6.17. Training all parameters results in worse WER on IWSLT and MuST-C (compare with WER in Tables B.2 and B.3). By freezing the ASR parameters and only updating the MT part of the network, it favors the ASR task as we do not update the ASR parameters and guarantee exactly the same WER. Due to better WER compared to line 1, better BLEU and TER scores are achieved. This implies a high correlation between transcription and translation and indicates a strong consistency between them. We believe this consistency is an inherent property of the tight model as translation depends on transcription. Similar to [Ruiz & Gangi$^+$ 17], we also observe a correlation between recognition and translation quality, and the degradation in performance is mainly due to recognition errors. This produces different results on LibriSpeech with limited training data. As shown, joint training of all components slightly helps both recognition and translation performance. However, the correlation between the results stays consistent. We conclude that the tight integration cascade model is preferable in many practically relevant situations where there are relatively large speech-to-source and source-to-target data, but only limited amount of speech-to-target data. The model provides both transcribed speech utterances and translated texts with a high correlation between them.

Comparison with Other Works We also compare our models with various methods recently published on the benchmark LibriSpeech English→French task with only 100h of data. Table 6.18 shows the results of this comparison. The first three lines of the table represent different cascade systems. All are behind our cascade model, presented at the end of the table. Pretraining is used with direct modeling in [Berard & Besacier$^+$ 18]. The success of the self-attentive model inspires its network architecture for direct speech translation [Gangi & Negri$^+$ 19c]. A text translation model can be used as a teacher to transfer the knowledge from [Liu & Xiong$^+$ 19]. In their model, the speech translation task learns output probabilities from the MT model. Universal attention-based models in multilingual settings are also used in speech translation [Inaguma & Duh$^+$ 19], where a many-to-many model is trained on a mix of English to French, Spanish, and German with a total of 472h of English speech. The curriculum learning paradigm is combined with pretraining by gradually increasing the difficulty of the pretraining task [Wang & Wu$^+$ 20]. As shown in the table, our models beat almost all other methods from the literature except the direct model equipped with extra synthetic data [McCarthy & Puzon$^+$ 20].

Table 6.18: Comparison of the models with the recently published works on LibriSpeech English→French measured by `multi-blue.pl` on the tokenized lower-cased reference.

method	**model**	BLEU [%]
		test
cascade	RNN [Berard & Besacier$^+$ 18]	14.6
	self-attentive [Liu & Xiong$^+$ 19]	17.9
	RNN, synthetic data† [McCarthy & Puzon$^+$ 20]	21.3
direct	RNN, pretraining [Berard & Besacier$^+$ 18]	13.3
	self-attentive [Gangi & Negri$^+$ 19c]	13.8
	+ knowledge distillation† [Liu & Xiong$^+$ 19]	17.0
	+ multilingual† [Inaguma & Duh$^+$ 19]	17.6
	+ curriculum pretraining [Wang & Wu$^+$ 20]	17.7
	RNN, synthetic data† [McCarthy & Puzon$^+$ 20]	22.4
direct	RNN attention	21.3
cascade		21.4
tight integration cascade		21.4

† more training data are used.

6.6 Implementation

RETURNN offers a variety of dataset classes that are each specialized for a particular task. For instance, `TranslationDataset` loads parallel bilingual text data used for machine translation, and `ExternSprintDataset` loads audio features generated by RASR and their corresponding transcripts used for speech recognition. Combining all three types of data, i.e., the DST, MT, and ASR data, a speech translation system uses the `MetaDataset` or `CombinedDataset` class. These implementations allow us to create a combined corpus dynamically and avoid manual concatenation and shuffling. One of the main use cases for the `MetaDataset` class is multi-task learning in which the data of the co-trainer task, either audio features or texts, should be chosen at each batch. We define a pickle file containing a list of sequence tags for each dataset such that it defines which training samples from which datasets belong together.

Two-step modeling is simple in training because we obtain a single sequence of hidden vectors in the forward pass. This sequence can then be passed to the next layer, i.e., the MT encoder or decoder. In RETURNN, in order to write the output directly into a hypothesis file, the end-of-sentence token is omitted at the end of the generated output sequence and all internal states. On the other hand, in the cascade pipeline, the end-of-sentence token on the ASR output has to be kept as the output is passed to the MT model, where we expect to have an end-of-sentence token. To do so, one needs to set a boolean option called `include_eos` in the ASR decoder that keeps the token for the next step.

By default, the decoder returns N beams from the ASR model with the best choices at each position together with their respective scores and the index of choices made by the decoder for each subsequent position. Then a decision layer traces back through this sequence of choices to obtain the best hypothesis. This decision layer can also be applied to pass the best hypothesis from the ASR decoder to the MT encoder. However, when passing on internal states rather than the final decision, this feature no longer works because the layer that computes the internal state does not know the score by which the decisions are made or the actual choices. Therefore, tracing back through internal states is not possible. Our implementation reads the sequence of scores and decision indices from the decoder output. Together with the internal states, these are processed to perform the trace-back similarly to the decision layer, but on the sequence of hidden states.

Starting at the latest position, the sequence with the highest score is chosen based on the final states' scores. The indices provide information on which of the previous words in the beam belong to that sequence. Based on these indices, all previous positions are traced back, and the respective states are extracted. We trim the decision layer output to the maximum remaining sequence length as it causes issues in the following recurrent layers with mismatches between the true and predicted sequence lengths. Additionally, the N-th best beam can be extracted, allowing us to ensemble multiple ASR beams and pass on the results to the MT model. The implementation also enables the extraction of the internal states for a specific beam from the ASR decoder by adding them to the network during search. This is done mainly to verify the results of two-step modeling by [Anastasopoulos & Chiang 18, Sperber & Neubig^{+} 19a].

6.7 Conclusion

This chapter concentrated on the task of speech-to-text translation as a challenging task that combines both speech recognition and translation issues and aims to solve them. In particular, we explored different end-to-end models. In the extreme case, direct models were addressed instead of noisy channel decomposition and the independence assumption between the target sentence and the speech utterance. We examined the combination of CTC with attention and proposed the use of SpecAugment for speech translation, resulting in a practical training procedure. The CTC loss improved translation performance with faster training convergence (see Tables 6.5 and 6.6), and the SpecAugment method achieved some gains with better generalization (Tables 6.7 and 6.8).

To tackle the data scarcity issue of the end-to-end models, we studied transfer learning. A straightforward solution for utilizing all available training data is pretraining, which proved to be very effective if an adaptation is applied. We showed that the pretraining scheme with suitable source tasks like speech recognition and text translation led to significant improvement and could mitigate the data efficiency of the direct models even for a low-resource task (Tables 6.9 to 6.11). Another investigated method was multi-task learning, but we showed that there was no real benefit in using it (Table 6.12) since it led to a suboptimal solution for the whole optimization problem.

As another remedy, we proposed a tighter coupling between automatic speech recognition and text translation. We developed the tight integration cascade model that naturally decomposed into two modeling steps akin to the cascade but preserved end-to-end trainability and reduced the early-decision problem. We presented experimental results which proved that the integrated cascade model outperformed all other approaches under realistic data conditions, was much more effective at exploiting auxiliary data, and provided consistency between transcripts and translated texts due to its two-step nature (Tables 6.14 to 6.18). This indicates that the end-to-end models can realistically translate automatically recognized speech as a relevant solution to speech translation. Such models can replace the traditional cascade systems as they are able to compete with or outperform them.

6.8 Related Work

The research topic of speech translation has emerged with the success of statistical data-driven methods in both ASR and MT. Traditional speech translation systems are built in the cascade fashion and comprise an automatic speech recognition model trained on paired speech-transcribed data and a machine translation model trained on bilingual text data. In practice, a zero recognition error rate can rarely be achieved.

A theoretical justification of integrated speech translation by [Ney 99] has motivated some works for a tighter coupling of ASR and MT models. A better coupling can be done by processing ambiguities of ASR outputs in the form of word lattices, N-best lists, and confusion networks

[Matusov 09]. As stated in [Ney 99], a combination of recognition and translation model scores provided with a joint optimal scaling factor can lead to better translation quality. This combination is done by including word posterior probabilities [Bertoldi & Zens$^+$ 07] or using acoustic and source language model scores with word- and phrase-based translation scores in the log-linear framework [Shen & Delaney$^+$ 07, Matusov & Ney$^+$ 05] with an extension of the approach with phrase-level reordering [Matusov & Hoffmeister$^+$ 08]. To make long-range reordering possible, monotonic translation of word lattices [Casacuberta & Llorens$^+$ 01] or a very restricted word and phrase reordering [Zens & Bender$^+$ 05] can be conducted.

In the same line of motivation, the work by [Bertoldi & Zens$^+$ 07] introduces the translation of confusion networks to counter the inefficiency of word lattice reordering. Due to confusion network structures in which the recognized words are aligned to specific positions, reordering is no longer impossible with significant improvements [Bertoldi & Zens$^+$ 07, Besacier & Mahdhaoii$^+$ 07]. It is also shown that defining coverage vectors for MT search in terms of topologically ordered lattice states makes lattice reordering possible [Dyer & Muresan$^+$ 08]. Given that, the system combination of confusion network-based translation models is done to further improve the speech translation quality [Matusov & Ueffing$^+$ 06].

As stated before, in the cascade system, the translation performance is degraded by recognition errors. Therefore, training the ASR and MT parameters of the log-linear model in phrase-based translation systems can be done to optimize translation scores of the full speech translation pipeline [He & Deng$^+$ 11]. Such modeling provides a selection of recognition candidates that can be more easily translated, leading to better translation performance. In [Ohgushi & Neubig$^+$ 13], the authors further explore the possibility of joint optimization of ASR and MT models.

The research direction is not limited to the integration of recognition and translation systems. Cascade systems also encounter some problems associated with interfacing ASR outputs and MT inputs because the output of the recognizer is not a well-structured sentence with punctuation. To list a few interfaces between ASR and MT, we can optionally carry out word segmentation [Godard & Boito$^+$ 18], sentence segmentation [Fügen & Kolss 07, Fügen & Kolss 07], punctuation prediction [Lee & Roukos 04, Matusov & Mauser$^+$ 06], disfluency removal [Fitzgerald & Hall$^+$ 09], and domain adaptation [Liu & Gu$^+$ 03] to make the recognizer's outputs as close as possible to the translator's input.

Instead of augmenting source transcripts, we can also adapt the source side of machine translation systems to look like speech recognizers' outputs. One way is to develop an intermediate step that matches two components by a third module which classifies based on ASR-generated and ground-truth transcriptions [Dixon & Finch$^+$ 11]. Several studies further revisit similar ideas and propose the robustness of translation models against noisy inputs. In [Peitz & Wiesler$^+$ 12], the authors insert synthetic asr-like errors that might occur during transcribing utterances. This is close to our asr-like preprocessing. A phone-to-word phrase table is used to translate a sequence of phonemes generated by a recognizer to a sequence of words that is used as the source of MT data. One can further add, remove or shift words in source sentences of MT training to simulate ASR errors [Sperber & Neubig$^+$ 17] or introduce a noise model on the source side of MT training and experimentally illustrate that a certain amount of generative noise in training can be induced to the MT model [Sperber & Niehues$^+$ 17]. Similarly, it is shown that it is beneficial to include noisy data for MT training [Gangi & Enyedi$^+$ 19].

In addition to improving the interface between ASR and MT systems, the success of deep neural networks in both machine translation and automatic speech recognition has inspired the work of end-to-end speech translation. This motivates the research goal of the end-to-end models to avoid some of the problems associated with cascade systems. End-to-end stand-alone direct models translate speech in the source language directly into the target language text using RNN attention models [Berard & Pietquin$^+$ 16, Bansal & Kamper$^+$ 17, Berard & Besacier$^+$ 18] or using self-attentive networks [Gangi & Negri$^+$ 19a, Gangi & Negri$^+$ 19c]. Despite the benefits of direct models over cascade systems, their training requires a moderate amount of paired speech-to-target data, which is not easy to acquire. Therefore, these models are subject to overfitting.

Some techniques have been proposed to mitigate the lack of speech-to-target parallel training data. These methods aim to use weakly supervised data, i.e., speech-to-source or source-to-target pairs, in addition to fully supervised data, i.e., speech-to-target. As reported in the literature, all of these methods give a boost to some degree; however, each has its own problems. Multi-task learning aims at improving the generalization performance of a task using other related tasks as co-trainers [Luong & Le$^+$ 16]. Co-trainers can be text translation [Liu & Xiong$^+$ 19] or speech recognition [Weiss & Chorowski$^+$ 17, Anastasopoulos & Chiang 18, Sperber & Neubig$^+$ 19a]. A common way of mitigating the weakly supervised data is to pretrain different components of the network. The component of the direct models can be initialized using automatic speech recognition [Bansal & Kamper$^+$ 18, Bansal & Kamper$^+$ 19, Stoian & Bansal$^+$ 20] or speech recognition and text translation [Berard & Besacier$^+$ 18] model parameters.

In the absence of an adequate volume of training data, one remedy is to generate synthetic data like back-translation [Sennrich & Haddow$^+$ 16a], which is the most common method to leverage monolingual data in machine translation. The idea is to use a pretrained model to convert weakly supervised data into speech-to-target pairs for direct modeling. In [Jia & Johnson$^+$ 19], the authors apply a pretrained source-to-target MT model that translates ASR transcripts into target texts. Furthermore, they use a pretrained text-to-speech model to synthesize speech data from a bilingual text. The data scarcity problem by generating synthetic data from unlabeled audio is also addressed using an ST engine [Pino & Xu$^+$ 20].

Although the pretraining scheme and generating synthetic data usually seem to be effective, they rely on previously trained models, which in some cases are expensive to obtain. Furthermore, paired speech-to-source, source-to-target, or both are not always available. An unsupervised speech translation system, in contrast, uses only independent monolingual corpora of speech and text, even though its performance is behind the aforementioned approaches [Chung & Weng$^+$ 19].

Another method is data augmentation, which generates new synthetic training examples by corrupting the initial speech data and conserving the same target sentence as the original training sample. Audio-level speech augmentation can be done in different ways, such as noise injection (adding random noise), shifting time (transmitting time series forward/backward with a few seconds), speed perturbation (expanding time series by a speed rate), and changing the frequency pitch randomly. Besides increasing the quantity of training data, data augmentation often makes the model invariant to the applied noise and enhances its generalization. Inspired by the augmentation methods in ASR [Ko & Peddinti$^+$ 15, Park & Chan$^+$ 19, Nguyen & Stüker$^+$ 20] as a remedy to prevent overfitting, some efforts study the use of data augmentation in direct speech translation. Similar to our work on data augmentation, other works also improve translation performance with SpecAugment [Gangi & Negri$^+$ 19b, McCarthy & Puzon$^+$ 20]. The latter proposes a converter model to resemble another speaker's voice, resulting in a different audio sequence, thus better generalization and Bleu scores. The work by [Gaido & Gangi$^+$ 20] trains the model on suboptimal segments either by fine-tuning or considering more context. Their method can be seen as an augmentation approach that also solves the problem of badly-formed segmentation. They show that the model becomes more robust to randomly segmented data.

Other research direction applies multilingual speech translation that allows the exploitation of additional data [Gangi & Negri$^+$ 19d, Inaguma & Duh$^+$ 19, Wang & Pino$^+$ 20].

In realistic data conditions where the cascade and end-to-end models are trained on non-equal amounts of data, the performance of end-to-end, and in particular direct models is often behind cascade systems. Being akin to cascade modeling but being optimized in an end-to-end fashion, the idea of two-stage models is to provide source supervision by inducing transcripts. Since transcripts as discrete representations are problematic for end-to-end training via back-propagation, two-step models pass on hidden representations like decoder states or attention vectors from the ASR to the MT decoder [Anastasopoulos & Chiang 18, Sung & Liu$^+$ 19, Sperber & Neubig$^+$ 19a]. The two-step models are driven by the same motivation as our tight integration cascade model.

The end-to-end methods either translate without transcribing or suffer from inconsistency between transcriptions and translations [Sperber & Setiawan$^+$ 20]. In contrast, our tight integration cascade model generates both speech transcripts and translated texts with a high consistency between them. The work in [Osamura & Kano$^+$ 18] is the closest model to ours. The authors employ an ASR model to produce word posteriors and train a translation model given the obtained probabilistic features. They do not use renormalized posteriors and need to modify the MT architecture to adapt it to the ASR posteriors, whereas our model collapses both models side by side without any need for network modification. For more on speech translation developments, we refer the reader to [Sperber & Paulik 20], which offers a brief survey of speech translation methods.

7. Scientific Achievements

As stated in Chapter 2, this work concentrated on rethinking the design of the attention-based sequence-to-sequence models with the overall goal of establishing alternative models or advancing the existing models in one or more aspects.

Attention Refinements

As one of the first goals, we verified results from some extensions to the recurrent-based attention model from the literature and further refined the model. We explored different ways to incorporate information from the previous output positions into the attention computation of the current step. After comparing the performance of the standard and extended RNN attention models, we observed that fertility with past alignment information marginally improves performance in machine translation (Table 3.4). We also applied the same idea in attention-based automatic speech recognition and showed some gains over the standard RNN attention model (Table 3.11). These results encouraged us to include fertility with past alignments in the recurrent-based attention baseline. However, the self-attentive model still outperforms the recurrent baseline in machine translation (Tables 3.5 to 3.7) and is on par with it in automatic speech recognition if a sufficient amount of training data is utilized (Table 3.13).

Two-Dimensional Sequence-to-Sequence Modeling

We contributed to the advancement of sequence-to-sequence modeling and proposed a novel 2D recurrent-based sequence-to-sequence model, called 2D RNN, that establishes a 2D structure to define the correspondence between input and output sequences. With this joint representation, the input and output sequences meet at the final stage of encoding, allowing each of them to affect the representation of the other. Despite the fact that many new challenges emerge when introducing new stand-alone models, which then require finding the best design choice, we successfully applied the model to machine translation and automatic speech recognition. The 2D RNN models are comparable in performance with the RNN attention model, but behind the self-attentive models in machine translation (Tables 4.4 to 4.6) on average, and they are competitive in speech recognition (Tables 4.14 and 4.15). We also investigated the effect of bidirectional 2D RNN models where we concatenated the forward and backward 2DLSTM layers and found that there is no real benefit to using them (Table 4.1 for machine translation and Table 4.11 for automatic speech recognition). We further studied the contribution of different models in translating long sentences. 2D modeling shows better generalization ability over unseen sequence lengths compared to the attention-based models (Figure 4.9). However, adding relative positional encoding to the self-attentive model improves its performance on long sequences. Furthermore, we extended the concept of 2D sequence-to-sequence modeling to two-way translation. We translated both source-to-target and target-to-source directions using a single model, and showed some initial results.

Zero-order Hidden Markov Modeling

We reformulated the attention models and incorporated them into the direct HMM formulation by introducing a latent variable as an alignment sequence. We developed end-to-end models with tractable zero-order dependency on the alignments using the recurrent and self-attentive networks without any external alignment information. We firstly explored a speedup technique that increases training and decoding speed by a factor of three without loss of performance. Compared zero-order HMM with the RNN attention and self-attentive models, we demonstrated that its performance is behind the attention-based model on machine translation tasks. We observed that adding the attention context vector makes the model comparable with the recurrent-based models (Tables 5.5 to 5.7). However, the model showed a weaker performance on speech recognition tasks and has not yet reached the performance of the baseline models (Tables 5.14 and 5.15). Computed the alignment error rate of the models on golden alignments, we quantitatively found out that the zero-order HMM model generates slightly more focused alignments than the attention models (Table 5.9). Qualitative visualization of the alignment distribution of different models also confirmed our findings concerning better explainability in machine translation and speech recognition. Although alignments are of higher quality both qualitatively and quantitatively in zero-order HMM than the attention-based models, this improvement does not transfer to the final translation or recognition performance.

Development of End-to-End Models in the Context of Speech Translation

We examined the combination of CTC with the recurrent attention-based model and proposed the use of SpecAugment for speech translation, resulting in a practical training procedure. The CTC loss significantly improves the translation performance with faster training convergence (Tables 6.5 and 6.6), and the SpecAugment method also achieves remarkable gains with better generalization (Tables 6.7 and 6.8) in comparison with the recurrent attention baseline. We further investigated and developed various methods to exploit all types of relevant available training data. Sequential transfer learning shows significant improvement and eases the data scarcity issue of the direct models (Tables 6.9 to 6.11). However, the multi-task scenarios result in a suboptimal solution for the whole optimization problem (Table 6.12). Moreover, we proposed a tighter coupling between speech recognition and text translation. In this context, the tight integration cascade model is naturally decomposed into two steps. This is akin to cascade systems but preserves end-to-end trainability and reduces the early-decision problem. The experimental results show that the tight integration cascade model outperforms all other approaches under realistic data conditions (Tables 6.14 to 6.18), is much more effective at exploiting auxiliary data, and probably provides consistency between transcripts and translations due to its two-step nature. We illustrated that the end-to-end models can realistically translate speech utterances as a substitute solution to cascaded speech translation since they are able to compete with or outperform them.

8. Individual Contributions

According to §5.6 of the doctoral guidelines of RWTH Aachen University, Department of Computer Science, September 11, 2018, this chapter highlights the author's individual contributions as opposed to teamwork with respect to previously published materials, summarizing Chapters 3 to 6 in one place. We note that for all papers in which Parnia Bahar is the first author, she proposed the main idea and wrote the paper, typically with feedback from her co-authors, without indicating so for each paper. We also mention here that the models described in Sections 3.5.3, 4.3, and 5.2 were initially implemented in our Blocks software based on the Theano framework, and then were reimplemented in RETURNN to support more features in the future, as Theano development was stopped. All experiments reported in this work used the RETURNN toolkit with guidance for the implementation from Albert Zeyer.

The attention refinements in Section 3.5.3 were motivated by the works in the literature. They were initially implemented in the scope of two Bachelor's theses supervised by Jan-Thorsten Peter. After stopping Theano development, all implementations were adapted by Albert Zeyer in RETURNN to improve the recurrent attention models. The ease of network configuration in RETURNN allowed the author to explore further extensions. The author was also involved in integrating the self-attentive model in RETURNN where she verified the implementation done by Albert Zeyer and Julian Schamper. In addition, she contributed to the self-attentive model training and experimental results for speech recognition published in [Zeyer & Bahar^{+} 19] together with Albert Zeyer. All authors wrote the paper together. The i6 colleagues have taken the self-attentive configurations and the attention refinements for further research and development.

The 2D sequence-to-sequence modeling in Section 4.3 was proposed by the author. The first attempt to create a GPU-based implementation of 2DLSTM was written directly using Theano by the author. The core implementation of 2DLSTM used in this work is based on CUDA in C++, which was written by Paul Voigtlaender [Voigtlaender & Doetsch^{+} 16]. The CUDA implementation is faster than that in Theano, while it maintains the correctness of the gradient computation. However, this version of implementation is suitable for images that are inherently 2D. Christopher Brix adapted and reimplemented the 2DLSTM kernel function to ensure its usage for sequence modeling. He also used the kernel as an attention component between the encoder and decoder during his Bachelor's thesis [Brix 18], which the author of this dissertation supervised. The author implemented the different collapsing techniques described in Section 4.3.1 based on the initial implementation by Christopher Brix. The author developed and designed the stand-alone 2D RNN model and conducted all experiments in machine translation [Bahar & Brix^{+} 18] and automatic speech recognition [Bahar & Zeyer^{+} 19a]. Albert Zeyer assisted with discussions on the experimental design and with data preparation for the speech recognition experiments. The success of the 2D RNN model motivated the author, and she further contributed the idea and model design of using a single two-dimensional layout for two-way bidirectional translation models, i.e., source-to-target and target-to-source, as addressed in Section 4.4. To enable the two-way translation model, Christopher Brix extended some parts of the code with the author, who did further developments. The two-way translation model was published in [Bahar & Brix$^+$ 21], in which the experiments were done by the author together with Christopher Brix.

Inspired by internal works of the higher-order hidden Markov models for machine translation [Alkhouli & Bretschner$^+$ 16, Alkhouli & Ney 17, Wang & Alkhouli$^+$ 17, Wang & Zhu$^+$ 18], the author began working on zero-order HMM and its integration to the attention model as explained in Chapter 5. The author contributed the idea and the design of efficient algorithms to make it tractable in a reasonable amount of time and published the findings for machine translation in [Bahar & Makarov$^+$ 20a]. Nikita Makarov implemented the zero-order HMM model in RETURNN as part of his Bachelor's thesis under the author's supervision. The author suggested the usage of various speedup techniques and hence implemented the topK approximation and other refinements together with the student. Nikita Makarov only experimented with the self-attentive models, while the recurrent-based zero-order models were the contributions of the author. Nikita Makarov also helped write the paper. The author had discussions on neural HMM models for machine translation with Tamer Alkhouli and Weiyue Wang. She further extended the idea to speech recognition and made the necessary modifications to the model and its training procedure as published in [Bahar & Makarov$^+$ 20b]. The author designed the experimental setups based on Albert Zeyer's well-established baselines. Nikita Makarov and the author tried different extensions to the model that are not included in this dissertation but can be found in [Makarov 19].

For Chapter 6, the author initiated a joint work with the colleagues at Application Technology (AppTek): Evgeny Matusov, Patrick Wilken, Julian Schamper, Pavel Golik, and Albert Zeyer [Matusov & Wilken$^+$ 18]. The author later verified and established different end-to-end methods from the literature with the help of Tobias Bieschke, who conducted some of the experiments during his student work with the author's supervision. Some of the findings were published in [Bahar & Bieschke$^+$ 19]. The author motivated the exploration of all discussed methods, configured the network structures, borrowed promising methods from speech recognition, and established best practices for direct modeling. She also applied spectrogram augmentation for speech translation [Bahar & Zeyer$^+$ 19b] by adapting it from recognition experiments by Albert Zeyer. The author further developed the idea of the tight integration cascade model, proposed a practical solution for its training and decoding, and published the results in [Bahar & Bieschke$^+$ 21]. Tobias Bieschke carried out the initial implementation of the search for two-step modeling during his Master's thesis. The author further implemented a feature for jointly generating transcripts and translations in search through a single forward pass. In this dissertation, the hybrid HMM automatic speech recognition model for the cascade system was trained by Pavel Golik, whereas the author trained all the attention-based recognition models.

Apart from these contributions to the three tasks of automatic speech recognition, text and speech translation, some other achievements were attained during the process of this work which are not elaborated on in this dissertation. Among those achievements are the empirical investigation of different optimization techniques and proposals for a practical recipe for better convergence [Bahar & Alkhouli$^+$ 17] and model sparsity for text translation [Brix & Bahar$^+$ 20], as well as teaching activities, supervision of students, collaboration on industrial projects, and participation in several international evaluation campaigns. Jan-Thorsten Peter organized the participation in IWSLT 2015 [Peter & Toutounchi$^+$ 15] and WMT 2017 [Peter & Guta$^+$ 17] evaluations. In the former, the author trained an attention model and employed it in the N-best list rescoring of phrase-based systems. She also implemented unknown target tokens replacement using IBM-1 after search. In the latter system paper, she helped with the attention model training. In both papers, the author contributed to writing the text. The participation in the IWSLT 2017 evaluation was organized by the author jointly with Jan Rosendahl and Nick Rossenbach for the German$\leftrightarrow$English translation tasks [Bahar & Rosendahl$^+$ 17]. The experiments were carried out together with the co-authors of the paper. The author contributed to the WMT 2018 evaluation participation and wrote the paper with her co-authors [Schamper & Rosendahl$^+$ 18]. The author was in charge of the

Chinese→English task. For the IWSLT 2018 speech translation participation, the author assisted with machine translation training of the cascade models, carried out different ensemble schemes, helped end-to-end direct model training, and contributed to writing the paper [Matusov & Wilken$^+$ 18]. The WMT 2019 German→English evaluation was prepared and built by Jan Rosendahl, Christian Herold, and the author [Rosendahl & Herold$^+$ 19]. All authors wrote the paper together. The combination of methods presented in Chapter 6 culminated in the submissions to IWSLT 2020 that achieved high-ranking positions [Bahar & Wilken$^+$ 20]. The author conducted all experiments for direct speech translation. The ensemble of different cascade models created by the author of this thesis placed first out of ten submissions for offline speech translation and first out of eight systems for the non-native speech translation track. The direct speech translation system built by the author also ranked second out of eight submissions in the end-to-end track. More details can be found in [Ansari & Axelrod$^+$ 20]. This achievement led to further developments of end-to-end speech translation systems and showed the impact and continued usage of the author's contributions to RWTH Aachen University state-of-the-art speech translation system. The developed end-to-end speech translation models have motivated external teams and have been taken up by some of them.

A. Overview of the Corpora

A.1 Machine Translation

The workshop on statistical machine translation (WMT)[1] is an annual evaluation campaign that provides training and test data for a wide range of language pairs. Its main focus is the translation of news texts.

A.1.1 WMT 2018 German→English

Table A.1 lists the statistics for the training, development, and test sets of the WMT 2018 German→English task[2]. The bilingual data contains the standard parallel WMT 2018 corpora, i.e., transcribed European parliament speeches (`Europarl v7`), data crawled from the web (`Common Crawl`), news commentaries (`News Commentary v12`) and EU press releases (`Rapid`), in total 5.9M sentence pairs. `newstest2015` is used for parameter tuning and `newstest2017` and `-2018` as the blind test sets. Using the Moses toolkit [Koehn & Hoang$^+$ 07], on both German and English sides, we remove sentence pairs with a large difference in their length, apply a simple preprocessing pipeline consisting of minor text normalization steps (e.g., removal of some special UTF-8 characters) followed by tokenization and true-casing from Moses. BPE segments words into subword units. The BPE models are trained jointly for German and English on the training data using 50k merge operations with a vocabulary threshold of 50 tokens to diminish the number of rare words. The detailed statistics of the training, development, and test sets on both the word and subword levels are shown in Table A.1.

A.1.2 WMT 2018 Chinese→English

The statistics for the training, development, and test sets of the Chinese→English task provided for WMT 2018 are given in Table A.2. The bilingual data contains `News Commentary v14`, `Wiki Titles v1`, `UN Parallel Corpus v1` and the news domain of the China workshop on machine translation `(CWMT)` with almost 26M sentence pairs. We use `newsdev2017` for parameter optimization and `newstest2017` and `-2018` as the unseen test sets. For this task, we first apply uniblock[3] [Gao & Wang$^+$ 19], an approach to filter sentences consisting of illegal characters on either side. After removing deduplication on both sides, which further removes around 5.8M sentence pairs, we select 17M sentence pairs as the training data.

On the Chinese side, the THULAC[4] segmenter [Li & Sun 09] is used. As a standard remedy, we apply normalization, tokenization, and true-casing from Moses on the English text. The BPE models are trained independently for Chinese and English on the training data using 32k symbols.

[1] `http://www.statmt.org/`

[2] `http://www.statmt.org/wmt18`

[3] `https://github.com/ringoreality/uniblock`

[4] `https://github.com/thunlp/THULAC-Python`

Table A.1: Corpus statistics for WMT 2018 German→English. A joint BPE with 50k merge operations is applied.

		German	English
`train`:	sentences	5920281	
	running words	137.3M	144.9M
	vocabulary words	2.2M	1.1M
	running subwords	160.1M	157.7M
	vocabulary subwords	45409	33522
`newstest2015 (dev)`:	sentences	2169	
	running words	44079	46831
	vocabulary words	9891	7871
	OOV words (rate)	1102 (2.4%)	562 (1.2%)
	running subwords	54841	53097
	vocabulary subwords	10458	8151
	OOV subwords (rate)	0 (0.0%)	0 (0.0%)
`newstest2017`:	sentences	3004	
	running words	60961	64765
	vocabulary words	12760	9416
	OOV words (rate)	1551 (2.54%)	684 (1.06%)
	running subwords	76342	73107
	vocabulary subwords	12576	9388
	OOV subwords (rate)	0 (0.0%)	0 (0.0%)
`newstest2018`:	sentences	2998	
	running words	64037	67527
	vocabulary words	13067	10079
	OOV words (rate)	1582 (2.47%)	791 (1.17%)
	running subwords	79361	76854
	vocabulary subwords	12979	9970
	OOV subwords (rate)	0 (0.0%)	0 (0.0%)

A.1.3 WMT 2014 English→German

The statistics for the training, development, and test sets of the WMT 2014 English→German task provided for the WMT 2014[5] are given in Table A.3. Similar to WMT German→English, the bilingual data also contains `Europarl v7`, `Common Crawl`, and `News Commentary v12`, in total 4.5M sentence pairs. We follow a practical recipe from the fairseq toolkit[6] [Ott & Edunov+ 19] and use a random subset of the training data as the development set for parameter optimization, `newstest2013` to pick the model checkpoint, and `newstest2014` as the blind test set. In contrast to WMT 2018 German→English, after tokenization, we do not apply true-casing and let the model itself learn it. A 40k joint BPE operation yields subword-level sentences.

A.2 Automatic Speech Recognition

A.2.1 Switchboard 300h

We conduct experiments on Switchboard-1 (LDC97S62) [Godfrey & Holliman+ 92]. This dataset consists of approximately 260 hours of English narrow-band 8kHz telephone conversations about

[5] `http://www.statmt.org/wmt14`
[6] `https://github.com/pytorch/fairseq`

Table A.2: Corpus statistics for WMT 2018 Chinese→English. A separate BPE with 32k merge operations is applied.

		Chinese	**English**
`train`:	sentences	17000000	
	running words	362M	396M
	vocabulary words	1255279	747594
	running subwords	372M	408M
	vocabulary subwords	46983	32159
`newsdev2017 (dev)`:	sentences	2002	
	running words	53678	59816
	vocabulary words	9182	8915
	OOV words (rate)	747 (1.5%)	349 (0.6%)
	running subwords	58701	65062
	vocabulary subwords	9151	9163
	OOV subwords (rate)	9 (0.0%)	17 (0.0%)
`newstest2017`:	sentences	2001	
	running words	50108	54904
	vocabulary words	8850	8245
	OOV words (rate)	956 (1.9%)	429 (0.8%)
	running subwords	55891	59696
	vocabulary subwords	8878	8470
	OOV subwords (rate)	8 (0.0%)	12 (0.0%)
`newstest2018`:	sentences	3981	
	running words	96670	113449
	vocabulary words	13384	12688
	OOV words (rate)	1582 (1.6%)	755 (0.7%)
	running subwords	106859	123016
	vocabulary subwords	12417	12286
	OOV subwords (rate)	12 (0.0%)	16 (0.0%)

various casual topics containing multiple speakers. A small portion of the training data is extracted as the cross-validation set to tune parameters. We use `Hub5'00` (LDC2002S09) to evaluate the model, consisting of `Switchboard` and `Callhome` parts. We report WER on both parts separately and their average (Σ). `Hub5'00` is used as the development set to select the best checkpoint, tune language model scales, etc., and `Hub5'01` (LDC2002S13) is considered as the test set.

Language Model

Language model training consists of Switchboard and Fisher parts of transcriptions resulting in 27M words. There is an overlap of 3M out of 27M words between the acoustic and language model training. The words are segmented into subword units using the same BPE operation as the acoustic model. We use both an RNN-based [Sundermeyer & Schlüter^{+} 12] and a self-attentive-based language model [Liu & Saleh^{+} 18, Al-Rfou & Choe^{+} 19] from [Irie 20]. The RNN-based model is composed of an embedding layer of size 256 followed by two LSTM layers with 2048 cells. A dropout of 0.2 is applied. It is trained using SGD, and gradient clipping [Zeyer & Irie^{+} 18]. The self-attentive-based language model consists of 30 layers with 8 heads each. The dimensions of the input embedding, self-attention, and feedforward layers are chosen to be 128, 512, and 2048, respectively. It is trained using the standard SGD update rule with a learning rate of 1, a decay factor of 0.9, and a gradient clipping of 1. More details can be found in [Zeyer & Bahar^{+} 19].

Table A.3: Corpus statistics for WMT 2014 English→German. The development set is a random subset of the training data. A joint BPE with 40k merge operations is applied.

		English	German
`train`:	sentences	3961179	
	running words	103M	98M
	vocabulary words	688k	1.4M
	running subwords	112M	115M
	vocabulary subwords	40356	42709
`dev`:	sentences	40058	
	running words	1048320	995714
	vocabulary words	53142	90108
	OOV words (rate)	3662 (0.3%)	8157 (0.8%)
	running subwords	1140227	1169686
	vocabulary subwords	23913	32001
	OOV subwords (rate)	38 (0.0%)	14 (0.0%)
`newstest2013`:	sentences	3000	
	running words	65599	64251
	vocabulary words	9552	12683
	OOV words (rate)	472 (0.7%)	943 (1.5%)
	running subwords	73637	78381
	vocabulary subwords	9410	12315
	OOV subwords (rate)	0 (0.0%)	0 (0.0%)
`newstest2014`:	sentences	3003	
	running words	68948	64506
	vocabulary words	10230	13818
	OOV words (rate)	727 (1.1%)	2202 (3.4%)
	running subwords	78182	81626
	vocabulary subwords	9848	12719
	OOV subwords (rate)	2 (0.0%)	768 (0.9%)

A.2.2 LibriSpeech 960h

LibriSpeech is a large-scale corpus of approximately 1000 hours of 16kHz read English speech prepared by [Panayotov & Chen$^+$ 15][7]. The data are derived from reading audio books from the LibriVox project and have been carefully segmented and aligned. The speakers in the corpus have been ranked according to WER of a model's transcripts and designated as `clean` and `other` for the more challenging (noisy) speech. Both the development and test sets are composed of both male and female speakers.

Language Model

In order to train the language models, a corpus is employed in conjunction with the LibriSpeech ASR corpus, including 800M words without overlap to the parallel acoustic data[8]. Two language models are used for this task, with more details in [Irie 20]. The RNN-LM is deeper than that for Switchboard, with 4 layers of LSTM with 2048 nodes each. The size of input embedding is set to 128. No dropout is applied. The self-attentive-based LM consists of 24 layers of 8-head self-attention with 1024 nodes. The feedforward hidden dimension is 4096 [Irie & Zeyer$^+$ 19].

[7] `http://www.openslr.org/12/`
[8] `https://www.openslr.org/11/`

A.3 Speech Translation

The international workshop on spoken language translation (IWSLT) is an annual evaluation campaign centering on the translation of technical lectures, the TED talks[9]. The campaign is particularly concerned with improving the translation of spoken and speech forms.

A.3.1 IWSLT English→German

IWSLT provides one of the oldest speech translation datasets and contains English transcripts of TED talks aligned at the sentence level and augmented with German translations. The statistics for the training, development, and test data of the IWSLT 2018 TED talks English→German speech translation task [Cho & Fünfer$^+$ 14] provided for IWSLT 2018[10] are given in Table A.4. We realized that the provided audio-to-source-sentence alignments of the TED talks are often not correct in any of the training, development, and test sets. As this can significantly degrade the performance of the audio encoder for the end-to-end speech translation approaches described in Chapter 6, we automatically recompute alignments by force-aligning each TED recording to its corresponding source sentences, and apply heuristics to overcome the problem of transcription gaps (speech segments without a translation in the parallel data) [Matusov & Wilken$^+$ 18]. In addition, we perform a manual revision of these new alignments on the development and test sets to ensure reliable evaluation results.

Table A.4: Corpus statistics for speech translation IWSLT 2018 English→German. A joint BPE of 20k symbols is applied.

		English	**German**
`train`:	segments	171121	
	duration[h]	272h	
	running words	2624952	2833952
	vocabulary	45289	113192
	running subwords	2787311	3351851
	vocabulary subwords	17909	19361
`TED dev2010`:	segments	888	
	duration[h]	1h11m	
	running Words	17467	19044
	vocabulary	3180	4058
	OOV words (rate)	224 (1.3%)	623 (3.3%)
	running subwords	18769	22676
	vocabulary subwords	3520	4726
	OOV subwords (rate)	13 (0.0%)	80 (0.4%)
`TED test2015`:	segments	1080	
	duration[h]	1h52m	
	running words	17977	19743
	vocabulary	3110	3971
	OOV words (rate)	210 (1.2%)	521 (2.6%)
	running subwords	19109	23004
	vocabulary subwords	3452	4630
	OOV subwords (rate)	8 (0.4%)	92 (0.4%)

[9] `https://www.ted.com/talks`
[10] `https://sites.google.com/site/iwsltevaluation2018/Lectures-task`

Table A.5: Corpus statistics for speech translation MuST-C English→Spanish. Separate BPE of 5k and 32k symbols are used on English and Spanish, respectively.

		English	Spanish
`train`:	segments	261794	
	duration[h]	496h	
	running words	6138743	5742761
	vocabulary	69457	112753
	running subwords	6181807	6154156
	vocabulary subwords	4934	30879
`dev`:	segments	1316	
	duration[h]	2h33m	
	running words	32194	30524
	vocabulary	3898	5135
	OOV words (rate)	154 (0.5%)	341 (1.1%)
	running subwords	31135	32560
	vocabulary subwords	3234	5412
	OOV subwords (rate)	0 (0.0%)	6 (0.0%)
`testHE`:	segments	600	
	duration[h]	1h15m	
	running words	14306	13551
	vocabulary	2601	3165
	OOV words (rate)	104 (0.7%)	151 (1.1%)
	running subwords	14587	14458
	vocabulary subwords	2465	3398
	OOV subwords (rate)	1 (0.0%)	6 (0.0%)
`testCOMMON`:	segments	2502	
	duration[h]	4h10m	
	running words	54344	49034
	vocabulary	5757	7708
	OOV words (rate)	354 (0.7%)	592 (1.2%)
	running subwords	54398	52572
	vocabulary subwords	3891	7773
	OOV subwords (rate)	0 (0.0%)	13 (0.0%)

Additional Machine Translation Data In line with the IWSLT 2018 evaluation campaign specifications, we use the `TED`, `OpenSubtitles2018` [Lison & Tiedemann 16] corpora, as well as the data provided by WMT 2018, i.e., (`Europarl`, `ParaCrawl`, `CommonCrawl`, `News Commentary`, and `Rapid`) as the potential training data for the machine translation system training, amounting to 65M lines of parallel sentence pairs. We then filter this data based on several heuristics. We follow a similar scheme to that described in [Matusov & Wilken$^+$ 18] to filter noisy segments with programming code and XML markup. Sentences with at least 3 and at most 80 words are kept. Those sentence pairs whose source and target lengths differ by a factor of 5 or more are also ignored. Doing so results in a total of 37.6M aligned sentence pairs with 556M and 615M words on the English and German sides, respectively. On the German text, we apply the standard simple preprocessing pipeline consisting of minor text normalization steps, tokenization, and true-casing using Moses, followed by 20k BPE segmentation with a vocabulary threshold of 50 tokens.

Additional Automatic Speech Recognition Data A total of 390 hours of transcribed speech from the `TED-LIUM v2` and `IWSLT speech translation` corpora are used where we exclude the black-listed talks. The `TED-LIUM` corpus is used to create the pronunciation lexicon. Altogether,

Table A.6: Corpus statistics for speech translation LibriSpeech English→French. A joint BPE of 5k symbols is applied.

		English	French
`train`:	segments	47271	
	duration[h]	100h	
	running words	960999	1091118
	vocabulary	33052	106310
	running subwords	1315889	1801264
	vocabulary subwords	3657	4170
`dev`:	segments	1071	
	duration[h]	2h	
	running words	18651	19226
	vocabulary	3680	6099
	OOV words (rate)	188 (1.0%)	1010 (5.3%)
	running subwords	25012	31098
	vocabulary subwords	2581	2835
	OOV subwords (rate)	0 (0.0%)	0 (0.0%)
`test`:	segments	2048	
	duration[h]	3h40m	
	running words	36336	37416
	vocabulary	5689	9963
	OOV words (rate)	475 (1.3%)	1855 (5.0%)
	running subwords	48651	60336
	vocabulary subwords	2926	3205
	OOV subwords (rate)	2 (0.0%)	2 (0.0%)

the ASR data contains 260K segments with approximately 5M subwords.

A.3.2 MuST-C English→Spanish

The data statistics for the training, development, and test sets of the MuST-C English→Spanish speech translation task [Gangi & Cattoni$^+$ 19][11] are given in Table A.5. The direct speech translation data includes around 496 hours of transcribed and translated speech data for training.

Additional Machine Translation Data For English→Spanish, we select a huge available bilingual dataset downloaded from the OPUS project [Tiedemann 12][12]. This contracts from the IWSLT TED task where there is a compulsion on allowed training data. The total bilingual machine translation data is composed of the `MuST-C`, `TED`, `News-Commentary`, `Europarl`, `Wikipedia`, `CommonCrawl`, `WMT-News`, `OpenSubtitles`, and `JRC-Acquis` corpora. A filtering approach based on sentence similarity reduces the size of the training data. After filtering, we obtain around 47.7M sentence pairs, which is equal to 520M and 477M tokens in English and Spanish, respectively. The large majority of MT parallel data comes from text sources and thus includes punctuation marks, digits, and special symbols. We apply additional preprocessing to the English side of the data to make it look like speech transcripts produced by the ASR system [Matusov & Wilken$^+$ 18].

Additional Automatic Speech Recognition Data To train ASR models, more data is taken, a total of approximately 2300 hours of transcribed speech including `ST-EuroParl` [Iranzo-Sánchez &

[11] `https://ict.fbk.eu/must-c/`
[12] `https://opus.nlpl.eu/`

Silvestre-Cerdà$^+$ 20][13], `How2` [Sanabria & Caglayan$^+$ 18][14], `MuST-C`, `TED-LIUM release3` [Rousseau & Deléglise$^+$ 14] (excluding the black-listed talks), `LibriSpeech`, `Mozilla Common Voice`[15], and `IWSLT TED`. Similar to IWSLT 2018, we apply an automatic re-alignment process to improve the quality of TED talk segmentation. We also fine-tune the models on the ASR part of the MuST-C English→Spanish speech translation data.

Additional Monolingual Data To train language models, we select 2.1B tokens where the in-domain text data can be acquired through publicly available data. The language models are trained using additional data (extracted from the `Opensubtitles`, `gigaword` and `WMT news` corpora) of 1.2B tokens where a small portion of it (50M tokens) corresponds to the transcription of 2300 hours of audio. This additional data is 24 times larger.

A.3.3 LibriSpeech English→French

The LibriSpeech English→French speech translation dataset [Berard & Besacier$^+$ 18] is an augmentation of its ASR corpus. It contains English utterances from audio books automatically aligned with French text, resulting in 100 hours of clean triple data. Table A.6 presents the corpus statistics of all available sets. In contrast to the previous tasks, we utilize neither additional ASR nor MT data and study LibriSpeech English→French as a low-resource task.

[13]`https://www.mllp.upv.es/europarl-st/`
[14]`https://github.com/srvk/how2-dataset`
[15]`https://voice.mozilla.org/en/datasets`

B. Additional Results

Chapter 4

This section provides the detailed results on length analysis in Chapter 4.

Table B.1 shows the performance difference of the models with respect to source sequence lengths on WMT German→English. The sentences are obtained from the concatenation of `newstest2008-2018`. Sentence pairs are grouped based on the source length J with intervals of 25 subwords each. The number of sentences per bin is given by $|G|$.

Table B.1: Performance difference of the models with respect to source sequence lengths on WMT 2018 German→English.

model	**max length in training**	$J \leq 25$		$26 \leq J \leq 50$		$51 \leq J \leq 75$		$76 \leq J \leq 100$		$J \geq 101$	
		$\lvert G_{0:25}\rvert = 1070$		$\lvert G_{26:50}\rvert = 6333$		$\lvert G_{51:75}\rvert = 5084$		$\lvert G_{76:100}\rvert = 1880$		$\lvert G_{101:\mathrm{inf}}\rvert = 755$	
		Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	Bleu[%]	Ter[%]	Bleu[%]	Ter[%]
self-attentive	100	25.9	64.1	26.5	62.9	26.8	62.8	26.3	63.7	19.2	70.7
	75	26.0	63.9	26.6	62.6	26.6	62.9	22.4	67.2	14.1	75.3
	50	25.9	64.1	26.3	62.7	20.3	67.8	13.8	74.8	7.7	81.9
+ $d = 1024$	100	25.6	65.4	26.3	63.3	26.4	63.1	26.1	64.3	21.6	68.8
	75	25.9	64.4	26.6	62.8	27.0	62.5	25.3	63.8	15.7	73.1
	50	25.6	64.7	26.2	62.8	22.1	66.2	14.6	73.7	7.8	81.3
+ relative encoding	100	26.1	64.3	26.3	63.3	26.6	63.3	26.7	63.8	24.2	67.5
	75	25.8	63.8	26.4	62.8	26.6	62.8	26.2	63.6	23.2	66.9
	50	25.5	64.3	26.1	62.9	25.8	63.0	25.0	63.8	21.5	67.3
RNN attention	100	25.4	63.3	26.0	62.5	26.1	63.0	25.8	63.7	23.0	68.2
	75	25.1	63.5	26.0	62.5	26.2	62.8	25.5	64.0	23.1	67.5
	50	25.2	63.7	26.0	62.3	25.5	62.6	21.3	66.6	14.5	73.9
2D RNN	100	24.9	65.0	25.4	64.4	25.7	65.0	26.0	64.8	24.0	68.1
	75	24.8	65.0	25.1	64.6	25.2	65.1	24.9	66.1	22.1	69.1
	50	24.5	65.5	25.1	64.3	24.9	64.7	25.0	65.2	21.9	69.8

Chapter 6

This section is reserved for detailed secondary results for different speech translation tasks in Chapter 6.

IWSLT English→German

Automatic Speech Recognition Results Table B.2 shows the error rates of ASR models for IWSLT 2018 English→German. The hybrid HMM model with a 4-gram language model is on par with the attention-based model. The count-based LM has the OOV rate of 0.2% on `dev2010` and 0.5% on `test2015`. The language models' perplexities are 133 and 122 on `dev2010` and `test2015`, respectively. It confirms that a carefully trained attention model can substitute the hybrid HMM model. This observation along with the fact that hybrid HMM are out of the scope of this work persuaded us to continue only with the attention-based ASR model in the cascaded pipeline.

Table B.2: The word error rate of the ASR models on IWSLT English→German.

AM	LM	WER[%]		#
		dev2010	**test2015**	**parameters**
hybrid HMM [Matusov & Wilken+ 18]	4-gram	11.1	10.7	26M
RNN attention	none	11.2	10.6	227M

MuST-C English→Spanish

Automatic Speech Recognition Results For the MuST-C English→Spanish speech translation task, we train self-attentive in addition to the RNN model. Table B.3 compares the results in terms of WER. Since we utilize a large amount of the ASR data (approx 2300h), the self-attentive model performs as strong as the recurrent model on average (lines 1 and 4).

Table B.3: The word error rate of ASR models on MuST-C English→Spanish.

AM	LM	WER[%]		#
		test-HE	**test-COMMON**	**parameters**
RNN attention	none	7.3	10.4	162M
	RNN	6.3	9.5	+130M
	self-attentive	6.0	9.1	+308M
self-attentive	none	7.5	10.5	85M
	RNN	6.2	9.5	+130M
	self-attentive	5.9	9.4	+308M

For this task, we also train two language models: LSTM with 4 layers of size 2048 each and a 24-layer 8-head self-attention LM with 1024 nodes. The details of the LM data (including 2.1B tokens) are discussed in Section A.3.2. The language model score is combined with that of the acoustic attention-based model as in Equation 3.72 in decoding. Table B.4 indicates that the deep self-attentive model has lower perplexity, thus it leads to relatively lower WER. The use of external language models, as presented in Table B.3 directly affects the performance of the ASR models.

Table B.4: The language model perplexities for MuST-C English→Spanish.

LM	**test-HE**	**test-COMMON**	**# parameters**
RNN	45.4	46.2	130M
self-attentive	35.3	35.9	308M

Cascade Speech Translation Results As reported in Table B.3, the source language models reduce the word error rates for the MuST-C task. Hence, we integrate them on the ASR side of the cascade system and show the results in Table B.5.

Table B.5: Performance of the ST cascade systems on MuST-C English→Spanish.

ASR	**MT**	**LM**	**test-HE**		**test-COMMON**	
			BLEU [%]	TER [%]	BLEU [%]	TER [%]
RNN	RNN	none	37.6	46.5	30.2	55.4
		RNN	38.5	45.8	31.2	54.3
		self-attentive	38.9	45.3	31.3	54.3
	self-attentive	none	37.9	46.5	31.0	54.8
		RNN	39.0	45.7	32.1	53.6
		self-attentive	39.5	45.4	32.2	53.6
self-attentive	RNN	none	37.1	47.1	30.2	55.1
		RNN	38.6	45.6	31.2	54.2
		self-attentive	38.7	45.3	31.2	54.5
	self-attentive	none	37.5	47.4	31.2	54.7
		RNN	38.7	46.3	32.2	53.3
		self-attentive	39.0	46.0	31.5	55.7

LibriSpeech English→French

Automatic Speech Recognition Results The word error rates on the LibriSpeech English→French sets in Table B.6 also indicate the reasonably good performance of the RNN attention model with 16.2% on the test set.

Table B.6: The word error rate of the ASR model on LibriSpeech English→French.

AM	**LM**	WER [%]		**#**
		dev	**test**	**parameters**
RNN attention	none	16.4	16.2	168M

List of Figures

List of Tables

Bibliography

[Abadi & Agarwal$^+$ 15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

[Aharoni & Goldberg 17] R. Aharoni, Y. Goldberg: Morphological Inflection Generation with Hard Monotonic Attention. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2004–2015, Vancouver, Canada, July - August 2017.

[Al-Rfou & Choe$^+$ 19] R. Al-Rfou, D. Choe, N. Constant, M. Guo, L. Jones: Character-Level Language Modeling with Deeper Self-Attention. In *Conference on Artificial Intelligence (AAAI)*, pp. 3159–3166, Honolulu, HI, USA, January - February 2019.

[Alkhouli 20] T.A.N. Alkhouli: *Alignment-based Neural Networks for Machine Translation.* Ph.D. thesis, RWTH Aachen University, 2020.

[Alkhouli & Bretschner$^+$ 16] T. Alkhouli, G. Bretschner, J. Peter, M. Hethnawi, A. Guta, H. Ney: Alignment-based Neural Machine Translation. In *Workshop on Statistical Machine Translation (WMT)*, pp. 54–65, Berlin, Germany, August 2016.

[Alkhouli & Bretschner$^+$ 18] T. Alkhouli, G. Bretschner, H. Ney: On the Alignment Problem in Multi-Head Attention-based Neural Machine Translation. In *Workshop on Statistical Machine Translation (WMT)*, pp. 177–185, Belgium, Brussels, October - November 2018.

[Alkhouli & Ney 17] T. Alkhouli, H. Ney: Biasing Attention-based Recurrent Neural Networks using External Alignment Information. In *Workshop on Statistical Machine Translation (WMT)*, pp. 108–117, Copenhagen, Denmark, September 2017.

[Allen 87] R.B. Allen: Several Studies on Natural Language and Back-Propagation. In *IEEE International Conference on Neural Networks*, San Diego, CA, USA, June 1987.

[Anastasopoulos & Chiang 18] A. Anastasopoulos, D. Chiang: Tied Multitask Learning for Neural Speech Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 82–91, New Orleans, LA, USA, June 2018.

[Ansari & Axelrod$^+$ 20] E. Ansari, A. Axelrod, N. Bach, O. Bojar, R. Cattoni, F. Dalvi, N. Durrani, M. Federico, C. Federmann, J. Gu, F. Huang, K. Knight, X. Ma, A. Nagesh, M. Negri, J. Niehues, J. Pino, E. Salesky, X. Shi, S. Stüker, M. Turchi, A. Waibel, C. Wang: Findings of the IWSLT

2020 Evaluation Campaign. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 1–34, Seattle, WA, USA, July 2020.

[Arivazhagan & Cherry$^+$ 19] N. Arivazhagan, C. Cherry, W. Macherey, C. Chiu, S. Yavuz, R. Pang, W. Li, C. Raffel: Monotonic Infinite Lookback Attention for Simultaneous Machine Translation. In A. Korhonen, D.R. Traum, L. Màrquez, editors, *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1313–1323, Florence, Italy, July - August 2019.

[Auli & Galley$^+$ 13] M. Auli, M. Galley, C. Quirk, G. Zweig: Joint Language and Translation Modeling with Recurrent Neural Networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1044–1054, Seattle, WA, USA, October 2013.

[Auli & Gao 14] M. Auli, J. Gao: Decoder Integration and Expected BLEU Training for Recurrent Neural Network Language Models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 136–142, Baltimore, MD, USA, June 2014.

[Ba & Kiros$^+$ 16] L.J. Ba, J.R. Kiros, G.E. Hinton: Layer Normalization. *CoRR*, Vol. abs/1607.06450, 2016.

[Bahar & Alkhouli$^+$ 17] P. Bahar, T. Alkhouli, J.T. Peter, C.J.S. Brix, H. Ney: Empirical Investigation of Optimization Algorithms in Neural Machine Translation. *The Prague bulletin of mathematical linguistics*, Vol. 108, pp. 13–26, May 2017.

[Bahar & Bieschke$^+$ 19] P. Bahar, T. Bieschke, H. Ney: A Comparative Study on End-to-End Speech to Text Translation. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 792–799, Sentosa, Singapore, December 2019.

[Bahar & Bieschke$^+$ 21] P. Bahar, T. Bieschke, R. Schlueter, H. Ney: Tight Integrated End-to-End Training for Cascaded Speech Translation. In *IEEE Spoken Language Technology Workshop (SLT)*, pp. 950–957, Shenzhen, China, January 2021.

[Bahar & Brix$^+$ 18] P. Bahar, C. Brix, H. Ney: Towards Two-Dimensional Sequence to Sequence Model in Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3009–3015, Brussels, Belgium, November 2018.

[Bahar & Brix$^+$ 21] P. Bahar, C. Brix, H. Ney: Two-Way Neural Machine Translation: A Proof of Concept for Bidirectional Translation Modeling using a Two-Dimensional Grid. In *IEEE Spoken Language Technology Workshop (SLT)*, pp. 1065–1070, Shenzhen, China, January 2021.

[Bahar & Makarov$^+$ 20a] P. Bahar, N. Makarov, H. Ney: Investigation of Transformer-based Latent Attention Models for Neural Machine Translation. In *Association for Machine Translation in the Americas (AMTA)*, pp. 7–20, Orlando, FL, USA, September 2020.

[Bahar & Makarov$^+$ 20b] P. Bahar, N. Makarov, A. Zeyer, R. Schlüter, H. Ney: Exploring a Zero-Order Direct HMM based on Latent Attention for Automatic Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7854–7858, Barcelona, Spain, May 2020.

[Bahar & Rosendahl$^+$ 17] P. Bahar, J. Rosendahl, N. Rossenbach, H. Ney: The RWTH Aachen Machine Translation Systems for IWSLT 2017. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 29–34, Tokyo, Japan, December 2017.

[Bahar & Wilken$^+$ 20] P. Bahar, P. Wilken, T. Alkhouli, A. Guta, P. Golik, E. Matusov, C. Herold: Start-Before-End and End-to-End: Neural Speech Translation by AppTek and RWTH Aachen

University. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 44–54, Seattle, WA, USA, July 2020.

[Bahar & Zeyer$^+$ 19a] P. Bahar, A. Zeyer, R. Schlüter, H. Ney: On using 2D Sequence-to-Sequence Models for Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5671–5675, Brighton, UK, May 2019.

[Bahar & Zeyer$^+$ 19b] P. Bahar, A. Zeyer, R. Schlüter, H. Ney: On using SpecAugment for End-to-End Speech Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, China, November 2019.

[Bahdanau & Cho$^+$ 15] D. Bahdanau, K. Cho, Y. Bengio: Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[Bahdanau & Chorowski$^+$ 16] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio: End-to-End Attention-based Large Vocabulary Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4945–4949, Shanghai, China, March 2016.

[Bahl & Jelinek$^+$ 83] L.R. Bahl, F. Jelinek, R.L. Mercer: A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, No. 2, pp. 179–190, 1983.

[Baker 90] J.K. Baker: *Stochastic Modeling for Automatic Speech Understanding*, 297–307. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[Baldi & Pollastri 03] P. Baldi, G. Pollastri: The Principled Design of Large-Scale Recursive Neural Network Architectures–DAG-RNNs and the Protein Structure Prediction Problem. *Journal of Machine Learning Research*, Vol. 4, pp. 575–602, 2003.

[Bansal & Kamper$^+$ 17] S. Bansal, H. Kamper, A. Lopez, S. Goldwater: Towards Speech-to-Text Translation without Speech Recognition. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 474–479, Valencia, Spain, April 2017.

[Bansal & Kamper$^+$ 18] S. Bansal, H. Kamper, K. Livescu, A. Lopez, S. Goldwater: Low-Resource Speech-to-Text Translation. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1298–1302, Hyderabad, India, September 2018.

[Bansal & Kamper$^+$ 19] S. Bansal, H. Kamper, K. Livescu, A. Lopez, S. Goldwater: Pre-training on High-Resource Speech Recognition Improves Low-Resource Speech-to-Text Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 58–68, Minneapolis, MN, USA, June 2019.

[Baquero-Arnal & Iranzo-Sánchez$^+$ 19] P. Baquero-Arnal, J. Iranzo-Sánchez, J. Civera, A. Juan: The MLLP-UPV Spanish-Portuguese and Portuguese-Spanish Machine Translation Systems for WMT19 Similar Language Translation Task. In *Workshop on Statistical Machine Translation (WMT)*, pp. 179–184, Florence, Italy, August 2019.

[Bastien & Lamblin$^+$ 12] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I.J. Goodfellow, A. Bergeron, N. Bouchard, Y. Bengio: Theano: New Features and Speed Improvements, 2012.

[Battenberg & Chen$^+$ 17] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, A. Sriram, Z. Zhu: Exploring Neural Transducers for End-to-End Speech Recognition. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 206–213, Okinawa, Japan, December 2017.

[Baum & Eagon 67] L.E. Baum, J.A. Eagon: An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology. *Bulletin of the American Mathematical Society*, Vol. 73, pp. 360–363, 1967.

[Baxter 00] J. Baxter: A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, Vol. 12, pp. 149–198, 2000.

[Bayes 63] T. Bayes: An essay towards solving a problem in the doctrine of chances. *Philosophical Transaction of the Royal Society of London*, Vol. 53, pp. 370–418, 1763.

[Belinkov & Bisk 18] Y. Belinkov, Y. Bisk: Synthetic and Natural Noise Both Break Neural Machine Translation. In *International Conference on Learning Representations (ICLR)*, Vancouver, Canada, April - May 2018.

[Bellegarda 97] J.R. Bellegarda: A latent Semantic Analysis Framework for Large-Span Language Modeling. In *European Conference on Speech Communication and Technology (EUROSPEECH)*, Rhodes, Greece, September 1997.

[Bellman 54] R. Bellman: Some Applications of the Theory of Dynamic Programming - A Review. *Journal of the Operations Research Society of America*, Vol. 2, No. 3, pp. 275–288, 1954.

[Bengio & Ducharme$^+$ 03] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin: A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, Vol. 3, pp. 1137–1155, 2003.

[Bengio & Frasconi$^+$ 93] Y. Bengio, P. Frasconi, P.Y. Simard: The Problem of Learning Long-Term Dependencies in Recurrent Networks. In *International Conference on Neural Networks (ICNN)*, pp. 1183–1188, San Francisco, CA, USA, March - April 1993.

[Bengio & Schwenk$^+$ 06] Y. Bengio, H. Schwenk, J.S. Senécal, F. Morin, J.L. Gauvain: *Neural Probabilistic Language Models*, pp. 137–186. 2006.

[Bengio & Simard$^+$ 94] Y. Bengio, P.Y. Simard, P. Frasconi: Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 157–166, 1994.

[Berard & Besacier$^+$ 18] A. Berard, L. Besacier, A.C. Kocabiyikoglu, O. Pietquin: End-to-End Automatic Speech Translation of Audiobooks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 6224–6228, Calgary, Canada, April 2018.

[Berard & Pietquin$^+$ 16] A. Berard, O. Pietquin, C. Servan, L. Besacier: Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation. In *Workshop on End-to-End Learning for Speech and Audio Processing, Conference on Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain, December 2016.

[Bergstra & Breuleux$^+$ 10] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio: Theano: A CPU and GPU Math Expression Compiler. In *Python for Scientific Computing Conference (SciPy)*, June 2010.

[Bertoldi 05] N. Bertoldi: *Statistical Models and Search Algorithms for Machine Translation.* Ph.D. thesis, Universita degli Studi di, Trento, Italy, 2005.

[Bertoldi & Federico 05] N. Bertoldi, M. Federico: A New Decoder for Spoken Language Translation based on Confusion Networks. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 86–91, Cancun, Mexico, November - December 2005.

[Bertoldi & Zens$^+$ 07] N. Bertoldi, R. Zens, M. Federico: Speech Translation by Confusion Network Decoding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1297–1300, Honolulu, HI, USA, April 2007.

[Besacier & Mahdhaoii$^+$ 07] L. Besacier, A. Mahdhaoii, V.B. Le: The LIG Arabic/English Speech Translation System at IWSLT07. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 137–141, Trento, Italy, October 2007.

[Bishop 07] C.M. Bishop: *Pattern Recognition and Machine Learning, 5th Edition*. Information science and statistics. Springer, 2007.

[Bourlard & Morgan 93] H.A. Bourlard, N. Morgan: *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, USA, 1993.

[Bourlard & Wellekens 90] H. Bourlard, C. Wellekens: Links Between Markov Models and Multilayer Perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 12, pp. 1167–1178, 1990.

[Brix 18] C.J.S. Brix: Extension of the Attention Mechanism in Neural Machine Translation, 2018.

[Brix & Bahar$^+$ 20] C. Brix, P. Bahar, H. Ney: Successfully Applying the Stabilized Lottery Ticket Hypothesis to the Transformer Architecture. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3909–3915, Seattle, WA, USA, July 2020.

[Brown & Cocke$^+$ 88] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Rossin: A Statistical Approach to Language Translation. In *International Conference on Computational Linguistics (COLING)*, pp. 71–76, Buffalo, NY, USA, August 1988.

[Brown & Cocke$^+$ 90] P.F. Brown, J. Cocke, S.D. Pietra, V.J.D. Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Roossin: A Statistical Approach to Machine Translation. *Computational Linguistics*, Vol. 16, No. 2, pp. 79–85, 1990.

[Brown & Pietra$^+$ 93] P.F. Brown, S.D. Pietra, V.J.D. Pietra, R.L. Mercer: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, 1993.

[Bruguier & Zen$^+$ 18] A. Bruguier, H. Zen, A. Arkhangorodsky: Sequence-to-Sequence Neural Network Model with 2D Attention for Learning Japanese Pitch Accents. pp. 1284–1287, Hyderabad, India, September 2018.

[Caruana 93] R. Caruana: Multitask Learning: A Knowledge-based Source of Inductive Bias. In *Machine Learning, the Tenth International Conference*, pp. 41–48, Amherst, MA, USA, June 1993.

[Caruana 98] R. Caruana: Multitask Learning. In *Learning to Learn*, pp. 95–133. 1998.

[Casacuberta & Federico$^+$ 08] F. Casacuberta, M. Federico, H. Ney, E. Vidal: Recent Efforts in Spoken Language Translation. *IEEE Signal Processing Magazine*, Vol. 25, No. 3, pp. 80–88, 2008.

[Casacuberta & Llorens$^+$ 01] F. Casacuberta, D. Llorens, C.D. Martínez-Hinarejos, S. Molau, F. Nevado, H. Ney, M. Pastor, D. Picó, A. Sanchís, E. Vidal, J.M. Vilar: Speech-to-Speech Translation based on Finite-State Transducers. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 613–616, Salt Lake City, UT, USA, May 2001.

[Casacuberta & Ney$^+$ 04] F. Casacuberta, H. Ney, F.J. Och, E. Vidal, J.M. Vilar, S. Barrachina, I. García-Varea, D. Llorens, C.D. Martínez-Hinarejos, S. Molau: Some Approaches to Statistical and Finite-State Speech-to-Speech Translation. *Computational Speech Language*, Vol. 18, No. 1, pp. 25–47, 2004.

[Castano & Casacuberta$^+$ 97] M.A. Castano, F. Casacuberta, E. Vidal: Machine Translation using Neural Networks and Finite-State Models. Vol., 1997.

[Cauchy 47] A. Cauchy: Méthode Générale Pour la résolution des Systemes d'éQuations Simultanées. *Comp. Rend. Sci. Paris*, Vol. 25, No. 1847, pp. 536–538, 1847.

[Chan & Jaitly$^+$ 16] W. Chan, N. Jaitly, Q.V. Le, O. Vinyals: Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4960–4964, Shanghai, China, March 2016.

[Chang & Li$^+$ 17] S. Chang, B. Li, T.N. Sainath, G. Simko, C. Parada: Endpoint Detection using Grid Long Short-Term Memory Networks for Streaming Speech Recognition. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3812–3816, Stockholm, Sweden, August 2017.

[Chen & Firat$^+$ 18] M.X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G.F. Foster, L. Jones, M. Schuster, N. Shazeer, N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, M. Hughes: The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 76–86, Melbourne, Australia, July 2018.

[Chen & Goodman 99] S.F. Chen, J. Goodman: An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech and Language*, Vol. 13, No. 4, pp. 359–393, 1999.

[Chen & Matusov$^+$ 16] W. Chen, E. Matusov, S. Khadivi, J. Peter: Guided Alignment Training for Topic-Aware Neural Machine Translation. *CoRR*, Vol. abs/1607.01628, 2016.

[Cheng & Dong$^+$ 16] J. Cheng, L. Dong, M. Lapata: Long Short-Term Memory-Networks for Machine Reading. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 551–561, Austin, TX, USA, November 2016.

[Cheng & Shen$^+$ 16] Y. Cheng, S. Shen, Z. He, W. He, H. Wu, M. Sun, Y. Liu: Agreement-based Joint Training for Bidirectional Attention-based Neural Machine Translation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2761–2767, New York, NY, USA, July 2016.

[Chiu & Raffel 18] C. Chiu, C. Raffel: Monotonic Chunkwise Attention. In *International Conference on Learning Representations (ICLR)*, Vancouver, Canada, April - May 2018.

[Chiu & Sainath$^+$ 18] C. Chiu, T.N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R.J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, M. Bacchiani: State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4774–4778, Calgary, Canada, April 2018.

[Cho & Fünfer$^+$ 14] E. Cho, S. Fünfer, S. Stüker, A. Waibel: A Corpus of Spontaneous Speech in Lectures: The KIT Lecture Corpus for Spoken Language Processing and Translation. In *International Conference on Language Resources and Evaluation (LREC)*, pp. 1554–1559, Reykjavik, Iceland, May 2014.

[Cho & van Merrienboer$^+$ 14a] K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio: On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*, pp. 103–111, Doha, Qatar, October 2014.

[Cho & van Merrienboer$^+$ 14b] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014.

[Choi & Cho$^+$ 18] H. Choi, K. Cho, Y. Bengio: Fine-Grained Attention Mechanism for Neural Machine Translation. *Neurocomputing*, Vol. 284, pp. 171–176, 2018.

[Chorowski & Bahdanau$^+$ 15] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio: Attention-based Models for Speech Recognition. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 577–585, Montreal, Canada, December 2015.

[Chrisman 91] L. Chrisman: Learning Recursive Distributed Representations for Holistic Computation. *Connection Science*, Vol. 3, No. 4, pp. 345–366, 1991.

[Chung & Weng$^+$ 19] Y. Chung, W. Weng, S. Tong, J.R. Glass: Towards Unsupervised Speech-to-Text Translation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7170–7174, Brighton, UK, May 2019.

[Cohn & Hoang$^+$ 16] T. Cohn, C.D.V. Hoang, E. Vymolova, K. Yao, C. Dyer, G. Haffari: Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 876–885, San Diego, CA, USA, June 2016.

[Collobert & Weston 08] R. Collobert, J. Weston: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *International Conference on Machine Learning (ICML)*, Vol. 307, pp. 160–167, Helsinki, Finland, June 2008.

[Cybenko 89] G. Cybenko: Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, Vol. 2, No. 4, pp. 303–314, 1989.

[Dai & Yang$^+$ 19] Z. Dai, Z. Yang, Y. Yang, J.G. Carbonell, Q.V. Le, R. Salakhutdinov: Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2978–2988, Florence, Italy, July - August 2019.

[Davis & Mermelstein 80] S. Davis, P. Mermelstein: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 28, No. 4, pp. 357–366, 1980.

[Dempster & Laird$^+$ 77] A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B*, Vol. 39, pp. 1–38, 1977.

[Deng & Kim$^+$ 18] Y. Deng, Y. Kim, J. Chiu, D. Guo, A.M. Rush: Latent Alignment and Variational Attention. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 9735–9747, Montreal, Canada, December 2018.

[Devlin & Zbib$^+$ 14] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R.M. Schwartz, J. Makhoul: Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1370–1380, Baltimore, MD, USA, June 2014.

[Dixon & Finch$^+$ 11] P.R. Dixon, A.M. Finch, C. Hori, H. Kashioka: Investigation on the Effects of ASR Tuning on Speech Translation Performance. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 167–174, San Francisco, CA, USA, December 2011.

[Doetsch & Hannemann$^+$ 17] P. Doetsch, M. Hannemann, R. Schlüter, H. Ney: Inverted Alignments for End-to-End Automatic Speech Recognition. *IEEE Journal of Selected Topics in Signal Processing*, Vol. 11, No. 8, pp. 1265–1273, 2017.

[Doetsch & Hegselmann$^+$ 16] P. Doetsch, S. Hegselmann, R. Schlüter, H. Ney: Inverted HMM - a Proof of Concept. In *Workshop in End-to-end Learning for Speech and Audio Processing, Conference on Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain, December 2016.

[Doetsch & Zeyer$^+$ 17] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, H. Ney: Returnn: The RWTH Extensible Training Framework for Universal Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5345–5349, New Orleans, LA, USA, March 2017.

[Dong & Wu$^+$ 15] D. Dong, H. Wu, W. He, D. Yu, H. Wang: Multi-Task Learning for Multiple Language Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1723–1732, Beijing, China, July 2015.

[Dong & Zhou$^+$ 18] L. Dong, S. Zhou, W. Chen, B. Xu: Extending Recurrent Neural Aligner for Streaming End-to-End Speech Recognition in Mandarin. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 816–820, Hyderabad, India, September 2018.

[Dyer & Chahuneau$^+$ 13] C. Dyer, V. Chahuneau, N.A. Smith: A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 644–648, Atlanta, GA, USA, June 2013.

[Dyer & Muresan$^+$ 08] C. Dyer, S. Muresan, P. Resnik: Generalizing Word Lattice Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1012–1020, Columbus, OH, USA, June 2008.

[Edunov & Ott$^+$ 18] S. Edunov, M. Ott, M. Auli, D. Grangier, M. Ranzato: Classical Structured Prediction Losses for Sequence to Sequence Learning. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 355–364, New Orleans, LA, USA, June 2018.

[Elbayad & Besacier$^+$ 18] M. Elbayad, L. Besacier, J. Verbeek: Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction. In *Conference on Computational Natural Language Learning (CoNLL)*, pp. 97–107, Brussels, Belgium, October 2018.

[Feng & Liu$^+$ 16] S. Feng, S. Liu, N. Yang, M. Li, M. Zhou, K.Q. Zhu: Improving Attention Modeling with Implicit Distortion and Fertility for Machine Translation. In *International Conference on Computational Linguistics (COLING)*, pp. 3082–3092, Osaka, Japan, December 2016.

[Firat & Cho$^+$ 16] O. Firat, K. Cho, Y. Bengio: Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 866–875, San Diego, CA, USA, June 2016.

[Fitzgerald & Hall$^+$ 09] E. Fitzgerald, K.B. Hall, F. Jelinek: Reconstructing False Start Errors in Spontaneous Speech Text. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 255–263, Athens, Greece, March - April 2009.

[Forcada & Ñeco 97] M.L. Forcada, R.P. Ñeco: Recursive Hetero-Associative Memories for Translation. In *Biological and Artificial Computation: From Neuroscience to Technology, International Work-Conference on Artificial and Natural Neural Networks (IWANN)*, pp. 453–462, Canary Islands, Spain, June 1997.

[Fügen & Kolss 07] C. Fügen, M. Kolss: The Influence of Utterance Chunking on Machine Translation Performance. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2837–2840, Antwerp, Belgium, August 2007.

[Gaido & Gangi$^+$ 20] M. Gaido, M.A.D. Gangi, M. Negri, M. Cettolo, M. Turchi: Contextualized Translation of Automatically Segmented Speech. In H. Meng, B. Xu, T.F. Zheng, editors, *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1471–1475, 2020.

[Gangi & Cattoni$^+$ 19] M.A.D. Gangi, R. Cattoni, L. Bentivogli, M. Negri, M. Turchi: MuST-C: A Multilingual Speech Translation Corpus. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 2012–2017, Minneapolis, MN, USA, April 2019.

[Gangi & Enyedi$^+$ 19] M.A.D. Gangi, R. Enyedi, A. Brusadin, M. Federico: Robust Neural Machine Translation for Clean and Noisy Speech Transcripts. In *International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, China, November 2019.

[Gangi & Negri$^+$ 19a] M.A.D. Gangi, M. Negri, R. Cattoni, R. Dessì, M. Turchi: Enhancing Transformer for End-to-end Speech-to-Text Translation. In *Machine Translation Summit (MTSummit)*, pp. 21–31, Dublin, Ireland, August 2019.

[Gangi & Negri$^+$ 19b] M.A.D. Gangi, M. Negri, N.V. Nhat, A. Tebbifakhr, M. Turchi: Data Augmentation for End-to-End Speech Translation: FBK@IWSLT '19. In *International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, China, November 2019.

[Gangi & Negri$^+$ 19c] M.A.D. Gangi, M. Negri, M. Turchi: Adapting Transformer to End-to-End Spoken Language Translation. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1133–1137, Graz, Austria, September 2019.

[Gangi & Negri$^+$ 19d] M.A.D. Gangi, M. Negri, M. Turchi: One-to-Many Multilingual End-to-End Speech Translation. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 585–592, Sentosa, Singapore, December 2019.

[Gao & Wang$^+$ 19] Y. Gao, W. Wang, H. Ney: uniblock: Scoring and Filtering Corpus with Unicode Block Information. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1324–1329, Hong Kong, China, November 2019.

[Garg & Peitz$^+$ 19] S. Garg, S. Peitz, U. Nallasamy, M. Paulik: Jointly Learning to Align and Translate with Transformer Models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4452–4461, Hong Kong, China, November 2019.

[Gehring & Auli$^+$ 17a] J. Gehring, M. Auli, D. Grangier, Y. Dauphin: A Convolutional Encoder Model for Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 123–135, Vancouver, Canada, July - August 2017.

[Gehring & Auli$^+$ 17b] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin: Convolutional Sequence to Sequence Learning. In *International Conference on Machine Learning (ICML)*, pp. 1243–1252, Sydney, Australia, August 2017.

[Gers & Schmidhuber$^+$ 00] F.A. Gers, J. Schmidhuber, F.A. Cummins: Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, Vol. 12, No. 10, pp. 2451–2471, 2000.

[Gers & Schraudolph$^+$ 02] F.A. Gers, N.N. Schraudolph, J. Schmidhuber: Learning Precise Timing with LSTM Recurrent Networks. *Journal of Machine Learning Research*, Vol. 3, pp. 115–143, 2002.

[Ghader & Monz 17] H. Ghader, C. Monz: What Does Attention in Neural Machine Translation Pay Attention to? In *International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 30–39, Taipei, Taiwan, November - December 2017.

[Gibson & Hain 06] M. Gibson, T. Hain: Hypothesis Spaces for Minimum Bayes Risk Training in Large Vocabulary Speech Recognition. In *Conference on Spoken Language Processing (ICSLP)*, Pittsburgh, PA, USA, September 2006.

[Godard & Boito$^+$ 18] P. Godard, M.Z. Boito, L. Ondel, A. Berard, F. Yvon, A. Villavicencio, L. Besacier: Unsupervised Word Segmentation from Speech with Attention. In B. Yegnanarayana, editor, *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2678–2682, Hyderabad, India, September 2018.

[Godfrey & Holliman$^+$ 92] J.J. Godfrey, E.C. Holliman, J. McDaniel: SWITCHBOARD: Telephone Speech Corpus for Research and Development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 517–520, San Francisco, CA, USA, March 1992.

[Goodfellow & Bengio$^+$ 16] I. Goodfellow, Y. Bengio, A. Courville: *Deep Learning.* The MIT Press, 2016.

[Goodfellow & Warde-Farley$^+$ 13] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A.C. Courville, Y. Bengio: Maxout Networks. In *International Conference on Machine Learning (ICML)*, Vol. 28, pp. 1319–1327, Atlanta, GA, USA, June 2013.

[Graves 08] A. Graves: *Supervised Sequence Labelling with Recurrent Neural Networks.* Ph.D. thesis, Technical University Munich, 2008.

[Graves 12a] A. Graves: Sequence Transduction with Recurrent Neural Networks. In *International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, July 2012.

[Graves 12b] A. Graves: *Supervised Sequence Labelling with Recurrent Neural Networks*, Vol. 385 of *Studies in Computational Intelligence.* 2012.

[Graves & Fernández$^+$ 06] A. Graves, S. Fernández, F.J. Gomez, J. Schmidhuber: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *International Conference on Machine Learning (ICML)*, Vol. 148, pp. 369–376, Pittsburgh, PA, USA, June 2006.

[Graves & Jaitly^{+} 13a] A. Graves, N. Jaitly, A. Mohamed: Hybrid Speech Recognition with Deep Bidirectional LSTM. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 273–278, Olomouc, Czech Republic, December 2013.

[Graves & Jaitly^{+} 13b] A. Graves, N. Jaitly, A. Mohamed: Hybrid Speech Recognition with Deep Bidirectional LSTM. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 273–278, Olomouc, Czech Republic, December 2013.

[Graves & Schmidhuber 08] A. Graves, J. Schmidhuber: Offline Handwriting Recognition with Multidimensional Recurrent Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 545–552, Vancouver, Canada, December 2008.

[Gulati & Qin^{+} 20] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, R. Pang: Conformer: Convolution-Augmented Transformer for Speech Recognition. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 5036–5040, Shanghai, China, October 2020.

[Gülçehre & Firat^{+} 15] Ç. Gülçehre, O. Firat, K. Xu, K. Cho, L. Barrault, H. Lin, F. Bougares, H. Schwenk, Y. Bengio: On using Monolingual Corpora in Neural Machine Translation. *CoRR*, Vol. abs/1503.03535, 2015.

[Ha & Niehues^{+} 16] T. Ha, J. Niehues, A.H. Waibel: Toward Multilingual Neural Machine Translation with Universal Encoder and Decoder. *CoRR*, Vol. abs/1611.04798, 2016.

[Han & Zhang^{+} 20] W. Han, Z. Zhang, Y. Zhang, J. Yu, C. Chiu, J. Qin, A. Gulati, R. Pang, Y. Wu: ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3610–3614, Shanghai, China, October 2020.

[Hannun & Lee^{+} 19] A. Hannun, A. Lee, Q. Xu, R. Collobert: Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3785–3789, Graz, Austria, September 2019.

[Hao & Wang^{+} 19] J. Hao, X. Wang, B. Yang, L. Wang, J. Zhang, Z. Tu: Modeling Recurrence for Transformer. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 1198–1207, Minneapolis, MN, USA, June 2019.

[Hasler & de Gispert^{+} 18] E. Hasler, A. de Gispert, G. Iglesias, B. Byrne: Neural Machine Translation Decoding with Terminology Constraints. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 506–512, New Orleans, LA, USA, June 2018.

[He & Deng^{+} 11] X. He, L. Deng, A. Acero: Why Word Error Rate Is Not a Good Metric for Speech Recognizer Training for the Speech Translation Task? In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5632–5635, Prague, Czech Republic, May 2011.

[He & Lin 16] H. He, J.J. Lin: Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 937–948, San Diego, CA, USA, June 2016.

[He & Sainath$^+$ 19] Y. He, T.N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K.C. Sim, T. Bagby, S. Chang, K. Rao, A. Gruenstein: Streaming End-to-End Speech Recognition for Mobile Devices. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 6381–6385, Brighton, UK, May 2019.

[He & Zhang$^+$ 16] K. He, X. Zhang, S. Ren, J. Sun: Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[Hermansky & Ellis$^+$ 00] H. Hermansky, D.P.W. Ellis, S. Sharma: Tandem Connectionist Feature Extraction for Conventional HMM Systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1635–1638, Istanbul, Turkey, June 2000.

[Ho & Yvon 19] A.K.N. Ho, F. Yvon: Neural Baselines for Word Alignment. In *International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, China, November 2019.

[Ho & Yvon 20] A.K.N. Ho, F. Yvon: Generative Latent Neural Models for Automatic Word Alignment. In M. Denkowski, C. Federmann, editors, *Association for Machine Translation in the Americas (AMTA)*, pp. 64–77, Orlando, FL, USA, September 2020.

[Hochreiter 91] S. Hochreiter: Untersuchungen zu Dynamischen Neuronalen Netzen. *Diploma, Technische Universität München*, Vol. 91, No. 1, 1991.

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber: Long Short-Term Memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, November 1997.

[Hori & Watanabe$^+$ 17] T. Hori, S. Watanabe, Y. Zhang, W. Chan: Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 949–953, Stockholm, Sweden, August 2017.

[Hou & Zhang$^+$ 17] J. Hou, S. Zhang, L. Dai: Gaussian Prediction based Attention for Online End-to-End Speech Recognition. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3692–3696, Stockholm, Sweden, August 2017.

[Hu & Auli$^+$ 14] Y. Hu, M. Auli, Q. Gao, J. Gao: Minimum Translation Modeling with Recurrent Neural Networks. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 20–29, Gothenburg, Sweden, April 2014.

[Hu & Lu$^+$ 14] B. Hu, Z. Lu, H. Li, Q. Chen: Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2042–2050, Montreal, Canada, December 2014.

[Hutter 95] H. Hutter: Comparison of a New Hybrid Connectionist-SCHMM Approach with Other Hybrid Approaches for Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 3311–3314, Detroit, MI, USA, May 1995.

[Inaguma & Duh$^+$ 19] H. Inaguma, K. Duh, T. Kawahara, S. Watanabe: Multilingual End-to-End Speech Translation. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 570–577, Sentosa, Singapore, December 2019.

[Iranzo-Sánchez & Silvestre-Cerdà$^+$ 20] J. Iranzo-Sánchez, J.A. Silvestre-Cerdà, J. Jorge, N. Roselló, A. Giménez, A. Sanchís, J. Civera, A. Juan: Europarl-ST: A Multilingual Corpus for Speech Translation of Parliamentary Debates. Vol., pp. 8229–8233, May 2020.

[Irie 20] K. Irie: *Advancing Neural Language Modeling in Automatic Speech Recognition*. Ph.D. thesis, RWTH Aachen University, 2020.

[Irie & Zeyer$^+$ 19] K. Irie, A. Zeyer, R. Schlüter, H. Ney: Language Modeling with Deep Transformers. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3905–3909, Graz, Austria, September 2019.

[Jean & Cho$^+$ 15] S. Jean, K. Cho, R. Memisevic, Y. Bengio: On using Very Large Target Vocabulary for Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1–10, Beijing, China, July 2015.

[Jelinek 97] F. Jelinek: *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[Jia & Johnson$^+$ 19] Y. Jia, M. Johnson, W. Macherey, R.J. Weiss, Y. Cao, C. Chiu, N. Ari, S. Laurenzo, Y. Wu: Leveraging Weakly Supervised Data to Improve End-to-End Speech-to-text Translation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7180–7184, Brighton, UK, May 2019.

[Johnson & Schuster$^+$ 17] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F.B. Viégas, M. Wattenberg, G. Corrado, M. Hughes, J. Dean: Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 339–351, 2017.

[Jouppi & Young$^+$ 17] N.P. Jouppi, C. Young, N. Patil, D.A. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T.V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C.R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, D.H. Yoon: In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, Toronto, Canada, June 2017.

[Jurafsky & Martin 00] D. Jurafsky, J.H. Martin: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 1 edition, 2000.

[Kaiser & Bengio 16] L. Kaiser, S. Bengio: Can Active Memory Replace Attention? In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 3774–3782, Barcelona, Spain, December 2016.

[Kalchbrenner & Blunsom 13] N. Kalchbrenner, P. Blunsom: Recurrent Continuous Translation Models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1700–1709, Seattle, WA, USA, October 2013.

[Kalchbrenner & Danihelka$^+$ 16] N. Kalchbrenner, I. Danihelka, A. Graves: Grid Long Short-Term Memory. In *International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[Karita & Wang$^+$ 19] S. Karita, X. Wang, S. Watanabe, T. Yoshimura, W. Zhang, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N.E.Y. Soplin, R. Yamamoto: A

Comparative Study on Transformer vs RNN in Speech Applications. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 449–456, Singapore, Singapore, December 2019.

[Kim & Denton$^+$ 17] Y. Kim, C. Denton, L. Hoang, A.M. Rush: Structured Attention Networks. In *ICLR*, Toulon, France, April 2017.

[Kim & Hori$^+$ 17] S. Kim, T. Hori, S. Watanabe: Joint CTC-Attention based End-to-End Speech Recognition using Multi-Task Learning. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4835–4839, New Orleans, LA, USA, March 2017.

[Kim & Petrov$^+$ 19] Y. Kim, P. Petrov, P. Petrushkov, S. Khadivi, H. Ney: Pivot-based Transfer Learning for Neural Machine Translation between Non-English Languages. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 866–876, Hong Kong, China, November 2019.

[Kingma & Ba 15] D.P. Kingma, J. Ba: Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[Kitza & Golik$^+$ 19] M. Kitza, P. Golik, R. Schlüter, H. Ney: Cumulative Adaptation for BLSTM Acoustic Models. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 754–758, Graz, Austria, September 2019.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved Backing-off for M-gram language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 181–184, Detroit, MI, USA, May 1995.

[Ko & Peddinti$^+$ 15] T. Ko, V. Peddinti, D. Povey, S. Khudanpur: Audio Augmentation for Speech Recognition. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3586–3589, Dresden, Germany, September 2015.

[Koehn 10] P. Koehn: *Statistical Machine Translation*. Cambridge University Press, 2010.

[Koehn & Hoang$^+$ 07] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst: Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 177–180, Prague, Czech Republic, June 2007.

[Koehn & Knowles 17] P. Koehn, R. Knowles: Six Challenges for Neural Machine Translation. In *Workshop on Neural Machine Translation (NMT)*, pp. 28–39, Vancouver, Canada, August 2017.

[Koehn & Och$^+$ 03] P. Koehn, F.J. Och, D. Marcu: Statistical Phrase-based Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Edmonton, Canada, May 2003.

[Kriman & Beliaev$^+$ 20] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, Y. Zhang: Quartznet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 6124–6128, Barcelona, Spain, May 2020.

[Lawson & Chiu$^+$ 18a] D. Lawson, C. Chiu, G. Tucker, C. Raffel, K. Swersky, N. Jaitly: Learning Hard Alignments with Variational Inference. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5799–5803, Calgary, Canada, April 2018.

[Lawson & Chiu$^+$ 18b] D. Lawson, C. Chiu, G. Tucker, C. Raffel, K. Swersky, N. Jaitly: Learning Hard Alignments with Variational Inference. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5799–5803, Calgary, Canada, April 2018.

[Le & Allauzen$^+$ 12] H.S. Le, A. Allauzen, F. Yvon: Continuous Space Translation Models with Neural Networks. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 39–48, Montreal, Canada, June 2012.

[Lee & Roukos 04] Y. Lee, S. Roukos: IBM Spoken Language Translation System Evaluation. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 39–46, Keihanna Science City, Kyoto, Japan, September - October 2004.

[Leifert & Strauß$^+$ 16] G. Leifert, T. Strauß, T. Grüning, W. Wustlich, R. Labahn: Cells in Multidimensional Recurrent Neural Networks. *Journal of Machine Learning Research*, Vol. 17, No. 1, pp. 3313–3349, 2016.

[Levenshtein 66] V. Levenshtein: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, Vol. 10, pp. 707–710, 1966.

[Li & Liu$^+$ 18] X. Li, L. Liu, Z. Tu, S. Shi, M. Meng: Target Foresight based Attention for Neural Machine Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 1380–1390, New Orleans, LA, USA, June 2018.

[Li & Mohamed$^+$ 16] J. Li, A. Mohamed, G. Zweig, Y. Gong: Exploring Multidimensional LSTMs for Large Vocabulary ASR. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4940–4944, Shanghai, China, March 2016.

[Li & Sun 09] Z. Li, M. Sun: Punctuation as Implicit Annotations for Chinese Word Segmentation. *Computational Linguistics*, Vol. 35, No. 4, pp. 505–512, 2009.

[Li & Wang$^+$ 20] Y. Li, Q. Wang, T. Xiao, T. Liu, J. Zhu: Neural Machine Translation with Joint Representation. In *Conference on Artificial Intelligence (AAAI)*, pp. 8285–8292, New York, NY, USA, February 2020.

[Lin & Sun$^+$ 18] J. Lin, X. Sun, X. Ren, S. Ma, J. Su, Q. Su: Deconvolution-based Global Decoding for Neural Machine Translation. In *International Conference on Computational Linguistics (COLING)*, pp. 3260–3271, Santa Fe, NM, USA, August 2018.

[Lippmann 87] R.P. Lippmann: An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, Vol., pp. 4–22, April 1987.

[Lison & Tiedemann 16] P. Lison, J. Tiedemann: OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia, May 2016.

[Liu & Duh$^+$ 20] X. Liu, K. Duh, L. Liu, J. Gao: Very Deep Transformers for Neural Machine Translation. *CoRR*, Vol. abs/2008.07772, 2020.

[Liu & Gu$^+$ 03] F. Liu, L. Gu, Y. Gao, M. Picheny: Use of Statistical N-gram Models in Natural Language Generation for Machine Translation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 636–639, Hong Kong, China, April 2003.

[Liu & Saleh$^+$ 18] P.J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, N. Shazeer: Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations (ICLR)*, Vancouver, Canada, April - May 2018.

[Liu & Utiyama$^+$ 16] L. Liu, M. Utiyama, A.M. Finch, E. Sumita: Neural Machine Translation with Supervised Attention. In *International Conference on Computational Linguistics (COLING)*, pp. 3093–3102, Osaka, Japan, December 2016.

[Liu & Xiong$^+$ 19] Y. Liu, H. Xiong, Z. He, J. Zhang, H. Wu, H. Wang, C. Zong: End-to-End Speech Translation with Knowledge Distillation. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1128–1132, Graz, Austria, September 2019.

[Luong & Le$^+$ 16] M. Luong, Q.V. Le, I. Sutskever, O. Vinyals, L. Kaiser: Multi-Task Sequence to Sequence Learning. In *International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[Luong & Pham$^+$ 15] T. Luong, H. Pham, C.D. Manning: Effective Approaches to Attention-based Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412–1421, Lisbon, Portugal, September 2015.

[Lüscher & Beck$^+$ 19] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, H. Ney: RWTH ASR Systems for LibriSpeech: Hybrid vs Attention - w/o Data Augmentation. Vol., pp. 231–235, September 2019.

[Makarov 19] N. Makarov: Investigating Attention and Alignments in Neural Machine Translation, 2019.

[Malaviya & Ferreira$^+$ 18] C. Malaviya, P. Ferreira, A.F.T. Martins: Sparse and Constrained Attention for Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 370–376, Melbourne, Australia, July 2018.

[Mangu & Brill$^+$ 00] L. Mangu, E. Brill, A. Stolcke: Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks. *Computational Speech Language*, Vol. 14, No. 4, pp. 373–400, 2000.

[Manning & Schütze 99] C.D. Manning, H. Schütze: *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, USA, 1999.

[Matusov 09] E. Matusov: *Combining Natural Language Processing Systems to Improve Machine Translation of Speech*. Ph.D. thesis, RWTH Aachen University, 2009.

[Matusov & Hoffmeister$^+$ 08] E. Matusov, B. Hoffmeister, H. Ney: ASR Word Lattice Translation with Exhaustive Reordering Is Possible. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2342–2345, Brisbane, Australia, September 2008.

[Matusov & Kanthak$^+$ 05] E. Matusov, S. Kanthak, H. Ney: On the Integration of Speech Recognition and Statistical Machine Translation. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3177–3180, Lisboa, Portugal, September 2005.

[Matusov & Mauser$^+$ 06] E. Matusov, A. Mauser, H. Ney: Automatic Sentence Segmentation and Punctuation Prediction for Spoken Language Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 158–165, Kyoto, Japan, November 2006.

[Matusov & Ney$^+$ 05] E. Matusov, H. Ney, R. Schluter: Phrase-based Translation of Speech Recognizer Word Lattices using Loglinear Model Combination. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 110–115, San Juan, Puerto Rico, November 2005.

[Matusov & Ueffing^{+} 06] E. Matusov, N. Ueffing, H. Ney: Computing Consensus Translation for Multiple Machine Translation Systems using Enhanced Hypothesis Alignment. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy, April 2006.

[Matusov & Wilken^{+} 18] E. Matusov, P. Wilken, P. Bahar, J. Schamper, P. Golik, A. Zeyer, J.A. Silvestre-Cerda, A. Martinez-Villaronga, H. Pesch, J.T. Peter: Neural Speech Translation at AppTek. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 104–111, Bruges, Belgium, October 2018.

[McCarthy & Puzon^{+} 20] A.D. McCarthy, L. Puzon, J. Pino: SkinAugment: Auto-Encoding Speaker Conversions for Automatic Speech Translation. Vol., pp. 7924–7928, May 2020.

[McFee & McVicar^{+} 17] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, et al.: Librosa 0.5.0. Vol., February 2017.

[Meng & Lu^{+} 16] F. Meng, Z. Lu, H. Li, Q. Liu: Interactive Attention for Neural Machine Translation. In *International Conference on Computational Linguistics (COLING)*, pp. 2174–2185, Osaka, Japan, December 2016.

[Meng & Tu^{+} 18] F. Meng, Z. Tu, Y. Cheng, H. Wu, J. Zhai, Y. Yang, D. Wang: Neural Machine Translation with Key-Value Memory-Augmented Attention. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2574–2580, Stockholm,Sweden, July 2018.

[Merboldt & Zeyer^{+} 19] A. Merboldt, A. Zeyer, R. Schlüter, H. Ney: An Analysis of Local Monotonic Attention Variants. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1398–1402, Graz, Austria, September 2019.

[Mi & Sankaran^{+} 16] H. Mi, B. Sankaran, Z. Wang, A. Ittycheriah: Coverage Embedding Models for Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 955–960, Austin, TX, USA, November 2016.

[Mi & Wang^{+} 16] H. Mi, Z. Wang, A. Ittycheriah: Supervised Attentions for Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2283–2288, Austin, TX, USA, November 2016.

[Mikolov & Karafiát^{+} 10] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, S. Khudanpur: Recurrent Neural Network based Language Model. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1045–1048, Makuhari, Chiba, Japana, September 2010.

[Mikolov & Kombrink^{+} 11] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, S. Khudanpur: Extensions of Recurrent Neural Network Language Model. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5528–5531, Prague, Czech Republic, May 2011.

[Mino & Utiyama^{+} 17] H. Mino, M. Utiyama, E. Sumita, T. Tokunaga: Key-Value Attention Mechanism for Neural Machine Translation. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 290–295, Taipei, Taiwan, November - December 2017.

[Ney 99] H. Ney: Speech Translation: Coupling of Recognition and Translation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 517–520, Phoenix, AZ, USA, March 1999.

[Ney 19] H. Ney: Lecture Notes in Statistical Methods in Natural Language Processing, Human Language Technology and Pattern Recognition Group, Computer Science Department, RWTH Aachen University, October 2019.

[Nguyen & Stüker$^+$ 20] T. Nguyen, S. Stüker, J. Niehues, A. Waibel: Improving Sequence-to-Sequence Speech Recognition Training with On-The-Fly Data Augmentation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7689–7693, Barcelona, Spain, May 2020.

[Och 03] F.J. Och: Minimum Error Rate Training in Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 160–167, Sapporo, Japan, July 2003.

[Och & Ney 00] F.J. Och, H. Ney: Improved Statistical Alignment Models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Hong Kong, China, October 2000.

[Och & Ney 02] F.J. Och, H. Ney: Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 295–302, Philadelphia, PA, USA, July 2002.

[Och & Ney 03] F.J. Och, H. Ney: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51, 2003.

[Ohgushi & Neubig$^+$ 13] M. Ohgushi, G. Neubig, S. Sakti, T. Toda, S. Nakamura: An Empirical Comparison of Joint Optimization Techniques for Speech Translation. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2619–2623, Lyon, France, August 2013.

[Osamura & Kano$^+$ 18] K. Osamura, T. Kano, S. Sakti, K. Sudoh, S. Nakamura: Using Spoken Word Posterior Features in Neural Machine Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 189–195, Bruges, Belgium, October 2018.

[Ott & Edunov$^+$ 18] M. Ott, S. Edunov, D. Grangier, M. Auli: Scaling Neural Machine Translation. In *Workshop on Statistical Machine Translation (WMT)*, pp. 1–9, Belgium, Brussels, October - November 2018.

[Ott & Edunov$^+$ 19] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, M. Auli: fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 48–53, Minneapolis, MN, USA, June 2019.

[Pan & Yang 10] S.J. Pan, Q. Yang: A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, pp. 1345–1359, 2010.

[Panayotov & Chen$^+$ 15] V. Panayotov, G. Chen, D. Povey, S. Khudanpur: Librispeech: An ASR Corpus based on Public Domain Audio Books. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5206–5210, South Brisbane, Australia, April 2015.

[Papineni & Roukos$^+$ 98] K.A. Papineni, S. Roukos, R.T. Ward: Maximum Likelihood and Discriminative Training of Direct Translation Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 189–192, Seattle, WA, USA, May 1998.

[Papineni & Roukos+ 02] K. Papineni, S. Roukos, T. Ward, W. Zhu: Bleu: A Method for Automatic Evaluation of Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 311–318, Philadelphia, PA, USA, July 2002.

[Park & Chan+ 19] D.S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E.D. Cubuk, Q.V. Le: SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Vol., pp. 2613–2617, September 2019.

[Peitz & Wiesler+ 12] S. Peitz, S. Wiesler, M. Nußbaum-Thom, H. Ney: Spoken Language Translation using Automatically Transcribed Text in Training. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 276–283, Hong Kong, China, December 2012.

[Peter & Guta+ 17] J.T. Peter, A. Guta, T. Alkhouli, P. Bahar, J. Rosendahl, N. Rossenbach, M. Graça, N. Hermann: The RWTH Aachen University English-German and German-English Machine Translation System for WMT 2017. In *Workshop on Statistical Machine Translation (WMT)*, pp. 358–365, Copenhagen, Denmark, September 2017.

[Peter & Nix+ 17] J. Peter, A. Nix, H. Ney: Generating Alignments using Target Foresight in Attention-based Neural Machine Translation. *The Prague bulletin of mathematical linguistics*, Vol. 108, pp. 27–36, May 2017.

[Peter & Toutounchi+ 15] J.T. Peter, F. Toutounchi, S. Peitz, P. Bahar, A. Guta, H. Ney: The RWTH Aachen German to English MT System for IWSLT 2015. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 15–22, Da Nang, Vietnam, December 2015.

[Pino & Xu+ 20] J. Pino, Q. Xu, X. Ma, M.J. Dousti, Y. Tang: Self-Training for End-to-End Speech Translation. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1476–1480, Shanghai, China, October 2020.

[Pollack 90] J.B. Pollack: Recursive Distributed Representations. *Artificial Intelligence*, Vol. 46, No. 1–2, pp. 77–105, November 1990.

[Popel & Bojar 18] M. Popel, O. Bojar: Training Tips for the Transformer Model. *Prague Bull. Math. Linguistics*, Vol. 110, pp. 43–70, 2018.

[Popovic 15] M. Popovic: ChrF: Character N-gram F-score for Automatic MT Evaluation. In *Workshop on Statistical Machine Translation (WMT)*, pp. 392–395, Lisbon, Portugal, September 2015.

[Post 18] M. Post: A Call for Clarity in Reporting BLEU Scores. In *Workshop on Statistical Machine Translation (WMT)*, pp. 186–191, Belgium, Brussels, October - November 2018.

[Prabhavalkar & Rao+ 17] R. Prabhavalkar, K. Rao, T.N. Sainath, B. Li, L. Johnson, N. Jaitly: A Comparison of Sequence-to-Sequence Models for Speech Recognition. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 939–943, Stockholm, Sweden, August 2017.

[Prabhavalkar & Sainath+ 17] R. Prabhavalkar, T.N. Sainath, B. Li, K. Rao, N. Jaitly: An Analysis of Attention in Sequence-to-Sequence Models. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3702–3706, Stockholm, Sweden, August 2017.

[Press & Smith 18] O. Press, N.A. Smith: You May Not Need Attention. *CoRR*, Vol. abs/1810.13409, 2018.

[Press & Wolf 17] O. Press, L. Wolf: Using the Output Embedding to Improve Language Models. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 157–163, Valencia, Spain, April 2017.

[Rabiner 89] L.R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286, 1989.

[Rabiner & Juang 93] L.R. Rabiner, B. Juang: *Fundamentals of Speech Recognition.* Prentice Hall signal processing series. Prentice Hall, 1993.

[Raffel & Luong$^+$ 17] C. Raffel, M. Luong, P.J. Liu, R.J. Weiss, D. Eck: Online and Linear-Time Attention by Enforcing Monotonic Alignments. In *International Conference on Machine Learning (ICML)*, Vol. 70, pp. 2837–2846, Sydney, Australia, August 2017.

[Raison & Mazaré$^+$ 18] M. Raison, P. Mazaré, R. Das, A. Bordes: Weaver: Deep Co-Encoding of Questions and Documents for Machine Reading. *CoRR*, Vol. abs/1804.10490, 2018.

[Reiter 18] E. Reiter: A Structured Review of the Validity of BLEU. *Computational Linguistics*, Vol. 44, No. 3, 2018.

[Robbins & Monro 51] H. Robbins, S. Monro: A Stochastic Approximation Method. *Annals of Mathematical Statistics*, Vol. 22, No. 3, pp. 400–407, 1951.

[Robinson 94] A.J. Robinson: An Application of Recurrent Nets to Phone Probability Estimation. *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 298–305, 1994.

[Robinson & Failside 87] A.J. Robinson, F. Failside: Static and Dynamic Error Propagation Networks with Application to Speech Coding. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 632–641, Denver, CO, USA, December 1987.

[Rosenblatt 58] F. Rosenblatt: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, Vol., pp. 65–386, 1958.

[Rosendahl & Herold$^+$ 19] J. Rosendahl, C. Herold, Y. Kim, M. Graça, W. Wang, P. Bahar, Y. Gao, H. Ney: The RWTH Aachen University Machine Translation Systems for WMT 2019. In *Workshop on Statistical Machine Translation (WMT)*, pp. 349–355, Florence, Italy, July 2019.

[Rosendahl & Tran$^+$ 19] J. Rosendahl, V.A.K. Tran, W. Wang, H. Ney: Analysis of Positional Encodings for Neural Machine Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, China, November 2019.

[Rousseau & Deléglise$^+$ 14] A. Rousseau, P. Deléglise, Y. Estève: Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks. In *International Conference on Language Resources and Evaluation (LREC)*, pp. 3935–3939, Reykjavik, Iceland, May 2014.

[Ruder 19] S. Ruder: *Neural Transfer Learning for Natural Language Processing.* Ph.D. thesis, National University of Ireland, Galway, 2019.

[Ruiz & Gangi$^+$ 17] N. Ruiz, M.A.D. Gangi, N. Bertoldi, M. Federico: Assessing the Tolerance of Neural Machine Translation Systems Against Speech Recognition Errors. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2635–2639, Stockholm, Sweden, August 2017.

[Rumelhart & Hinton+ 86] D.E. Rumelhart, G.E. Hinton, R.J. Williams: Learning Representations by Back-Propagating Errors. *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

[Rumelhart & Hinton+ 88] D.E. Rumelhart, G.E. Hinton, R.J. Williams: *Learning Representations by Back-Propagating Errors*, 696–699. MIT Press, Cambridge, MA, USA, 1988.

[Rybach & Gollan+ 09] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, H. Ney: The RWTH Aachen University Open Source Speech Recognition System. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2111–2114, Brighton, UK, September 2009.

[Sabet & Dufter+ 20] M.J. Sabet, P. Dufter, F. Yvon, H. Schütze: SimAlign: High Quality Word Alignments without Parallel Training Data using Static and Contextualized Embeddings. In T. Cohn, Y. He, Y. Liu, editors, *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1627–1643, Punta Cana, Dominican Republic, November 2020.

[Sainath & Li 16] T.N. Sainath, B. Li: Modeling Time-Frequency Patterns with LSTM vs. Convolutional Architectures for LVCSR Tasks. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 813–817, San Francisco, CA, USA, September 2016.

[Sak & Shannon+ 17] H. Sak, M. Shannon, K. Rao, F. Beaufays: Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1298–1302, Stockholm, Sweden, August 2017.

[Sanabria & Caglayan+ 18] R. Sanabria, O. Caglayan, S. Palaskar, D. Elliott, L. Barrault, L. Specia, F. Metze: How2: A Large-Scale Dataset for Multimodal Language Understanding. In *Workshop on Visually Grounded Interaction and Language (VIGIL)*, Montreal, Canada, December 2018.

[Sankaran & Mi+ 16] B. Sankaran, H. Mi, Y. Al-Onaizan, A. Ittycheriah: Temporal Attention Model for Neural Machine Translation. *CoRR*, Vol. abs/1608.02927, 2016.

[Schamper & Rosendahl+ 18] J. Schamper, J. Rosendahl, P. Bahar, Y. Kim, A. Nix, H. Ney: The RWTH Aachen University Supervised Machine Translation Systems for WMT 2018. In *Workshop on Statistical Machine Translation (WMT)*, pp. 496–503, Brussels, Belgium, October 2018.

[Schlüter & Bezrukov+ 07] R. Schlüter, I. Bezrukov, H. Wagner, H. Ney: Gammatone Features and Feature Combination for Large Vocabulary Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 649–652, Honolulu, HI, USA, April 2007.

[Schuster & Paliwal 97] M. Schuster, K.K. Paliwal: Bidirectional Recurrent Neural Networks. *IEEE Transactions of Signal Processing*, Vol. 45, No. 11, pp. 2673–2681, 1997.

[Schwenk 12] H. Schwenk: Continuous Space Translation Models for Phrase-based Statistical Machine Translation. In *International Conference on Computational Linguistics (COLING)*, pp. 1071–1080, Mumbai, India, December 2012.

[Schwenk & Déchelotte+ 06] H. Schwenk, D. Déchelotte, J. Gauvain: Continuous Space Language Models for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Sydney, Australia, July 2006.

[Schwenk & Gauvain 02] H. Schwenk, J. Gauvain: Connectionist Language Modeling for Large Vocabulary Continuous Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 765–768, Portland, OR, USA, May 2002.

[Schwenk & Gauvain 04] H. Schwenk, J. Gauvain: Neural Network Language Models for Conversational Speech Recognition. In *Conference on Spoken Language Processing (CSLP)*, Jeju Island, Korea, October 2004.

[Schwenk & Rousseau$^+$ 12] H. Schwenk, A. Rousseau, M. Attik: Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *Worshop on Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pp. 11–19, Montreal, Canada, June 2012.

[Sennrich & Haddow$^+$ 16a] R. Sennrich, B. Haddow, A. Birch: Improving Neural Machine Translation Models with Monolingual Data. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, August 2016.

[Sennrich & Haddow$^+$ 16b] R. Sennrich, B. Haddow, A. Birch: Neural Machine Translation of Rare Words with Subword Units. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, August 2016.

[Shankar & Garg$^+$ 18] S. Shankar, S. Garg, S. Sarawagi: Surprisingly Easy Hard-Attention for Sequence to Sequence Learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 640–645, Brussels, Belgium, October - November 2018.

[Shankar & Sarawagi 19] S. Shankar, S. Sarawagi: Posterior Attention Models for Sequence to Sequence Learning. In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[Shaw & Uszkoreit$^+$ 18] P. Shaw, J. Uszkoreit, A. Vaswani: Self-Attention with Relative Position Representations. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 464–468, New Orleans, LA, USA, June 2018.

[Shen & Delaney$^+$ 07] W. Shen, B. Delaney, T.R. Anderson, R.E. Slyh: The MIT-LL/AFRL IWSLT-2007 MT System. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 95–102, Trento, Italy, October 2007.

[Snover & Dorr$^+$ 06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul: A Study of Translation Edit Rate with Targeted Human Annotation. In *Association for Machine Translation in the Americas (AMTA)*, pp. 223–231, Cambridge, MA, USA, August 2006.

[Sperber & Neubig$^+$ 17] M. Sperber, G. Neubig, J. Niehues, A. Waibel: Neural Lattice-to-Sequence Models for Uncertain Inputs. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1380–1389, Copenhagen, Denmark, September 2017.

[Sperber & Neubig$^+$ 19a] M. Sperber, G. Neubig, J. Niehues, A. Waibel: Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation. *Transactions of the Association for Computational Linguistics*, Vol. 7, pp. 313–325, 2019.

[Sperber & Neubig$^+$ 19b] M. Sperber, G. Neubig, N. Pham, A. Waibel: Self-Attentional Models for Lattice Inputs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1185–1197, Florence, Italy, July - August 2019.

[Sperber & Niehues$^+$ 17] M. Sperber, J. Niehues, A. Waibel: Toward Robust Neural Machine Translation for Noisy Input Sequences. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 90–96, Tokyo, Japan, December 2017.

[Sperber & Paulik 20] M. Sperber, M. Paulik: Speech Translation and the End-to-End Promise: Taking Stock of Where We Are. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 7409–7421, Seattle, WA, USA, July 2020.

[Sperber & Setiawan$^+$ 20] M. Sperber, H. Setiawan, C. Gollan, U. Nallasamy, M. Paulik: Consistent Transcription and Translation of Speech. *Transaction of Association Computational Linguistics*, Vol. 8, pp. 695–709, 2020.

[Srivastava & Hinton$^+$ 14] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Resource*, Vol. 15, No. 1, pp. 1929–1958, 2014.

[Stahlberg & Saunders$^+$ 18] F. Stahlberg, D. Saunders, B. Byrne: An Operation Sequence Model for Explainable Neural Machine Translation. In *Workshop in Analyzing and Interpreting Neural Networks for NLP (BlackboxNLP)*, pp. 175–186, Brussels, Belgium, November 2018.

[Stoian & Bansal$^+$ 20] M.C. Stoian, S. Bansal, S. Goldwater: Analyzing ASR Pretraining for Low-Resource Speech-to-Text Translation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7909–7913, Barcelona, Spain, May 2020.

[Strauß & Grüning$^+$ 16] T. Strauß, T. Grüning, G. Leifert, R. Labahn: CITlab ARGUS for Keyword Search in Historical Handwritten Documents - Description of CITlab's System for the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task. In *Conference and Labs of the Evaluation Forum (CLEF)*, Vol. 1609, pp. 399–412, Évora, Portugal, September 2016.

[Sundermeyer & Alkhouli$^+$ 14] M. Sundermeyer, T. Alkhouli, J. Wuebker, H. Ney: Translation Modeling with Bidirectional Recurrent Neural Networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 14–25, Doha, Qatar, October 2014.

[Sundermeyer & Ney$^+$ 15] M. Sundermeyer, H. Ney, R. Schlüter: From Feedforward to Recurrent LSTM Neural Networks for Language Modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 23, No. 3, pp. 517–529, 2015.

[Sundermeyer & Oparin$^+$ 13] M. Sundermeyer, I. Oparin, J. Gauvain, B. Freiberg, R. Schlüter, H. Ney: Comparison of Feedforward and Recurrent Neural Network Language Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 8430–8434, Vancouver, Canada, May 2013.

[Sundermeyer & Schlüter$^+$ 12] M. Sundermeyer, R. Schlüter, H. Ney: LSTM Neural Networks for Language Modeling. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 194–197, Portland, OR, USA, September 2012.

[Sung & Liu$^+$ 19] T. Sung, J. Liu, H. Lee, L. Lee: Towards End-to-End Speech-to-Text Translation with Two-pass Decoding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7175–7179, Brighton, United Kingdom, May 2019.

[Sutskever & Vinyals$^+$ 14] I. Sutskever, O. Vinyals, Q.V. Le: Sequence to Sequence Learning with Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 3104–3112, Montreal, Canada, December 2014.

[Synnaeve & Xu^{+} 20] G. Synnaeve, Q. Xu, J. Kahn, E. Grave, T. Likhomanenko, V. Pratap, A. Sriram, V. Liptchinsky, R. Collobert: End-to-End ASR: from Supervised to Semi-Supervised Learning with Modern Architectures. In *Workshop on Self-supervision in Audio and Speech (SAS)*, Vienna, Austria, July 2020.

[Tamura & Watanabe^{+} 14] A. Tamura, T. Watanabe, E. Sumita: Recurrent Neural Networks for Word Alignment Model. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1470–1480, Baltimore, MD, USA, June 2014.

[Tang & Sennrich^{+} 19] G. Tang, R. Sennrich, J. Nivre: Understanding Neural Machine Translation by Simplification: The Case of Encoder-Free Models. In *International Conference on Recent Advances in Natural Language Processing (RANLP)*, pp. 1186–1193, Varna, Bulgaria, September 2019.

[Tiedemann 12] J. Tiedemann: Parallel Data, Tools and Interfaces in OPUS. In *International Conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey, may 2012.

[Tilk & Alumäe 16] O. Tilk, T. Alumäe: Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3047–3051, San Francisco, CA, USA, September 2016.

[Tjandra & Sakti^{+} 17] A. Tjandra, S. Sakti, S. Nakamura: Local Monotonic Attention Mechanism for End-to-End Speech And Language Processing. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 431–440, Taipei, Taiwan, November - December 2017.

[Toshniwal & Kannan^{+} 18] S. Toshniwal, A. Kannan, C. Chiu, Y. Wu, T.N. Sainath, K. Livescu: A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition. In *IEEE Spoken Language Technology Workshop (SLT)*, pp. 369–375, Athens, Greece, December 2018.

[Tripathi & Lu^{+} 19] A. Tripathi, H. Lu, H. Sak, H. Soltau: Monotonic Recurrent Neural Network Transducer and Decoding Strategies. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 944–948, Sentosa, Singapore, December 2019.

[Tu & Lu^{+} 16] Z. Tu, Z. Lu, Y. Liu, X. Liu, H. Li: Modeling Coverage for Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, August 2016.

[Tüske & Saon^{+} 20] Z. Tüske, G. Saon, K. Audhkhasi, B. Kingsbury: Single Headed Attention based Sequence-to-Sequence Model for State-of-the-Art Results on Switchboard-300. Vol., pp. 551–555, October 2020.

[van Merriënboer & Bahdanau^{+} 15] B. van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, Y. Bengio: Blocks and Fuel: Frameworks for Deep Learning. *CoRR*, Vol. abs/1506.00619, 2015.

[Vaswani & Shazeer^{+} 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin: Attention Is All You Need. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, Long Beach, CA, USA, December 2017.

[Vaswani & Zhao^{+} 13] A. Vaswani, Y. Zhao, V. Fossum, D. Chiang: Decoding with Large-Scale Neural Language Models Improves Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1387–1392, Seattle, WA, USA, October 2013.

[Vilar & Popovic$^+$ 06] D. Vilar, M. Popovic, H. Ney: AER: do we need to improve our alignments? In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 205–212, Kyoto, Japan, November 2006.

[Viterbi 67] A.J. Viterbi: Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transaction of Information Theory*, Vol. 13, No. 2, pp. 260–269, 1967.

[Vogel & Ney$^+$ 96] S. Vogel, H. Ney, C. Tillmann: HMM-based Word Alignment in Statistical Translation. In *International Conference on Computational Linguistics (COLING)*, pp. 836–841, Copenhagen, Denmark, August 1996.

[Voigtlaender 16] P. Voigtlaender: Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks, 2016.

[Voigtlaender & Doetsch$^+$ 16] P. Voigtlaender, P. Doetsch, H. Ney: Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 228–233, Shenzhen, China, October 2016.

[Voita & Talbot$^+$ 19] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, I. Titov: Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5797–5808, Florence, Italy, July - August 2019.

[Waibel & Hanazawa$^+$ 88] A. Waibel, T. Hanazawa, G.E. Hinton, K. Shikano, K.J. Lang: Phoneme Recognition: Neural Networks vs. Hidden Markov Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 107–110, New York, NY, USA, April 1988.

[Wang & Alkhouli$^+$ 17] W. Wang, T. Alkhouli, D. Zhu, H. Ney: Hybrid Neural Network Alignment and Lexicon Model in Direct HMM for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 125–131, Vancouver, Canada, July - August 2017.

[Wang & Lu$^+$ 16] M. Wang, Z. Lu, H. Li, Q. Liu: Memory-Enhanced Decoder for Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 278–286, Austin, TX, USA, November 2016.

[Wang & Mohamed$^+$ 20] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang, C. Fuegen, G. Zweig, M.L. Seltzer: Transformer-based Acoustic Modeling for Hybrid Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 6874–6878, Barcelona, Spain, May 2020.

[Wang & Peter$^+$ 16] W. Wang, J. Peter, H. Rosendahl, H. Ney: CharacTer: Translation Edit Rate on Character Level. In *Workshop on Statistical Machine Translation (WMT)*, pp. 505–510, Berlin, Germany, August 2016.

[Wang & Pino$^+$ 20] C. Wang, J. Pino, A. Wu, J. Gu: CoVoST: A Diverse Multilingual Speech-to-Text Translation Corpus. In *International Conference on Language Resources and Evaluation (LREC)*, pp. 4197–4203, Marseille, France, May 2020.

[Wang & Wu$^+$ 20] C. Wang, Y. Wu, S. Liu, M. Zhou, Z. Yang: Curriculum Pre-Training for End-to-End Speech Translation. In D. Jurafsky, J. Chai, N. Schluter, J.R. Tetreault, editors, *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3728–3738, Seattle, WA, USA, July 2020.

[Wang & Yang^{+} 21] W. Wang, Z. Yang, Y. Gao, H. Ney: Transformer-based Direct Hidden Markov Model for Machine Translation. In *Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop (ACL/IJCNLP)*, pp. 23–32, Bangkok, Thailand, August 2021.

[Wang & Zheng 15] D. Wang, T.F. Zheng: Transfer Learning for Speech and Language Processing. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1225–1237, Hong Kong, China, December 2015.

[Wang & Zhu^{+} 18] W. Wang, D. Zhu, T. Alkhouli, Z. Gan, H. Ney: Neural Hidden Markov Model for Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 377–382, Melbourne, Australia, July 2018.

[Watanabe & Hori^{+} 17] S. Watanabe, T. Hori, S. Kim, J.R. Hershey, T. Hayashi: Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. *IEEE Journal of Selected Topics in Signal Processing*, Vol. 11, No. 8, pp. 1240–1253, 2017.

[Weiss & Chorowski^{+} 17] R.J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, Z. Chen: Sequence-to-Sequence Models Can Directly Translate Foreign Speech. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2625–2629, Stockholm, Sweden, August 2017.

[Werbos 88] P.J. Werbos: Generalization of Backpropagation with Application to a Recurrent Gas Market Model. *Neural Networks*, Vol. 1, No. 4, pp. 339–356, 1988.

[Werbos 90] P.J. Werbos: Backpropagation through Time: What It Does and How to Do It. *Proceedings of the IEEE*, Vol. 78, No. 10, pp. 1550–1560, 1990.

[Werlen & Pappas^{+} 18] L.M. Werlen, N. Pappas, D. Ram, A. Popescu-Belis: Self-Attentive Residual Decoder for Neural Machine Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 1366–1379, New Orleans, LA, USA, June 2018.

[Weston & Chopra^{+} 15] J. Weston, S. Chopra, A. Bordes: Memory Networks. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[Wiesler & Richard^{+} 14] S. Wiesler, A. Richard, P. Golik, R. Schlüter, H. Ney: RASR/NN: The RWTH Neural Network Toolkit for Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 3281–3285, Florence, Italy, May 2014.

[Williams 92] R.J. Williams: Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, Vol. 8, pp. 229–256, 1992.

[Wilson & Martinez 03] D.R. Wilson, T.R. Martinez: The General Inefficiency of Batch Training for Gradient Descent Learning. Vol. 16, No. 10, 2003.

[Wu & Cotterell 19] S. Wu, R. Cotterell: Exact Hard Monotonic Attention for Character-Level Transduction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1530–1537, Florence, Italy, July - August 2019.

[Wu & Fan^{+} 19] F. Wu, A. Fan, A. Baevski, Y.N. Dauphin, M. Auli: Pay Less Attention with Lightweight and Dynamic Convolutions. In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[Wu & Schuster$^+$ 16] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, Vol. abs/1609.08144, 2016.

[Wu & Shapiro$^+$ 18] S. Wu, P. Shapiro, R. Cotterell: Hard Non-Monotonic Attention for Character-Level Transduction. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4425–4438, Brussels, Belgium, October - November 2018.

[Wu & Xia$^+$ 18] L. Wu, Y. Xia, F. Tian, L. Zhao, T. Qin, J. Lai, T. Liu: Adversarial Neural Machine Translation. In *Asian Conference on Machine Learning (ACML)*, Vol. 95, pp. 534–549, Beijing, China, November 2018.

[Xu & Ba$^+$ 15] K. Xu, J. Ba, R. Kiros, K. Cho, A.C. Courville, R. Salakhutdinov, R.S. Zemel, Y. Bengio: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning (ICML)*, Vol. 37, pp. 2048–2057, Lille, France, July 2015.

[Yang & Hu$^+$ 17] Z. Yang, Z. Hu, Y. Deng, C. Dyer, A.J. Smola: Neural Machine Translation with Recurrent Attention Modeling. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 383–387, Valencia, Spain, April 2017.

[Yang & Liu$^+$ 13] N. Yang, S. Liu, M. Li, M. Zhou, N. Yu: Word Alignment Modeling with Context Dependent Deep Neural Network. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 166–175, Sofia, Bulgaria, August 2013.

[Yu & Blunsom$^+$ 17] L. Yu, P. Blunsom, C. Dyer, E. Grefenstette, T. Kociský: The Neural Noisy Channel. In *International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017.

[Zamora-Martínez & Bleda$^+$ 10] F. Zamora-Martínez, M.J.C. Bleda, H. Schwenk: N-gram-based Machine Translation Enhanced with Neural Networks for the French-English BTEC-IWSLT'10 Task. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 45–52, Paris, France, December 2010.

[Zenkel & Wuebker$^+$ 19] T. Zenkel, J. Wuebker, J. DeNero: Adding Interpretable Attention to Neural Translation Models Improves Word Alignment. *CoRR*, Vol. abs/1901.11359, 2019.

[Zenkel & Wuebker$^+$ 20] T. Zenkel, J. Wuebker, J. DeNero: End-to-End Neural Word Alignment Outperforms GIZA++. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1605–1617, Seattle, WA, USA, July 2020.

[Zens 08] R. Zens: *Phrase-based Statistical Machine Translation: Models, Search, Training.* Ph.D. thesis, RWTH Aachen University, 2008.

[Zens & Bender$^+$ 05] R. Zens, O. Bender, S. Hasan, S. Khadivi, E. Matusov, J. Xu, Y. Zhang, H. Ney: The RWTH Phrase-based Statistical Machine Translation System. In *International Workshop on Spoken Language Translation (IWSLT)*, pp. 145–152, Pittsburgh, PA, USA, October 2005.

[Zens & Och$^+$ 02] R. Zens, F.J. Och, H. Ney: Phrase-based Statistical Machine Translation. In *Advances in Artificial Intelligence, 25th Annual German Conference on AI*, Vol. 2479 of *Lecture Notes in Computer Science*, pp. 18–32, Aachen, Germany, September 2002.

[Zeyer & Alkhouli$^+$ 18] A. Zeyer, T. Alkhouli, H. Ney: RETURNN as a Generic Flexible Neural Toolkit with Application to Translation and Speech Recognition. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 128–133, Melbourne, Australia, July 2018.

[Zeyer & Bahar$^+$ 19] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, H. Ney: A Comparison of Transformer and LSTM Encoder Decoder Models for ASR. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 8–15, Sentosa, Singapore, December 2019.

[Zeyer & Irie$^+$ 18] A. Zeyer, K. Irie, R. Schlüter, H. Ney: Improved Training of End-to-End Attention Models for Speech Recognition. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 7–11, Hyderabad, India, September 2018.

[Zeyer & Merboldt$^+$ 18] A. Zeyer, A. Merboldt, R. Schlüter, H. Ney: A Comprehensive Analysis on Attention Models. In *Workshop on Interpretability and Robustness in Audio, Speech, and Language (IRASL), Conference on Neural Information Processing Systems (NeurIPS)*, Montreal, Canada, December 2018.

[Zhang & Lu$^+$ 20] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, S. Kumar: Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7829–7833, Barcelona, Spain, May 2020.

[Zhang & Wang$^+$ 17] J. Zhang, M. Wang, Q. Liu, J. Zhou: Incorporating Word Reordering Knowledge into Attention-based Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1524–1534, Vancouver, Canada, July - August 2017.

[Zhang & Xiong$^+$ 16] B. Zhang, D. Xiong, J. Su: Recurrent Neural Machine Translation. *CoRR*, Vol. abs/1607.08725, 2016.

[Zhang & Xiong$^+$ 17a] B. Zhang, D. Xiong, J. Su: A GRU-Gated Attention Model for Neural Machine Translation. *CoRR*, Vol. abs/1704.08430, 2017.

[Zhang & Xiong$^+$ 17b] B. Zhang, D. Xiong, J. Su, H. Duan: A Context-Aware Recurrent Encoder for Neural Machine Translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 25, No. 12, pp. 2424–2432, 2017.