Improving Neural Machine Translation on resource-limited pairs using auxiliary data of a third language

Ander Martínez ander.martinez.zy4@is.naist.jp
Yuji Matsumoto matsu@is.naist.jp
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192, Japan

Abstract

In the recent years interest in Deep Neural Networks (DNN) has grown in the field of Natural Language Processing, as new training methods have been proposed. The usage of DNN has achieved state-of-the-art performance in various areas. Neural Machine Translation (NMT) described by Bahdanau et al. (2014) and its successive variations have shown promising results. DNN, however, tend to over-fit on small data-sets, which makes this method impracticable for resource-limited language pairs. This article combines three different ideas (splitting words into smaller units, using an extra dataset of a related language pair and using monolingual data) for improving the performance of NMT models on language pairs with limited data. Our experiments show that, in some cases, our proposed approach to subword-units performs better than BPE (Byte pair encoding) and that auxiliary language-pairs and monolingual data can help improve the performance of languages with limited resources.

1 Introduction

In the recent years interest in Deep Neural Networks (DNN) has grown in the field of Natural Language Processing (NLP), as new training methods (Blunsom and Kalchbrenner, 2013; Sutskever et al., 2014) have been proposed. The encoder-decoder approach for Neural Machine Translation (NMT) consists in encoding the source sentence into an intermediate vector representation and then generating (decoding) the target sentence from this representation. Cho et al. (2014) is an example of this approach.

The NMT approach of jointly training alignment and translation models described by Bahdanau et al. (2014) and its successive variations have shown promising results. Its attention mechanism deals with the problem of having a fixed length vector for sentences of varying length by encoding the source sentence into a set of vectors, one vector for each of the tokens in the source sentence.

NMT doesn't need complex feature engineering, which is convenient when dealing with resource-limited languages. However, a large parallel corpus is still needed in order to get competitive performance and avoid overfitting. As an example, Bahdanau et al. (2014) and Jean et al. (2015) use a dataset of about 12 million parallel sentences.

Two main problems arise when using a small dataset for training a MT model.

One of those problems is that the vocabulary exposed by a small dataset is inherently small. Also, even if a word shows up in the data it may occur too few times for learning a reliable representation. One strategy for minimizing this problem is subdividing words into subword

units, like syllables. Doing so reduces the total vocabulary size and increases the hit-rate of each symbol in the dataset. This has been explored in Sennrich et al. (2015b).

Another problem concerns large NMT models. When the number of parameters is too large compared to the data size, the model may optimize for memorizing all or a large part of the dataset instead of modeling the translation; overfitting in practice. A solution to this problem is to reduce the model size to better match the amount of data. However, many relevant features may not be modeled with a smaller number of parameters. Another approach is to artificially increase the number of samples by counterfeiting or using data of a third language.

This research explores how much of an improvement using auxiliary parallel sentences from a third language to the target language (A $\rightarrow$ T) in modeling MT from of a resource-limited language pair (S $\rightarrow$ T) brings. We also explore the effect of using an auxiliary language on the decoder side.

For the case of phrase-based Statistical Machine Translation, similar ideas have been explored before with varied results. For example, for closely related pairs (Nakov and Ng, 2012) or through lexical triangulation (Crego et al., 2010; Dholakia and Sarkar, 2014). For NMT, a couple of authors have also explored this possibility. Dong et al. (2015) modeled translation to different targets from a common source with shared representation. Firat et al. (2016) also explored the case of a common target language for different source languages. Both papers claimed to get higher translation quality over individually trained models. A comparison to these papers follows in Section 2.

In order to assist the learning of the target language pair, MT for the auxiliary pair is trained jointly. We argue that doing so prevents the language pair with the small dataset from overfitting and leads to more robust models. This problem could also be addressed through Transfer Learning, as explored in Yosinski et al. (2014) and Zoph et al. (2016), but that approach falls outside the scope of this article.

A third solution to parallel data scarcity is using monolingual data in addition in order to make the Language Model (LM) at the target side stronger. A strong LM at the decoder can increase performance, as already tried by Sennrich et al. (2015a).

We perform experiments that are analogous to the ones described in Firat et al. (2016) with focus on the more resource-limited pairs and use their results as a baseline for comparison.

This article has the following sections: Section 2 summarizes previous related work; in Section 3, the proposed approach is described; Section 4 presents some experiments with their results and analysis; finally, in Section 5, we draw some conclusions.

2 Related work

2.1 Neural Machine Translation

The Neural Machine Translation method proposed in Bahdanau et al. (2014) on which this work is based is briefly described in this section.

NMT models, like SMT models, are trained to maximize the conditional log-probability of every translation in the training set $Y^{(i)}$ w.r.t. their corresponding source sentence $X^{(i)}$ and model's parameters θ.

$$\theta^* = \arg\max_{\theta} \sum_{i:Y^{(i)} \in Y} \log p(Y^{(i)}|X^{(i)}, \theta). \tag{1}$$

In order to compute the probability of a translation, the sequence $x = (x_1, \ldots, x_T)$ is first encoded into a sequence of annotations $h = (h_1, \ldots, h_T)$, by a bidirectional recurrent neural network.

$$h_t = [\overleftarrow{h}_t; \vec{h}_t] = [f_1(x_t, \overleftarrow{h}_{t+1}); f_2(x_t, \vec{h}_{t-1})], \tag{2}$$

where f_1 and f_2 are both Gated Recurrent Units (GRU) as described in Cho et al. (2014).

This annotations are used by the decoder to estimate the probability of the translation, one word at a time, based also on the previous words in the sentence. A convex sum of the annotations, c_τ, is computed each time according to their contribution to the upcoming word. The weight of each annotation is computed in the following way:

$$c_\tau = \sum_{i=1}^{T_x} \alpha_{\tau,i} h_i, \tag{3}$$

$$\alpha_{\tau,i} = \frac{\exp\{a(h_i, z_{\tau-1}, E_y[\tilde{y}_{\tau-1}])\}}{\sum_{j=1}^{T_x} \exp\{a(h_j, z_{\tau-1}, E_y[\tilde{y}_{\tau-1}])\}}, \tag{4}$$

where $z_{\tau-1}$ is the previous hidden state of the decoder GRU and $E_y[\tilde{y}_{\tau-1}]$ is the word embedding of the previously produced word. During training, the embedding of the expected previous word is used instead. The function a scores the relevance of the annotation in the current context. It is defined as a projection of the sum of the projections of each of its parameters.

A softmax layer can be applied on a projection of c_τ, z_τ and the embedding of the previous work to compute the probability of each candidate word in the target vocabulary.

$$p(y_\tau | y_{<\tau}, x) \propto exp\{q(y_{\tau-1}, z_\tau, c_\tau)\}. \tag{5}$$

When generating translations of new sentences beam-search is used based on the trained probability model.

The hidden state of the decoder z_τ is updated with respect to the convex sum c_τ, as described in Cho et al. (2014).

$$z_\tau = g(y_{\tau-1}, z_{\tau-1}, c_\tau). \tag{6}$$

In this article, the part of the network that produces h is referred as *encoder* and the rest of the network as *decoder*. In articles by other authors, the parameters used exclusively to compute α_τ may not be considered part of the decoder.

2.2 Subword-units

Out-of-vocabulary words and low word hits are an inherent problem to small datasets. Instead of using words as translation unit, sub-word elements can be used. By doing so, we get more symbols from the same dataset and thus a bigger percentage of all possible symbols will appear and the number of appearances of each of these symbols will become higher. Also, by making the symbols shorter the number of possible symbols also reduces.

Sennrich et al. (2015b) tried using subword units for improving translation of rare words. For doing so, they first applied BPE (Byte pair encoding) (Gage, 1994) to the word list at the character level. BPE consists in replacing the most common symbol pairs with a new symbol consecutively until the symbol table has a certain size. They didn't merge symbols across words.

In order to have more similar subword units at the source and target languages, they transliterated the Cyrillic characters into Latin characters and trained the merging jointly. This works for the use case explored in that article of translating proper names and other words that can be mapped phonetically. However, it doesn't match well with morphemes (the smallest meaningful unit of a language), which are usually short, and it minimizes word hits, which isn't desirable in the case of small datasets.

They tried two vocabulary sizes: one of 60,000 words and another one of 90,000 words for the joint case.

2.3 Multilingual Neural Machine translation

The idea of jointly training additional language pairs to improve the translation quality of a model has already been tried.

The method explained in Dong et al. (2015) consists in translating to a set of languages from English using the same English encoder for each pair. They only investigate the case where the source language is shared. This approach mainly improves the quality of the encoder side as it studies more datapoints, but the quality of the decoder doesn't improve that much because its amount of data remains the same.

The model described by Bahdanau et al. (2014) has 85,967,240 floating point parameters when using a vocabulary size of 30,000 at both ends. Of these parameters 32.95% are related to the encoder and 67.05% to the decoder and soft-alignment system. This suggests that the decoder will overfit more easily under data-scarce conditions.

Another article investigating the use of additional language pairs is Firat et al. (2016). They train five language pairs in both directions, which makes ten individual models. For each of the six languages, the same encoder and decoder parameters are used when repeated in a pair and the parameters of the attention mechanism (function a in Equation (4) in this article) are shared for all pairs.

In the encoder, the hidden states, called *annotations*, obtained from the forward and backward RNN are projected into a new vector. This allows for the annotations to be more language-independent, as it doesn't make a difference whether a feature is extracted by the network iterating over the source sentence forwards or backwards.

$$h_t^n = W_{adp}^n[\overleftarrow{h}_t; \vec{h}_t] \tag{7}$$

For each of the language pairs, they feed a minibatch to the corresponding model and update its weights accordingly; one language-pair at a time.

They obtained significant improvement for small datasets and when the repeated language (English) was in the decoder.

They applied this method to datasets of varying sizes, starting on 100K parallel sentences for English-French translation and 210K for German-English.

2.4 Monolingual data

Sennrich et al. (2015a) introduced a method for integrating pre-trained LMs with NMT models in order to improve the translation quality. In their Deep-Fusion approach, they trained a LM on monolingual data and a NMT model on parallel data, and then integrated the LM prediction before the Softmax layer to contribute in the selection of the next word.

The LM was based on the RNNLM (Deoras, 2011) approach using GRUs in the decoder, which in effect, is very similar to the model described in Section 2.1 but without the attention mechanism. That is, the decoder only depends on one monolingual y sentence, without any encoder. The equivalents to Equations 5 and 6 are:

$$p(y_\tau|y_{<\tau}) \propto exp\{q(y_{\tau-1}, z_\tau^{\mathrm{LM}})\}. \tag{8}$$

$$z_\tau^{\mathrm{LM}} = g(y_{\tau-1}, z_{\tau-1}^{\mathrm{LM}}). \tag{9}$$

In order to integrate this LM with the Translation Model (TM), they first scale the hidden state of the LM and then concatenate it to the hidden state of the TM before computing the softmax. The scale factor g_τ for the LM is given by:

$$g_\tau = \sigma(v_g^T z_\tau^{\mathrm{LM}} + b_g), \tag{10}$$

where v_g and b_g are learned weights and bias, respectively. After merging, the equivalent to Equation 5 is

$$p(y_\tau|y_{<\tau}, x) \propto exp\{q(y_{\tau-1}, z_\tau, z_\tau^{\mathrm{LM}}, c_\tau)\}. \quad (11)$$

The model from Bahdanau et al. (2014) already trains something close to a LM from the parallel data, as can be seen in Equation 5. This approach increases the complexity of the model, as two states for two decoders need to be updated for every timestep.

3 Proposed solution

The method proposed here consists in jointly training various models with shared parameters; either the encoder or the decoder parameters. By jointly training the models they will co-adapt and benefit from each other. In the experiments we only explore the possibility of using one extra auxiliary language, either as the source language or as the target language, in addition to the intended source and target languages. More than one extra language could be used in a similar fashion.

In addition to using parallel data from an auxiliary language we also experiment with using monolingual data to obtain improved results.

We used subword units instead of words as the translation unit as a method to deal with large vocabulary sizes and differences in morpheme-per-word ratios (synthetic vs isolating).

3.1 Subword units

We pre-processed the data to split words into subword units for training.

Any word in the dataset was split into a number of subword units equal to its number of vowels. Each of the subword units consists of a vowel with all its surrounding consonants. Numbers aren't split. Word-boundary marks were included into the subwords. As an example, the sentence "the 54 polychromatic mats" is split into the sequence ["_the_", "_54_", "_pol", "lychr", "chrom", "mat", "tic_", "_mats_"].

The motivation behind this approach is that the subword-units generated by this approach are similar in shape to syllables and syllables map relatively well to morphemes in many languages. By using these short subword-units the model could learn some kind of morphological derivation.

In our experiments we used a vocabulary size of 30k symbols. Using only these symbols would result in too many out-of-vocabulary symbols. In order to alleviate this problem we tried to fit the infrequent symbols into the vocabulary by trimming one consonant at a time from the beginning or the end of the symbol until they matched a symbol in the vocabulary. We didn't trim consonants from word-boundaries and when for the inner symbols we trimmed from the side with more consonants. This reduced the number of unknown symbols considerably but didn't eliminate them. Note that sometimes excessive trimming can happen to the point that the original word cannot be restored, but this is still preferable to an unknown symbol.

The subword units are merged after translation before computing the BLEU score in order to be comparable to other authors' results. The subword units are merged using a simple regular expression of the form "`s/([:consonant:]+) \1/\1/g`".

We notice two problems with this approach. For languages with long consonants clusters like Czech the vocabulary size will grow faster. We observed this problem with German when compared to English. The number of out-of-vocabulary symbols in our datasets can be seen in Table 1.

The other problem is related to sound changes. For languages with a lot of sound changes, like consonant gradation in Finnish, morphological changes can produce a different symbol, which increases the need of data. As an example, the word *poika* will use the symbols `_poik` and `ka_` but in genitive it changes to *pojan* resulting in `_poj` and `jan_`, two different symbols.

This approach produces more symbols for each sentence than the BPE approach. As an example, Firat et al. (2016) got 43.67M Finnish tokens from a 2.03M sentence dataset, i.e. 21.5 symbols per sentence, while our method produces 52.0 symbols. For English and German they got 26.9 and 28.3 respectively, while we get 47.9 and 45.2. This increased number of symbols helps specially with small datasets and languages with many one-syllable morphemes.

3.2 Auxiliary language parallel data

We train a model on an auxiliary language pair which shares one of its languages with the target language pair together with the main model. Both models, the target translation model and the auxiliary translation model, are trained to minimize the Negative Log Likelihood (NLL) on their corresponding datasets. In order to train both of them jointly, for each of them, the mean NLL and the gradients with respect to each parameter are computed on a minibatch. The weighted sum of these gradients are further used by ADADELTA (Zeiler, 2012) to update the weights.

To prevent one of the models' updates from outweighing the other's, gradients of shared parameters are scaled based on their previous d costs, in such a way that the weakest model's gradients get promoted. The intuition behind this is that, if they are generalizing correctly, the main model and the auxiliary model should produce similar costs, as they are defined in a similar fashion. If a model performs better than the other we can decelerate its optimization while accelerating the other to balance them properly. The auxiliary model has less risk of overfitting, as it trains on a bigger dataset. Therefore, if the main model produces costs similar to those produced by the auxiliary model it is generalizing better than when the cost function evaluates lower. The parameters that are not shared by the two models are not scaled.

The scale factor at epoch t for the gradient's of the auxiliary model s_t^{aux} and the target model s_t^{tgt} are computed as follows:

$$s_t^{aux} = \left(\frac{1}{2} - \frac{1}{1 + e^{(\mu_t^{tgt} - \mu_t^{aux})s}}\right) l + \frac{1}{2}, \tag{12}$$

$$s_t^{tgt} = 1 - s_t^{aux}; \tag{13}$$

where s and l are hyperparameters that control the *steepness* and he *range* of the function, respectively. μ_t^{tgt} and μ_t^{aux} are the mean of the last d costs for the target model and the auxiliary model, calculated as follows:

$$\mu_t^{\mathrm{m}} = \sum_{j=t-d}^{t} J_j^m, \tag{14}$$

where $m \in \{\mathrm{t}gt, \mathrm{a}ux\}$ and J_j^m is the cost computed by model m at timestep j. The pseudo-code for this algorithm can be seen in Algorithm 1 and the shape of Equation 12 in Figure 1.

We group sentences of similar length in minibatches so that each minibatch contains roughly the same number of symbols (in our experiments, no more than 4000 symbols per batch). Therefore, minibatches of longer sentences contain less sentence pairs. Because each minibatch can contain a different number of sentences and different number of words per sentence, averaging over words is necessary. Our cost function averaged the cost of every word in a sentence in every sentence in a minibatch. The cost of a word was measured as Negative Log Likelihood,

$$J_t^m(X_t, Y_t) = \frac{1}{N_t} \sum_{i=1}^{N_t} \left(\frac{1}{\mathrm{length}(Y_t^{(i)})} \sum_{j=1}^{\mathrm{length}(Y_t^{(i)})} -\log P_\theta(Y_t^{(i,j)} | X_t) \right), \tag{15}$$

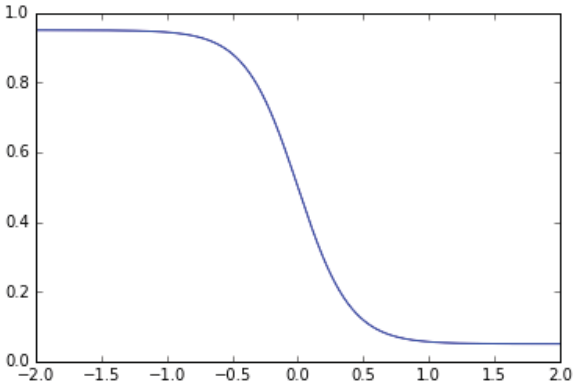

Figure 1: Equation 12 with parameters $s = 5$ and $l = 0.9$

where X_t and Y_t are the source and translation sentences in the minibatch at timestep t, N_t is the number of sentence pairs in the minibatch and length$(Y_t^{(i)})$ is the length in symbols of the the translation i at timestep t.

In Figure 2 we can see how the costs from the German-English and the French-English models descend at the same pace. The s hyperparameter was set to 50. The weighted sum of these gradients are further used by ADADELTA Zeiler (2012) to update the weights.

3.3 Monolingual data

Monolingual data can be leveraged in a similar way to auxiliary data. We input monolingual sentences as both, input and expected output, in the manner of an auto-encoder (e.g. De→De).

In contrast with the data from an auxiliary pair the model for the monolingual data is trained after the target model has converged. The encoder and the decoder are trained separately. First the non-shared part (either encoder or decoder) is trained until convergence and then the rest of the model is trained using Equation 12 to scale the updates.

We used different parameters for Equation 12 for the auxiliary language-pair data and the monolingual data.

This kind of model, similar to an auto-encoder, is expected to quickly memorize a copying mechanism from source to target. The method here described can slow down this memorization.

4 Evaluation

4.1 Data and Methods

The datasets used in the experiments are similar to some described in Firat et al. (2016). We performed two sets of experiments: one for the case where the auxiliary language is in the source (De+Fr → En) side and the other one for the case where the auxiliary language is in the target side (En → Fi+Fr). In both cases French is the auxiliary language. The parallel data is from the datasets available for WMT'15 for each language pair. We randomly picked 100k and 200k sentences for the En-Fi case and 210k and 420k sentences for the De-En case. The development and test sets are for En-Fi were `newsdev-2015` and `newstest-2015` respectively; and for De-En, `newstest-2013` and `newstest-2015` respectively.

All the sentences were tokenized using the `tokenize.perl` script included in Moses and then cleaned using the scripts `normalize-punctuation.perl`, `remove-non-printing-char.perl` and `deescape-special-chars.perl`. In-

Algorithm 1 Training

procedure TRAIN
  $\texttt{costs\_aux}, \texttt{costs\_tgt} \leftarrow ([], [])$
  $\texttt{b} \leftarrow 0.5$
  while not done **do**
    $M_{tgt}, M_{aux} \leftarrow \texttt{getNextMinibatches}()$
    $J_{tgt}, J_{aux} \leftarrow (\texttt{costTgt}(\theta_{tgt}, M_{tgt}), \texttt{costAux}(\theta_{aux}, M_{aux}))$
    for $\theta_i \in (\theta_{tgt} \cup \theta_{aux})$ **do**
      if $\theta_i \in (\theta_{tgt} \cap \theta_{aux})$ **then**
        $g_{tgt} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costTgt}(\theta_{tgt}, M_{tgt})$
        $g_{aux} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costAux}(\theta_{aux}, M_{aux})$
        $\texttt{g} \leftarrow \texttt{b} \cdot g_{aux} + (1 - \texttt{b}) \cdot g_{tgt}$
      else if $\theta_i \in \theta_{tgt}$ **then**
        $\texttt{g} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costTgt}(\theta_{tgt}, M_{tgt})$
      else
        $\texttt{g} \leftarrow \frac{\partial}{\partial \theta_i} \texttt{costAux}(\theta_{aux}, M_{aux})$
      $\theta_i \leftarrow \texttt{applyADADELTA}(\theta_i, g)$
    push J_{tgt} onto $\texttt{costs\_tgt}$
    push J_{aux} onto $\texttt{costs\_aux}$
    if $|\texttt{costs\_aux}| > d$ **then** shift $\texttt{costs\_aux}$
    if $|\texttt{costs\_tgt}| > d$ **then** shift $\texttt{costs\_tgt}$
    $(\mu_{aux}, \mu_{tgt}) \leftarrow$ (mean($\texttt{costs\_aux}$), mean($\texttt{costs\_tgt}$))
    $\texttt{b} \leftarrow \left(\frac{1}{2} - \frac{1}{1+e^{(\mu_{tgt} - \mu_{aux})s}}\right) l + \frac{1}{2}$

stead of space separated words, we used the subword units described in Section 3.1 with a vocabulary size of 30k symbols. Unlike the BPE subword method used in Firat et al. (2016) our method allows for *UNK* symbols. The amount of these symbols and other statistics of the datasets can be seen in Table 1.

Twelve models were trained, three on each of the four described datasets. We first trained four models using only the described subword approach, without any additional data. Then, we also trained four models using the auxiliary data. Finally, we further train the models from the previous step using the monolingual data in addition to the main and auxiliary datasets.

All the models had the same number of parameters in their encoders and decoders.

We evaluate the performance of the trained models based on their BLEU score using the `multi-bleu.perl` script from Moses. We merged the subword tokens into words before evaluation and computed the score on lowercase text.

4.2 Implementation

The code[1] for the proposed method was implemented in Python using Theano (2016). It runs on a single GPU computing the cost and gradients for each language pair one at a time. This could be parallelized using an extra GPU to speed the training up. We used three different models of GPU for training: *GeForce GTX 980 Ti*, *Tesla K40m* and *GeForce GTX TITAN X*.

We used a vocabulary size of 30k symbols for each language. All the hidden layers had 1,000 units. The embeddings for the each symbol were 620 dimensions long. The attention vectors from the bidirectional RNNs where projected into 1,000 dimension vector as described

[1] Code available at `https://github.com/basaundi/amta2016`

	role	sentence pairs	symbols		UNK	
			src	tgt	src	tgt
en-fi	train	100k	4.7m	5.2m	11.6k (0.2%)	39.6k (0.7%)
	train	200k	9.3m	10.4m	23.3k (0.2%)	78.6k (0.7%)
	development	1500	54.2k	60.6k	904 (1.7%)	986 (1.6%)
	test	1370	45.4k	51k	940 (2.1%)	858 (1.7%)
en-fr	auxiliary	4m	222.3m	262.6m	783.6k (0.3%)	747.7k (0.4%)
fi	monolingual	8m	399.5m		3.6m (0.9%)	
de-en	train	210k	10m	9.5m	350.2k (3.5%)	90.2k (0.9%)
	train	420k	20.1m	19m	700.3k (3.5%)	180.3k (0.9%)
	development	3000	115.5k	105.6k	3641 (3.2%)	1120 (1.1%)
	test	2169	80k	75.3k	2801 (3.5%)	1330 (1.8%)
fr-en	auxiliary	4m	267.4m	227.2m	1m (0.4%)	862.3k (0.4%)
de	monolingual	8m	357m		5.2m (1.5%)	

Table 1: Statistics of the corpora used in the experiments. The symbols are subword tokens as described in this article.

	Size	BPE	Firat	SW	+Aux	+Aux +Mono
En → Fi	100k	3.93/3.42	3.21/4.2	4.17/3.89	3.81/3.74	4.54/3.99
	200k	5.21/4.79	4.16/5.71	5.28/4.70	5.15/5.08	5.63/5.32

Table 2: BLEU scores for the Finnish development and test datasets (separated by /). SW is our new subword-unit method, `+Aux` is using subwords and English-French auxiliary data and `+Aux +Mono` is using subwords with auxiliary and monolingual data.

in Equation 7.

The models are optimized using ADADELTA Zeiler (2012) with the ρ parameter set to 0.95. We clipped all the gradients to an L2 norm of 1 after weight-summing them as described in Section 3.2.

During training, a minibatch from each used dataset was fed to the corresponding computational graph. Sentences of similar length were grouped together in minibatches of no more than 40k symbols. Therefore, minibatches of longer sentences contained less sentences than those with shorter sentences.

We trained the models for a week and kept the one that performed best on the development set. For the models using monolingual data, we first trained the model on the corresponding training and auxiliary datasets. Then, we further trained the model using also the auxiliary data and monolingual data stopping after the third drop of performance on the development data. This happened quite fast, as the model memorizes the copy mechanism easily.

We used different values for hyperparameters s and l in Equation 12 for those parameters shared with the auxiliary language-pair model ($s_{aux} = 50$ and $l_{aux} = 1$) and those parameters shared with the monolingual model ($s_{mono} = 30$ and $l_{mono} = 0.9$). For both cases the number of previous costs d used for the running average was 30.

4.3 Results and analysis

The results for the experiments are summarized in Tables 2 and 3. For German, we can see that the results from Firat et al. (2016) for their original approach using the same dataset were better for the same target dataset in all cases. They didn't try English-Finnish translation with small datasets.

	Size	Single †	Firat †	SW	+Aux	+Aux +Mono
De → En	210k	14.27/13.20	16.96/16.26	15.85/14.24	16.61/15.39	16.66/15.30
	420k	18.32/17.32	19.81/19.63	18.72/17.10	18.58/17.17	18.62/17.26

Table 3: BLEU scores for the German development and test datasets (separated by /). SW is our new subword-unit method, `+Aux` is using subwords and auxiliary data and `+Aux +Mono` is using subwords with auxiliary and monolingual data. † Results from Firat et al. (2016).

When compared to the results from the single language-pair model using BPE, we can see that our approach for the subword units worked well with German. The performance gap is bigger with the smaller dataset. Our approach produces more symbols for the same dataset, which means there is more data for training. However, guessing the right word also becomes harder because more subwords need to be decoded correctly in order to get one correct word. This difference should be more noticeable for languages with longer words. In Table 5, we can see that many subwords were guessed correctly by the different models but didn't form the right word.

Using the auxiliary language pair helped for the German-English model with the small dataset. The extra data didn't help with the bigger dataset. Our guess is that one week wasn't enough for the model to benefit from all the data, as our method takes a lot of time to complete one iteration when using a single GPU. The score on the development dataset was still growing when the training was stopped. The performance for the model trained on the bigger dataset improved when trained with the extra monolingual dataset. This improvement in the performance could be an effect of the extra training time. In Table 4 we can see the translations generated by the different models to a sample sentence.

For the English-Finnish models the auxiliary data helped only for the bigger dataset because the smaller dataset overfitted for Finnish before it could use any French data. Our results are inferior to the results from the approach described by Firat et al. (2016) [2]. The auxiliary dataset helped better when the auxiliary language was in the encoder (i.e., the language in the decoder was repeated).

In Figure 2 we can see how the NLL cost descended at the same pace for both German-English and French-English effectively preventing the model from overfitting for the smaller dataset. In our experiments we observe that the BLEU score on the validation set starts to decrease when the cost goes under 1.0 when using our proposed subword units.

Introducing the monolingual data helped prevent the English-Finnish models from overfitting but wasn't very helpful for German. In Figure 3 we observe how, without the additional data, the model over-fits and prepends an unnecessary adjective to the word Ukraine associated to *school* and *Ukraine*. The incorrect translation means "All children return to latter Ukraine".

The proposed subword-units are able to generate new words that do not appear in the training or auxiliary dataset by analogy. As an example, the word *biometrically* was guessed correctly by the German-English model even though this word appears in the test set for the first time and the English-Finnish model could generate the word *liitoksen* (*liitos* + *GEN*, of the annexation).

5 Conclusion

We evaluated three different ideas that could help improve NMT for language pairs with limited resources. We trained three models applying from one to the three of the ideas on four different datasets and assessed them by measuring their BLEU scores on the same test-set.

[2] Computed using the code at `https://github.com/nyu-dl/dl4mt-multi`

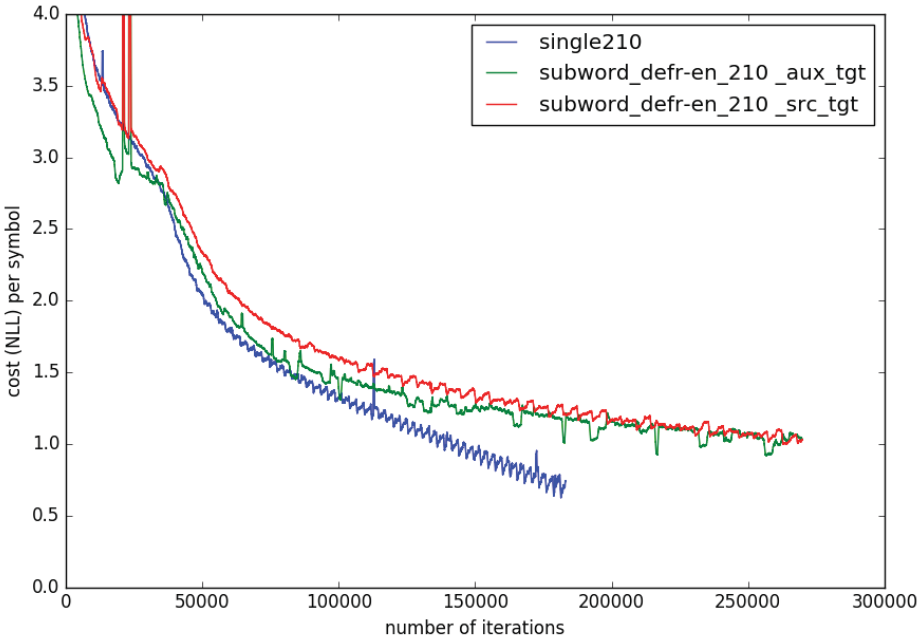

Figure 2: NLL-per-word cost evolution for a model trained on a single language pair (de-en) and one trained using an auxiliary language pair. The cost for the model trained without auxiliary data sinks because of overfitting. For the other model, the costs of the target language-pair (upper) and the auxiliary language-pair (lower) descend at the same pace.

The first idea was to use subword-units instead of words. The subword-units we applied were intended to map better to single morphemes when compared to other methods as BPE. Our experiments showed that these kind of subword-units can help with smaller datasets, getting a +1.24 BLEU score improvement when compared to BPE for German-English translation when trained on a 210k sentence pair dataset.

Using data form an auxiliary language-pair helped improve the performance for small datasets (about 200k parallel sentences) but when the dataset was too small (100k parallel sentences) and the auxiliary language was in the decoder side the model ended memorizing the small dataset despite the extra data. Even though the extra data improved the performance the solution by Firat et al. (2016) got better results for German-English translation.

The monolingual data helped prevent overfitting in the cases when the dataset in the decoder side was too small. The monolingual data didn't help very much with German-English translation. Our proposed solution increased considerably the time needed for each iteration and thus the time for convergence.

In the future, we would like to rethink the subword-unit approach to represent better the consonant-gradation and other small sound changes related to morpheme combination. Also, faster GPUs may make our solution more feasible in the future.

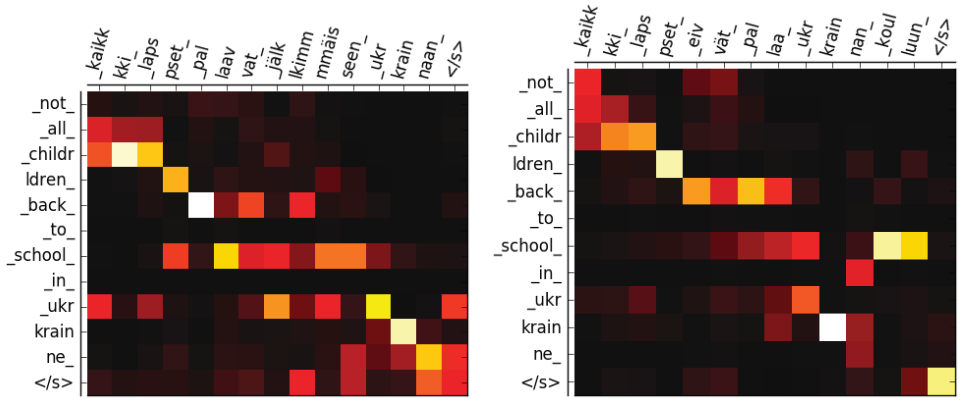

Figure 3: Alignments for English-Finnish translations by the ”SW” (left) and ”+Aux+Mono” models (right). The first model prepends the adjective ”jälkimmäiseen” (latter) to Ukraine and omits the words *eivät* (not) and *koulu* (school).

source	die premierminister indiens und japans trafen sich in tokio .
reference	india and japan prime ministers meet in tokyo
subword 210k	the prime minister of india and japan in tokyo in tokyo .
+aux 210k	the prime ministers of india and japan met came in tokyo .
+aux+mono 210k	the prime minister of india and japan came to tokyo .
subword 420k	prime minister india and japan met in tokio .
+aux 420k	the prime minister of india and japan joined in tokyo .
+aux+mono 420k	the prime minister , india and japan , met in tokyo .

Table 4: Sample translations of the first German sentence in the test set produced by the different models.

source	the organisations have promised a career solution by the end of autumn , but according to the latest estimation it would be achieved this week .
reference	järjestöt ovat luvanneet työuraratkaisun syksyyn mennessä , mutta tuoreimman arvion mukaan se syntyisi tämän viikon aikana .
subword 100k	järjestöt ovat luvanneet erinomaisen urauden loppuun saattamisen loppuun saakka , mutta myöhemmin tällä viikolla toteutetulla arviolla voitaisiin saavuttaa tämän viikon kuluessa .
+aux 100k	järjestöt ovat luvanneet suorituskeskustelun jälkeen syksyn loppuun mennessä , mutta viimeisimpien arvioiden mukaan tämä ehdotus voitaisiin .
+aux+mono 100k	järjestöt ovat luvanneet tehtyä urakentamalla syksyn loppuun mennessä , mutta viimeisimpänä arvUNKstana se olisi saavuttanut tämän viikon kuluesss .
subword 200k	järjestöillä on luvattu uralla uratkaisu vuoden loppuun mennessä , mutta viimeisimmän arvion mukaan se olisi saavutettavissa .
+aux 200k	järjestöt ovat luvanneet saavutettavan uran loppuun mennessä , mutta viimeisten arvioiden mukaan tämä olisi mahdollista saavuttaa tällä viikolla .
+aux+mono 200k	järjestöt ovat luvanneet suoran ratkaisun syksyn loppuun mennessä , mutta viimeisin arvioitu olisi viime viikolla saavutettavissa .

Table 5: Sample translations into Finnish. Many subwords were guessed correctly but didn’t form the correct word.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. arXiv: 1409.0473.

Blunsom, P. and Kalchbrenner, N. (2013). Recurrent Continuous Translation Models. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*. arXiv: 1406.1078.

Crego, J. M., Max, A., and Yvon, F. (2010). Local lexical adaptation in Machine Translation through triangulation: SMT helping SMT. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee.

Deoras, A. (2011). RNNLM - Recurrent Neural Network Language Modeling Toolkit. *Microsoft Research*.

Dholakia, R. and Sarkar, A. (2014). Pivot-based triangulation for low-resource languages. In *Proc. AMTA*, pages 315–328.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-Task Learning for Multiple Language Translation. In *ACL*.

Firat, O., Cho, K., and Bengio, Y. (2016). Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Gage, P. (1994). A New Algorithm for Data Compression. *C Users Journal*, 12(2):23–38.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Nakov, P. I. and Ng, H. T. (2012). Improving Statistical Machine Translation for a Resource-Poor Language Using Related Resource-Rich Languages. *Journal of Artificial Intelligence Research*, 44. arXiv: 1401.6876.

Sennrich, R., Haddow, B., and Birch, A. (2015a). Improving Neural Machine Translation Models with Monolingual Data. *arXiv:1511.06709 [cs]*. arXiv: 1511.06709.

Sennrich, R., Haddow, B., and Birch, A. (2015b). Neural Machine Translation of Rare Words with Subword Units. *arXiv:1508.07909 [cs]*. arXiv: 1508.07909.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Theano, D. T. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688 [cs]*. arXiv: 1605.02688.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv:1411.1792 [cs]*. arXiv: 1411.1792.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*. arXiv: 1212.5701.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer Learning for Low-Resource Neural Machine Translation. *arXiv:1604.02201 [cs]*. arXiv: 1604.02201.