Probabilistic Inference for Phrase-based Machine Translation: A Sampling Approach

Abhishek Arun

Doctor of Philosophy

Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh

2011

Abstract

Recent advances in statistical machine translation (SMT) have used dynamic programming (DP) based beam search methods for approximate inference within probabilistic translation models. Despite their success, these methods compromise the probabilistic interpretation of the underlying model thus limiting the application of probabilistically defined decision rules during training and decoding.

As an alternative, in this thesis, we propose a novel Monte Carlo sampling approach for theoretically sound approximate probabilistic inference within these models. The distribution we are interested in is the conditional distribution of a log-linear translation model; however, often, there is no tractable way of computing the normalisation term of the model. Instead, a Gibbs sampling approach for phrase-based machine translation models is developed which obviates the need of computing this term yet produces samples from the required distribution.

We establish that the sampler effectively explores the distribution defined by a phrase-based models by showing that it converges in a reasonable amount of time to the desired distribution, irrespective of initialisation. Empirical evidence is provided to confirm that the sampler can provide accurate estimates of expectations of functions of interest. The mix of high probability and low probability derivations obtained through sampling is shown to provide a more accurate estimate of expectations than merely using the n-most highly probable derivations.

Subsequently, we show that the sampler provides a tractable solution for finding the maximum probability translation in the model. We also present a unified approach to approximating two additional intractable problems: minimum risk training and minimum Bayes risk decoding. Key to our approach is the use of the sampler which allows us to explore the entire probability distribution and maintain a strict probabilistic formulation through the translation pipeline. For these tasks, sampling allies the simplicity of n-best list approaches with the extended view of the distribution that lattice-based approaches benefit from, while avoiding the biases associated with beam search. Our approach is theoretically well-motivated and can give better and more stable results than current state of the art methods.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Abhishek Arun*)

Acknowledgements

I would like to begin by thanking my supervisor, Philipp Koehn, for his invaluable guidance and unwavering support throughout my graduate life. This thesis would not have been possible without his faith in me.

I would also like to acknowledge the faculty at the University of Edinburgh: my second supervisor, Miles Osborne, who introduced me to Statistical NLP during my M.Sc and whose door has always been open for me; Frank Keller, who got me excited about research and then allowed me to pursue my academic interests; and my committee members: Mirella Lapata, Sharon Goldwater and Charles Sutton, for providing valuable advice.

This work has been greatly facilitated by the assistance of members, past and present, of the University of Edinburgh's Statistical Machine Translation group. I would especially like to acknowledge the contribution of Barry Haddow, Phil Blunsom, Chris Dyer and Adam Lopez, my brilliant team members at the MT Marathon in Prague where the ideas presented in this thesis first began to take shape. This thesis would have taken much longer without Barry's support.

I have been very fortunate to have had wonderful office mates in Edinburgh. I had many fun and enlightening discussions with Sebastian Riedel and James Clarke in our time at Buccleuch Place. In the Informatics Forum, I was very lucky to share an office with Hieu Hoang, who was ever so generous with his time answering my all too frequent questions about Moses.

Kudos to my flatmates Dave and Helen and Alex for putting up with me and my taste in music all these years and to my friends Gregor and Paul who I could always rely upon to entertain me whenever i needed a break from work. Many thanks also to the dearly missed Penny, Ashwin, Jacob, Laura and Howard for providing me a home away from home whenever I needed to get away from Edinburgh.

Finally, thank you to my family for their constant love, support and encouragement.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Statistical Machine Translation (SMT; Brown et al. (1990); Och and Ney (2002)) is a data-driven approach which treats translation as a machine learning problem. Given a source sentence as input, a function which maps each possible output target language sentence to a real valued score is developed. The score is based on predefined *features* which identify characteristics of the input/output pairs that are indicative of whether the output is good or not. Each feature is weighted by a corresponding parameter. During a *training* phrase, the parameters are adjusted so that the function assigns high scores to good outputs and low scores to bad outputs. The translation of an input sentence is the output with the highest score. Since there are usually many possible outputs, finding the highest scoring one requires a search component. The search process is also known as *decoding*.

A *model* describes the scoring function, the set of features, the set of corresponding parameters and the set of rules used to transform an input sentence into a output sentence. A sequence of rule application which translates the entire input sentence is commonly referred to as a *derivation*. Multiple derivations can map to the same output string or *translation*. An SMT model is *probabilistic* if it assigns a non-negative score to each derivation and if the scores of all the derivations sum up to 1.

Probabilistic models have many advantages. In the training phase, we can tune the model parameters by using well-motivated algorithms such as conditional likelihood training (Lafferty et al., 2001) or minimum *risk* training (Smith and Eisner, 2006). These training algorithms involve an optimisation problem which can be solved efficiently using powerful numerical methods which scale to a large number of features.

Maximum a posteriori (MAP) decoding finds the most probable output string. While strictly speaking MAP decoding does not require that the base model be probabilistic, a probabilistic model allows for alternate decoding techniques. An example is minimum Bayes risk (MBR; Kumar and Byrne (2004)) decoding, a technique which uses the notion of risk for making optimal decisions, often outperforming MAP decoding (Tromble et al., 2008; Kumar et al., 2009).

Finally, probabilistic models can be used to obtain confidence measures which might be needed for downstream processing tasks. They can also be combined together in a product-of-experts or can be easily chained e.g. we can build a speech to speech translation system by concatenating a probabilistic speech recogniser and a probabilistic speech synthesiser at each end of a probabilistic SMT model.

An SMT model which produces a finite number of derivations when translating an input sentence can be converted into a probabilistic one by first exponentiating the score of every derivation (this ensures that every score is non-negative) and then normalising the resulting score by the sum of the exponentiated scores of each derivation. The latter ensures that all scores sum up to 1 and that therefore the model is a probability distribution over outputs given an input. A probabilistic model which uses exponentiated weighted features to model the conditional probability of outputs given an input is often referred to as a *log-linear model*. SMT models define a conditional distribution over *derivations* given an input. By summing up the probabilities of derivations yielding the same translation, a conditional model over *translations* is obtained.

There is a vast number of possible translations allowed by the model for any input sentence. In most models, computing the normalisation term for all but short inputs is too expensive to be practical, i.e. the computation is *intractable*. As a result, current state of the art SMT systems have eschewed the calculation of the denominator in the log-linear model and have instead used an *unnormalised* linear model (Och and Ney, 2002). Making exact decisions in such models remains intractable and efficient bespoke approximate algorithms have been designed for *inference* tasks such as decoding and training. For instance, approximate polynomial-time algorithms for MAP decoding have been developed by placing restrictions on the feature space and by substituting the search for the most probable translation with a search for the most probable derivation. The most commonly used algorithm is beam search (Koehn, 2004a) which performs tractable decoding by *heuristically pruning* the search space. At training time, the most commonly used algorithm tunes the parameters such that

the derivation considered the "best" by a given error function is also the most probable derivation given the current parameters. The optimisation of this algorithm is made efficient by exploiting characteristics of the search and feature spaces (Och, 2003). SMT systems have gained a lot in terms of efficiency by jettisoning the probabilistic interpretation of the model. This increased efficiency has meant that SMT systems have been able to scale to massive datasets with a resulting increase in translation quality.

Probabilistic inference techniques offer the promise of training better models. For example, while minimum error rate training (Och (2003), MERT), the most prevalent training algorithm, is efficient, it is unstable (Foster and Kuhn, 2009) and can only be applied to a handful of features (Och et al., 2004). This limitation of MERT has hindered the exploration of potentially informative features. In fact, success in scaling linear models of SMT to a large number of features (Watanabe et al., 2007; Chiang et al., 2008b, 2009) has only recently been achieved. Even then, the number of features used in these models is only of the order of thousands. On the other hand, the ready availability of general purpose probabilistic optimisation algorithms has enabled the few existing normalised conditional probability models of translation (Ittycheriah and Roukos, 2007; Blunsom et al., 2008) to successfully scale to millions of rich features.

Probabilistic models also allow for risk minimisation techniques which have shown to improve translation performance when applied during training (Li and Eisner, 2009) and during decoding (Tromble et al., 2008; Kumar et al., 2009).

The latter methods have performed probabilistic inference by approximating the space of all derivations with a subset of high scoring derivations. In fact, due to the intractability of computing the normalisation term of the log-linear model, *any* attempt at probabilistic inference in SMT models will have to resort to approximations: the success of any approach is likely to hinge on the quality of the approximations used. Existing methods have approximated the exponential space of all derivations by first running beam search and then mining a compact polynomial space representation of the resulting pruned search space; the early methods of (Kumar and Byrne, 2004; Smith and Eisner, 2006) used the n most probable derivations drawn from this space as proxy to the entire space, whereas more recent work has seen the development of specialised polynomial time algorithms (Tromble et al., 2008; Li and Eisner, 2009) to exploit the entirety of the unpruned space leading to performance improvements.

Both minimum risk training and decoding involve the calculation of *expectations*. The use of beam pruning to approximate the required expectations however introduces

the concern that the resulting approximations might be arbitrarily biased (Bouchard-Côté et al., 2009). This is because the approximation is likely to be too concentrated around the mode of the distribution (Blunsom and Osborne, 2008). In the case of minimum risk training, the expectation required is that of the values of the features of the model. While the bias resulting from beam pruning might not be noticeable in current SMT models which only use a few features that are all active on every derivation, it is likely to matter more as future models scale to using a large number of features where only a minimal subset might be active on any given derivation: a feature with low expectation may be completely skipped if its supporting derivation is pruned.

In this thesis we introduce a novel Markov Chain Monte Carlo (MCMC) sampling framework for theoretically sound approximate inference in SMT models. The general idea behind MCMC methods is to draw samples from the distribution of interest after which the generated samples can be used for a number of inference tasks such as the calculation of expectations of values of interest. In this thesis, the distribution we are interested in is the conditional distribution of a log-linear translation model; however, recall that often there is no tractable way of computing the normalisation term of the model. Instead, we use a Gibbs sampling (Geman and Geman, 1984) approach which obviates the need of computing this term yet produces samples from the required distribution. Theoretically, the Gibbs sampling technique we present is general purpose in that it can be applied to a variety of SMT models such as phrase-based (Koehn et al., 2003) or syntax-based (Chiang, 2005) translation models. In this thesis, we develop a Gibbs sampler for phrase-based models since they remain the state of the art approach for modeling translation for many language pairs.

After showing that the Gibbs sampler is able to closely approximate the true distribution, we apply this framework as a tractable solution for a variety of inference tasks. For example, it can be employed for performing unbiased minimum risk training. Since the model is probabilistic, general purpose optimisation algorithms which scale to a large number of features can be used. The probabilistic interpretation of the model restored, the sampler facilitates MBR decoding. It also enables searching for the most probable translation.

We perform extensive experimentation to compare the performance on standard decoding tasks of a sampling-based pipeline with a beam search one. We find that the sampler may obtain results as good, or better, than beam search and that these results are often more stable. However, sampling is slower than current training and

decoding algorithms. As an alternative, we suggest using sampling for training and the MBR techniques of (Tromble et al., 2008; Kumar et al., 2009) for decoding, a compromise solution which maintains, and sometimes improves, the accuracy of the sampling pipeline while reducing decoding time.

1.2 Thesis Contributions

The major contributions of this thesis are as follows:

- We introduce a novel Gibbs sampling based approach for probabilistic inference in phrase-based translation models.
- We show that sampling provides a tractable solution for finding the maximum probability translation in the model.
- We perform unbiased minimum risk training and decoding using the sampler and find that this unified probabilistic risk minimisation approach can improve upon state of the art SMT models.

1.3 Structure of the Thesis

Chapter 2 provides a detailed survey of contemporary machine learning based approaches to SMT. We focus our review on probabilistic-based techniques for parameter estimation and decoding which motivate the work presented in this thesis.

Chapter 3 introduces sampling methods in general and Markov Chain Monte Carlo techniques in particular as theoretically well-motivated approaches for approximating expectations of functions of interest in distributions which preclude exact inference. In addition to the theoretical properties of these techniques, we also consider the practical aspects of designing and running a sampler including choices of proposal distributions, ways to improve mixing of the sampler, convergence diagnostics and variance reduction methods. Finally, we review prior sampling-based approaches to problems in natural language processing (NLP) and statistical machine translation.

Chapter 4 describes our experimental setup. We give details about our baseline model and on the corpora used for our experiments.

Chapter 5 presents a novel Gibbs sampler for phrase-based machine translation. We describe the block sampling based approach used for efficient sampling and provide a

rigorous formal description of the sampler as well as proofs for its correctness. We draw attention to cases in which the proofs of the correctness do not hold and describe methods to address this flaw in the sampler's design. We also discuss the algorithmic complexity of the sampling procedure and explain how the sampler is run in practice. Finally, we provide empirical evidence of the sampler's ability to effectively explore the probability space of a phrase-based translation model.

Chapter 6 discusses how the sampler described in the previous chapter can be used for two intractable decoding tasks, namely MAP and MBR decoding. We run extensive decoding experiments examining a number of factors which affect decoding quality including the scaling of the model parameters, the number of samples collected and the method with which the sampler is initialised. Results show the sampler performs best as an MBR decoder.

Chapter 7 shows how the sampler can be used to tune the model parameters using minimum risk training. A key quantity required for this training algorithm is the expected value of the features in the model. We experimentally compare a sampling-based approach for computation of this expectation with two beam search based approaches. We then propose two variants of minimum risk training. The first variant, sentence sampling, optimises an objective defined at the sentence level. In our experiments, we use a sentence level approximation to BLEU for risk computation. The second variant is *corpus sampling* which optimises a corpus-based objective and allows the direct use of BLEU in the calculation of the objective function. We also present ways to improve the optimisation of the non-convex minimum risk training objective function. We end the chapter by presenting results which compare translation performance using minimum risk trained weights with the alternate MERT optimisation technique.

Finally, in Chapter 8 we conclude the thesis by highlighting the major findings, and suggesting future research directions.

1.4 Published Work

This thesis is based on three publications. Chapter 5 extends the material presented in Arun et al. (2009) and Arun et al. (2010a) by providing a more thorough description of the sampler and by presenting experiments analysing the sampler's convergence.

Chapter 6 expands selected sections from both Arun et al. (2009) and Arun et al. (2010a) by giving more details about decoding experiments.

Finally, Chapter 7 elaborates on Arun et al. (2010b). It provides more analysis on approximating feature expectations, contains additional information on the methods used for parameter estimation, and details many more experiments.

Chapter 2

Statistical Machine Translation

Machine Translation (MT) is the task of using computers to automatically translate one natural language into another. The field of MT dates from the late 1940s when the success of computers to crack ciphers during World War II led to the hope that they could also be used for automatic translation. Statistical machine translation (SMT) is a data-driven paradigm which applies statistical learning methods to a *parallel corpus* or *bitext* consisting of sentences in a source language and their translation in the target language after which it can be used to translate seen or unseen sentences in the source language into sentences in the target language. SMT was first introduced by Brown et al. (1993) and rapidly became the dominant MT research methodology. In addition, the last few years have seen high-profile commercial MT systems such as Google Translate and Microsoft Translator switching from rule-based systems to SMT.

In the SMT framework, given a source (Foreign) sentence **f**, the translation problem is to generate a target (English) sentence **e** that maximises a scoring function **s** which is dependent on **f** and the generated **e**. The transformation of the source string into a string in the target language is governed by a *translation equivalence model* (Lopez, 2008), which describes a series of steps that perform this transformation.[1] Translation is often ambiguous, for example there may be multiple ways of translating the same word, so during the course of the transformation the model needs to have a mechanism to resolve ambiguity. This mechanism is provided by the *parameterisation* of the translation equivalence model. The parameterisation defines a number of knowledge sources also known as *parameters* or *features* which are then used to assign scores to each decision made during the translation process. The parameterisation also defines

[1]The translation equivalence model is also simply called the translation model. However, since the latter term is often also employed to describe a specific parameter in the SMT model, we use the expression translation equivalence model to refer to the general translation model.

how the features are *estimated* and how they are *combined* together to produce scores. We group together the translation equivalence model and its parameterisation under the heading *model* which we describe in detail in Section 2.1.

A good model is one in which good translations are scored higher than bad translations. But how can we assess whether a given target language string is a good or a bad translation of the source sentence? In a system development setup, human evaluation is impractical because it is too slow and too costly. As an alternative, a number of *automatic evaluation metrics* have been proposed. The development of metrics that correlate well with human judgements remains a very active field of research. We review some of the existing metrics in Section 2.2.

Once we have an adequate automatic evaluation method, we can use machine learning algorithms that automatically *tune* or refine our model such that it assigns higher scores to good translations than to bad ones. This tuning step is also referred to as *parameter estimation* and is discussed in Section 2.5.

Given a suitably tuned model, we are now ready to translate the new sentence $\mathbf{f}$. *Decoding* consists of finding the most likely target sentence from the space of all translations allowed by the model. The concept of most likely sentence is quantified by a *decision rule* (see Section 2.3). The number of possible target language translations that can be generated by the model is usually exponential in the length of $\mathbf{f}$. To efficiently explore this search space for the most likely translation, we require a *search algorithm*. In Section 2.4, we review a number of such algorithms.

2.1 Model

2.1.1 Models of Translation Equivalence

The first component of an SMT model is a translation equivalence model which is essentially the set of rules that map a source sentence into a target sentence. The rules are typically induced directly from a parallel corpus of the relevant languages, in which case the corpus can be referred to as a training corpus.

Translation equivalence models can range from being coarse-grained to being fine-grained. An extreme example of a coarse-grained model is one with rules that simply encode how whole sentences translate. These types of rule can translate previously seen sentences perfectly but do not *generalise* to unseen sentences. An example of a fine-grained model is a word-based model where the rules describe how individual

source language words translate. Assuming all the words in an unseen source sentence have been seen in the training corpus, the model will be able to translate the sentence. The seminal IBM models for SMT developed by Brown et al. (1990, 1993) are word-based models. The most simple IBM model, called Model 1, makes the simplifying assumption that each source word independently generates a target word. Of course, Model 1 is an inadequate representation of the true translation process. In reality, a source word can translate to 0 or many target words and conversely a target word can be generated by 0 or many source words. Additionally, there is no guarantee that words in appear in the same order in the target as in the source, that is, words can be *reordered*. Any model of translation needs to address the issue of reordering.

Brown et al. (1993) present systems of increasing complexity to better model word-based translation. Model 2 addresses reordering by introducing an absolute alignment (or distortion) model based on the positions of source and target words. The fertility model in Model 3 models how many target words are generated by a given source word. Model 4 and 5 improve reordering by replacing the absolute alignment model with a relative alignment one.

Even the most sophisticated IBM word-based models fail to correctly model the case of many words in the source translating as a unit into many words in the target. This many-to-many phenomenon is most clearly evident in the case of translations of idiomatic or similar multi word expressions. *Phrase-based models* (Och et al., 1999; Marcu and Wong, 2002; Koehn et al., 2003) address this problem and advanced the state of the art by moving from using words as the basic unit of translation to using phrases. Phrases in this context are simply consecutive sequence of words. They allow the translation models to learn local reordering and idioms, and account naturally for the insertion and deletion of words in a local context, something that word-based models have to model using notions of fertility and distortion. An example of a translation produced by a phrase-based model is shown in Figure 2.1. Notice that the first 3 words of the input French sentence are translated as one phrase and that the position of the source word "*contenu*" is reordered in the target translation. Phrase-based models address the issue of local reordering; however, they offer no principled way of dealing with long-distance reordering except for the relatively weak methods proposed in the higher order IBM models.

The need for a mechanism to adequately handle long distance reordering motivates the development of syntax-based models (Chiang, 2005, 2007; Quirk et al., 2005;

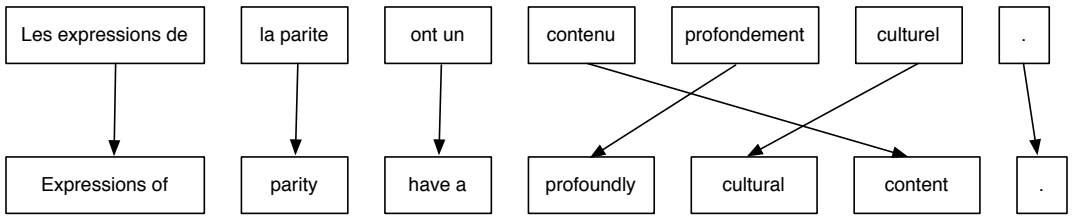


Figure 2.1: An example translation using a phrase-based model.

Marcu et al., 2006; Galley et al., 2006). A detailed exposition of these models is beyond the scope of this thesis; interested readers are referred to Lopez (2008).

2.1.2 Model Parameterisation

The second component of an SMT model is its parameterisation which allows the model to assign a score to every pair of source and target sentence. In SMT, this parameterisation is in terms of a *joint probability distribution* $p(\mathbf{e},\mathbf{f})$ over source sentences $\mathbf{f}$ and target sentences $\mathbf{e}$. At translation time, we are interested in the probability of an output string $\mathbf{e}$ given a fixed input string $\mathbf{f}$ given by the *conditional distribution* $p(\mathbf{e}|\mathbf{f})$.

In reality, the use of translation equivalence models to guide the production of output $\mathbf{e}$ given $\mathbf{f}$ as input also introduces in the parameterisation a *derivation* variable $\mathbf{d}$, which is a mapping from $\mathbf{f}$ to $\mathbf{e}$ indicating the rules used by the translation equivalence model.

For the phrase-based translation example in Figure 2.1, $\mathbf{f}$ denotes the source sentence in French, $\mathbf{e}$ refers to the produced English translation and $\mathbf{d}$ corresponds to the sequence of translation rules, illustrated as arrows from French phrases to English phrases, used to produce the translation.

At this point, it is necessary to introduce some terminology which we will use during the course of this document. Let $D(\mathbf{f})$ denote the set of derivations that can be generated given the translation equivalence model and input sentence $\mathbf{f}$. Each derivation $\mathbf{d} \in D(\mathbf{f})$ can be mapped to a string $\mathbf{e}$ in the target language using a *yield* function Y: $\mathbf{e} = Y(\mathbf{d})$. The set of all derivations yielding target string $\mathbf{e}$ given source string $\mathbf{f}$ is denoted by $D(\mathbf{e},\mathbf{f}) = \{\mathbf{d} \in D(\mathbf{f}) \text{ such that } Y(\mathbf{d}) = \mathbf{e}\}$. Given input string $\mathbf{f}$, the set of output strings that can be generated by the model is given by $T(\mathbf{f}) = \{Y(\mathbf{d}) \text{ such that } \mathbf{d} \in D(\mathbf{f})\}$. For most models, and this is certainly the case for the ones we look at in this thesis, both $|D(\mathbf{f})|$ and $|T(\mathbf{f})|$ are exponential in $|\mathbf{f}|$.

The model parameterisation therefore assigns a probability $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ to each derivation. Obtaining the term of interest, namely the probability $p(\mathbf{e}|\mathbf{f})$ assigned to a string, necessitates summing over (this is also called marginalising) all derivations that yield $\mathbf{e}$:

$$p(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{d}\in D(\mathbf{e},\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f}) \tag{2.1}$$

In most translation equivalence models, this sum involves an expensive sum over an exponential number of derivations. Therefore, most parameterisations simply model $p(\mathbf{e},\mathbf{d}|\mathbf{f})$, the distribution over derivations.

The *noisy-channel model* uses Bayes' rule to decompose the conditional distribution:

$$\begin{aligned} p(\mathbf{e},\mathbf{d}|\mathbf{f}) &= \frac{p(\mathbf{e},\mathbf{d},\mathbf{f})}{p(\mathbf{f})} && (2.2)\\ &= \frac{p(\mathbf{f},\mathbf{d}|\mathbf{e})\cdot p(\mathbf{e})}{p(\mathbf{f})} && (2.3)\\ &\propto p(\mathbf{f},\mathbf{d}|\mathbf{e})\cdot p(\mathbf{e}) && (2.4) \end{aligned}$$

Since the source sentence $\mathbf{f}$ remains fixed, the denominator can be ignored as in (2.4). The noisy channel formulation of translation is proposed by Brown et al. (1990). It decomposes the conditional distribution $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ allowing for an independent modeling of the *language model* $p(\mathbf{e})$ and the *translation model* $p(\mathbf{f},\mathbf{d}|\mathbf{e})$ - the language model being a measure of how well formed the target sentence is and the translation model measures the likelihood of the source sentence being a translation of the target sentence. Note that while we are interested in going from source $\mathbf{f}$ to target $\mathbf{e}$, in the noisy channel the reverse translation direction is modeled.

The language model is learned from large amounts of text in the target language and is usually based on n-gram frequencies and sophisticated smoothing methods which deal with data sparseness issues. The translation model is learned from parallel corpora.[2] We discuss these models in more detail in Chapter 4.

Och and Ney (2002) argue that the noisy-channel model is optimal for translation only if the true probability distributions $p(\mathbf{e})$ and $p(\mathbf{f},\mathbf{d}|\mathbf{e})$ are used. Since the used models and training methods provide only approximations of the true probability distributions, it is possible that a different combination of language model and translation

[2]Note that here translation model refers to the parameter in the SMT model.

model might yield better results. Another limitation of the noisy-channel model is that it does not offer a way to extend the baseline statistical MT model to include additional dependencies.

Instead, they suggest modelling the posterior distribution $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ directly with a *log-linear model*. This allows the use of an arbitrary set of *M feature functions* $h_m(\mathbf{e},\mathbf{d},\mathbf{f})$ defined over the input $\mathbf{f}$, the output $\mathbf{e}$ and the alignment $\mathbf{d}$ between them. The features are combined using weights λ_m which determine the contribution of each feature towards the total score.

$$p(\mathbf{e},\mathbf{d}|\mathbf{f}) = \frac{\exp\left[s(\mathbf{e},\mathbf{d},\mathbf{f})\right]}{Z(\mathbf{f})} \tag{2.5}$$

where $s(\mathbf{e},\mathbf{d},\mathbf{f})$ is an unnormalised score computed as a dot product between the features of the derivation and the model weights:

$$s(\mathbf{e},\mathbf{d},\mathbf{f}) = \lambda \cdot h(\mathbf{e},\mathbf{d},\mathbf{f}) = \left[\sum_{m=1}^{M} \lambda_m \cdot h_m(\mathbf{e},\mathbf{d},\mathbf{f})\right]$$

and $Z(\mathbf{f})$, the denominator in (2.5), is required to make the function a probability distribution. In log-linear models, the denominator is usually referred as the *normalisation constant* or the *partition function* or simply as Z. We use all three terms interchangeably in this document. $Z(\mathbf{f})$ is calculated by summing over all the derivations in the model:

$$Z(\mathbf{f}) = \sum_{\mathbf{d}' \in D(\mathbf{f})} \exp\left[s(\mathbf{Y}(\mathbf{d}'),\mathbf{d}',\mathbf{f})\right] \tag{2.6}$$

Note that this framework contains as special case the noisy-channel model if the following two feature functions are used, $h_1(\mathbf{e},\mathbf{d},\mathbf{f}) = \log p(\mathbf{e})$ and $h_2(\mathbf{e},\mathbf{d},\mathbf{f}) = \log p(\mathbf{f},\mathbf{d}|\mathbf{e})$ and if $\lambda_1 = \lambda_2 = 1$.

The log-linear formulation for SMT allows tremendous flexibility for integrating diverse knowledge sources in form of features. For example, Och and Ney report a substantial improvement in translation performance when switching from a noisy-channel model to a log-linear model to which they add additional features. Most current SMT systems use the log-linear model for parametrising the translation process due its power and simplicity. However while a vast number of arbitrary features can be added to the log-linear model, most SMT systems only use it to combine a small number of features.

2.2 Evaluation Metrics

As we saw in the previous section, a log-linear SMT model consists of knowledge sources called features and weights associated with these features. Assuming we have a means to quickly and reliably evaluate the quality of the translation output by our model, the log-linear parameterisation provides us with a way to build an SMT system iteratively. For instance, we can produce some translations with the current version of our system, evaluate its performance, alter the features and/or the weights of the log-linear model, run the modified system, and keep doing so until the performance can no longer be improved.

An obvious way to assess the quality of a translation is to judge by hand whether it is correct or not. This is called manual evaluation. However manual evaluation is an expensive process both in terms of time and money, making it unsuitable in a system development set-up. Many automatic evaluation metrics have instead been proposed in the literature. Automatic metrics work by evaluating the system output against a set of one or many human-produced reference translations. The use of the human translations as reference is motivated by the idea that a good MT output is one that closely resembles human translations of the same input sentence. A good automatic metric is one that correlates well with human judgments.

Automatic evaluation metrics are used to evaluate the final output of an SMT system or to compare outputs of different systems which, for example, might be taking part in a competition. They are also an important component of *discriminative training*. The latter refers to the phase in the SMT pipeline in which the parameters of the model are tuned using a machine learning algorithm so as to give good translation performance on a held-out set. Since the discriminative training algorithm is usually purely automated, one requires an evaluation metric that a) correlates well with human judgement and b) can be computed efficiently.

One of the first metrics to be applied to SMT was Word Error Rate (WER) (Och et al., 1999) which is based around the Levensthein distance, the minimum number of edit steps - insertion, deletions and substitutions - needed to match two sequences. WER was borrowed from Automatic Speech Recognition (ASR) evaluation but while it is suitable for that task, it is less so for SMT. This is because WER ignores word reorderings. This is not an issue in ASR where there are no reorderings, but in the case of MT where reordering is common, WER would penalise a word that is translated correctly but placed in the wrong location. A refinement of WER that does take into

account word reorderings is position-independent word error rate (PER), which was proposed by Och et al. (1999).

2.2.1 BLEU

Currently, the most popular automatic evaluation metric in the community is BLEU (Papineni et al., 2002), which has been shown to have a high correlation with human judgements. BLEU measures n-gram matches between the system output and a reference translation. It is a precision-based metric where short n-grams assess the *adequacy* of the translation in conveying the information content of the source sentence and longer n-grams measure the *fluency* of the translation in the target language.

Since BLEU uses n-gram matches to compute the goodness of a translation, it can be harsh to variability in lexical choices. An innovation of the BLEU metric is the use of multiple reference translations. The more reference translations there are, the higher the chance that an acceptable translation of an ambiguous part of the source shows up in one of the references. Indeed, the high correlation of the BLEU score with human judgements occurs when multiple references are used. The correlation is reduced or even disappears when only a single reference is used.

We now formally define the BLEU metric. For each n-gram g in a candidate translation $\mathbf{c}$, let $\#(g)$ be the count of the number of times g appears in $\mathbf{c}$ and $\#_{clip}(g)$ be the maximum number of times g appears in any of the corresponding reference translations. The n-gram precision p_n, aggregated over the *corpus* of candidate translations $\mathbb{C}$, is as follows:

$$p_n = \frac{\sum_{\mathbf{c}\in\mathbb{C}} \sum_{g\in n\text{-grams}(\mathbf{c})} \#_{clip}(g)}{\sum_{\mathbf{c}\in\mathbb{C}} \sum_{g\in n\text{-grams}(\mathbf{c})} \#(g)} \tag{2.7}$$

The n-gram precisions at each order n are combined together in a weighted geometric average. The lack of an implicit recall measure in the metric is addressed by the inclusion of a *brevity penalty*. The brevity penalty is applied if h, the overall length of the candidate translations aggregated over the corpus $\mathbb{C}$ of all candidate translations, is shorter than the *effective reference length*, r, aggregated over the corpus of reference translations.

In the single-reference case, we obtain r by simply summing up the length of each reference making up the reference corpus. In the multiple-reference case, there are several different ways for computing r:

- In the so called IBM BLEU, based on the original definition of Papineni et al. (2002), for each sentence we use the reference length which is *closest* in length to the hypothesised translation.
- In the NIST definition of BLEU, the length of the *shortest* reference sentence is used.
- A third variant uses the *average* length of the reference sentences.

Putting everything together, the brevity penalty, BP is given by:

$$BP = \begin{cases} 1 & \text{if } h > r \\ e^{1-r/h} & \text{otherwise} \end{cases} \tag{2.8}$$

and BLEU is given by:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{i}^{n} \lambda_i \log p_i\right) \tag{2.9}$$

Typically, the maximum order n for n-grams to be matched is set to 4 and the weights λ_i are set to 1.

Note that the BLEU score is *undefined* if any of the n-gram precisions is 0, meaning that no n-grams of a particular order are matched anywhere in the output. Since such a thing can happen at the sentence level, BLEU is usually computed on a *document level*.

It is often desirable to have a sentence-level metric that incorporates the properties of the BLEU metric. For example, the task of extracting an oracle translation, i.e. given a list of generated translations, finding the translation that is most similar to the reference translation, is much simplified if a metric defined on the sentence level exists.

A smoothed-BLEU implementation has been proposed in Lin and Och (2004) to compute BLEU at the sentence level. In this implementation, for n-gram orders higher than 1, add one smoothing is done to the n-gram match (numerator) and total n-gram counts (denominator). Sentences with no n-gram matches for high n-gram orders but with matches for lower n-gram orders can therefore still receive a non-zero score. The score of the sentence will be 0 only if there is no match even at the unigram level.

Criticisms

Although it is the most widely used automatic metric in MT, a number of criticisms have been voiced against BLEU. Callison-Burch et al. (2006) found a lack of correlation between BLEU scores and human judgements when comparing the outputs of a

SMT system with MT outputs produced using a rule-based MT system or a human post-edited system. They concluded that the use of BLEU scores is best suited to comparing related systems or in a system development setup where different versions of the same system need to be compared.

Further criticism of the metric is presented in Chiang et al. (2008a), drawing attention to counterintuitive results that can be obtained when using BLEU. Chiang et al. attribute these inconsistencies to the fact that BLEU is not *decomposable* at the sentence level: improving the translation of a sentence in the corpus is not guaranteed to improve the overall corpus score and conversely degrading the translation of a sentence in the corpus is not guaranteed to decrease the overall corpus score.

Why is BLEU not decomposable? Chiang et al. trace it to the way the brevity penalty is computed. Recall that the brevity penalty is a ratio of the sum of reference lengths across the corpus and the sum of the hypothesised lengths across the corpus. This means that if a system generates a too long translation for one sentence, then as long as the sum of the lengths of the two translations is not shorter than the sum of the reference lengths, the system can produce a short translation for another sentence without incurring a penalty. Whereas were the metric to be computed as a weighted average of sentence-level scores, the longer translation would not be able to compensate for the shorter translation and consequently the latter would have to face a penalty.

Going back to the oracle extraction problem mentioned previously, one consequence of BLEU's non-decomposability is that the set of oracle translations extracted on a sentence-by-sentence basis using a sentence-level approximation of BLEU is not guaranteed to be the optimal with respect to corpus-level BLEU.

2.2.2 Other Metrics

Automatic evaluation is a an active research field with many metrics proposed as alternatives to BLEU. An example alternative metric is METEOR (Banerjee and Lavie, 2005) which addresses some of the limitations of BLEU by rewarding near-matches and synonyms. This is done by first matching the proposed translation with the reference at the surface level and then backing off to stems and finally semantic classes (using WordNet synsets). Other metrics include translation edit rate (TER) (Snover et al., 2006) which measures the edit distance between the system output a human-corrected version of this output and GTM (Melamed et al., 2003) which is based around notions of *precision* and *recall* tailored for MT.

2.3 Decision Rule

At test time, the SMT system is presented with a source sentence for which it has to find the most *likely* target sentence from the space of all target sentences that can be produced by the model. This process is referred to as *decoding*. Decoding consists of two parts:

- *a decision rule* which defines the concept of most likely target sentence.
- a *search* algorithm which finds the most likely target sentence as defined by the decision rule. The choice of the search algorithm therefore depends on the chosen decision rule.

We will look at search algorithms in more detail in Section 2.4 but for now focus on decision rules. For the discussion in the rest of the chapter, we assume that the model under consideration is a probabilistic log-linear model.

2.3.1 Maximum A Posteriori Decoding

The first decision rule we consider is Maximum A Posteriori (MAP) decoding which corresponds to finding $\mathbf{e}^*_{\text{MAP}}$, the *mode* of the posterior distribution $p(\mathbf{e}|\mathbf{f})$:

$$\mathbf{e}^*_{\text{MAP}} = \arg\max_{\mathbf{e} \in T(\mathbf{f})} p(\mathbf{e}|\mathbf{f}) \tag{2.10}$$

In reality, however, the posterior distribution at hand is of the form $p(\mathbf{e},\mathbf{d}|\mathbf{f})$: a distribution over *derivations*. Therefore, MAP decoding requires marginalising over the derivations of $\mathbf{e}$.

$$\mathbf{e}^*_{\text{MAP}} = \arg\max_{\mathbf{e} \in T(\mathbf{f})} \sum_{\mathbf{d} \in D(\mathbf{e},\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f}) \tag{2.11}$$

Since both $|D(\mathbf{e},\mathbf{f})|$ and $|T(\mathbf{f})|$ may be exponential in the size of the input string, this maximisation turns out to be NP-complete as shown by Knight (1999). The decision rule in Equation 2.11 is also referred as *max translation* decoding. An often used approximation to *max translation* decoding is *max derivation* decoding:

$$
\begin{aligned}
\mathbf{e}^*_{\text{MAP}} &= \arg\max_{\mathbf{e}\in T(\mathbf{f})} p(\mathbf{e}|\mathbf{f}) \\
&= \arg\max_{\mathbf{e}\in T(\mathbf{f})} \sum_{\mathbf{d}\in D(\mathbf{e},\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f}) \\
&\approx \arg\max_{\mathbf{e}\in T(\mathbf{f})} \max_{\mathbf{d}\in D(\mathbf{e},\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f}) \qquad (2.12) \\
&= Y\left(\arg\max_{\mathbf{d}\in D(\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f})\right) \qquad (2.13)
\end{aligned}
$$

The expensive `sum` operation in (2.11) has been replaced by a simpler `max` operation in (2.12) which can be found using (2.13). The decision rule in (2.13) therefore corresponds to finding the mode of the posterior distribution over derivations. While in most cases computing the max derivation solution exactly remains intractable, by which we mean that the computation is too expensive to be practical, efficient approximate search algorithms do exist. Therefore, the max derivation decision rule is employed by most SMT systems (Koehn et al., 2003; Chiang, 2007).

The max derivation approximation and solution are often also referred to as the *Viterbi* approximation and solution respectively (Li et al., 2009b). However, it is worth pointing out that the term Viterbi solution implies an exact solution found using dynamic programming techniques whereas the max derivation solution cannot be computed exactly for all but the most trivial of translation models. In this thesis, so as to avert any misconception as to the exactness of the solution, we prefer the use of the term max derivation.

2.3.2 Minimum Bayes Risk Decoding

An alternate decision rule comes from statistical decision theory. This states that given the true probability distribution $p(\mathbf{e}|\mathbf{f})$, the optimal decision rule for any statistical model is the solution that *minimises its risk* or *expected loss*. This decision rule is often referred to as the minimum Bayes risk (MBR) rule (Kumar and Byrne, 2004) :

$$
\begin{aligned}
\mathbf{e}^*_{\text{MBR}} &= \arg\min_{\mathbf{e}\in T(\mathbf{f})} R_{\mathbf{f}}(\mathbf{e}) \\
&= \arg\min_{\mathbf{e}\in T(\mathbf{f})} \sum_{\mathbf{e}'\in T(\mathbf{f})} \ell(\mathbf{e},\mathbf{e}')p(\mathbf{e}'|\mathbf{f}) \qquad (2.14)
\end{aligned}
$$

where $R_{\mathbf{f}}(\mathbf{e})$ represents the risk when translating $\mathbf{f}$ of choosing $\mathbf{e}$ and $\ell(\mathbf{e},\mathbf{e}')$ is the loss incurred when choosing solution $\mathbf{e}$ if the true solution is $\mathbf{e}'$.

The MBR decision rule therefore chooses a high-probability translation that on average is most similar to any possible reference translation. This notion of similarity is captured using a loss function. When the loss function is an exact match criterion, also called as a 0/1 loss function, then the MBR decision rule is equivalent to the MAP rule.

However, as we saw in Section 2.2, SMT systems are typically evaluated using metrics (or loss functions) such as BLEU that reward partial matches. In such cases, it is preferable to use the MBR decision rule.

2.3.3 Consensus Decoding

The MBR decision rule is an example of a consensus decision rule - rather than simply returning the mode of the posterior distribution, it aims to find a solution which is most similar to other high-probability translations generated by the model. However, using the MBR decision rule can be expensive since its algorithmic complexity is $O(|T(\mathbf{f})|^2)$.

DeNero et al. (2009) propose a fast linear-time consensus decoding alternative to MBR. Their decision rule is of the form:

$$\mathbf{e}^*_{\text{Fast-CON}} = \arg\min_{\mathbf{e} \in T(\mathbf{f})} \ell(\mathbf{e}, \mathbb{E}_{P(\mathbf{e}'|\mathbf{f})}[\phi(\mathbf{e}')]) \tag{2.15}$$

where $\phi(e')$ maps the string e' to a feature-based representation. In contrast to MBR, the fast consensus decision rule moves the expectation term inside the loss function, thus reducing the complexity of the algorithm to linear while maintaining a consensus-like objective similar to MBR's.

2.4 Search Algorithms

Having established the different flavours of decision rules that exist, we now look at search algorithms which aim to find the solution of the chosen decision rule. The search algorithm to be used is not only dependent on the decision rule but also on the structure of the underlying probabilistic model. By structure, we broadly mean the *features* of the model. If the features of the model are local or near-local, then efficient *dynamic programming* based algorithms can be used. Otherwise, we need to resort to more computationally expensive algorithms.

In the rest of this section, we again assume that the model under consideration is in the form of a probabilistic log-linear model. We will illustrate the algorithms using a

phrase-based model but will highlight those cases where an alternate representation of the translation equivalence model (e.g a syntax-based model) necessitates a substantial change in the search algorithm.

2.4.1 Max Derivation Decoding

The search problem in max derivation decoding consists of finding the most probable derivation (Equation 2.13). While max derivation decoding is simpler than max translation decoding, its complexity is exponential in the length of the input so *exact inference* remains intractable.

Why is that so? Recall that in phrase-based decoding, the source sentence can be segmented in multiple ways. There are usually multiple ways of translating each segment. Also, source-side reordering is allowed. If unlimited reordering is allowed, the complexity of search for an input sentence of length I is $O(I^2 2^I)$ (Lopez, 2009), a complexity which is exponential. Most systems limit the extent to which reordering is allowed. This can reduce the search space drastically. For example, the Moses (Koehn et al., 2007) phrase-based decoder requires that first word of the current source phrase being considered for translation be within a window of Λ words from the last word of the most recently translated phrase. Also, the last word of the currently translated source phrase should be within Λ words of the leftmost untranslated source word. As a result, the complexity of the search space is reduced to $O(\Lambda^2 2^\Lambda I)$, which is exponential in Λ but linear in the size of the input (Lopez, 2009). Since Λ is usually set to a small value, the complexity of this restricted search algorithm is more tractable.

2.4.1.1 Beam Decoding

For phrase-based models, the most popular search algorithm for max derivation decoding is *stack decoding* with *beam search*, also referred to as simply *beam decoding* (Koehn, 2004a). Search proceeds as follows. First, the source sentence is segmented into phrases. All segmentations are equiprobable. For every source phrase, all possible *translation options* are collected from the phrase table. Starting with an initial empty *hypothesis*, the hypothesis is expanded by picking a source phrase to be translated and choosing a target phrase for that source. While the target is generated from left to right, source side reordering is allowed as long as reordering limits are respected. Each time a hypothesis is expanded, it keeps a backpointer to the hypothesis from which it expands and is assigned a score based on its partial model costs. Hypotheses are stored

in *stacks* based on the number of foreign words translated. Each hypothesis keeps track of the target phrase it generates. It is also annotated with a *signature* which consists of the source words that have been translated so far as well. Additionally, if one of the features of the model is a score from a language model of order n, then the last n-1 target words generated by the hypothesis are added to the signature too. A hypothesis is considered complete if all the source words have been covered. The best scoring complete hypothesis from the final stack is the max derivation solution.

The search procedure described above exhaustively expands all hypotheses, a process which is very inefficient. Decoding is speeded up in two ways. Firstly, we can use a dynamic programming algorithm: if two hypotheses have the same signature, then the two can be safely *recombined* and only the one with highest score retained for further expansion. This is a risk-free strategy because both hypotheses are certain to have identical expansions; therefore the highest scoring partial hypothesis is guaranteed to have a higher final score than its competitor.

The recombination strategy speeds up decoding but not to a large enough extent. Therefore, in addition, a *pruning*-based risk prone strategy is used. In *histogram pruning*, each stack only retains the n highest scoring partial hypotheses while in *threshold pruning* hypotheses with probabilities more than k times lower than the top scoring hypothesis in that stack are discarded. Pruning can lead to *search errors* i.e. the stack decoding algorithm can fail to find the true solution to the search problem. This can happen if at some point during search, the highest scoring complete solution has a lower partial score than other hypotheses in the same stack. To diminish the risks of search errors, the pruning criterion is altered to not only take into account the partial score of a hypothesis but also to consider an estimate of its *future cost* - the cost of translating the currently untranslated parts of the input. Calculating the future cost exactly is too inefficient (and is equivalent to running the search algorithm to completion) so a *heuristic* estimate is used instead. By including the future cost estimate, search errors are reduced dramatically.

Beam decoding is a very efficient algorithm for max derivation decoding. By carefully tuning the pruning parameters, a good balance between search accuracy and speed can be obtained. However, since the algorithm relies on dynamic programming for efficient search, the features of the model are restricted to be local or near-local precluding feature functions that look at long distance interactions on the target side.

2.4.1.2 Greedy Decoding

An alternative decoding algorithm is greedy decoding (Germann et al., 2001; Marcu and Wong, 2002; Langlais et al., 2007). First, a rough initial solution is generated. This solution is then iteratively improved using a *greedy hill-climbing* algorithm. The algorithm successively proposes small *local changes* to the solution, e.g it might propose to change the translation of a source word or swap the order of translation of two source words. If the new configuration has a higher probability than the existing one, then the new configuration is greedily accepted, otherwise the existing configuration is retained. The algorithm stops when none of the small local operations lead to a more probable translation.

Greedy decoding has several nice characteristics. Since the local operations are usually simple to compute, greedy decoding can quickly *converge* to a translation of high quality. Additionally, given that a full translation is available at all times, feature functions that operate over the whole translation (also known as global features) can be used. On the other hand, greedy decoding explores a smaller search space than beam decoding and may converge to a *local optima* rather than the global optimum. This may happen when a move to the global optimum requires first traversing through areas of low probability.

2.4.1.3 Optimal Decoding

Both greedy decoding and beam decoding run the risk of search errors. Algorithms that do not make any search errors are called *optimal decoding* algorithms. Germann et al. (2001) note the similarities between decoding and the Traveling Salesman problem and are able to recast decoding as a integer linear programming (ILP) problem which they then solve *exactly* using standard ILP solvers. For efficiency reasons, they limit their experiments to word-based models (IBM Model 4) on sentences of length up to 8 words using a bigram language model. Further work by Riedel and Clarke (2009) is able to scale Germann et al. (2001)'s approach to sentence lengths of up to 30 words by employing more sophisticated approaches to ILP solving.

Optimal decoding is also possible by using *A* search* instead of beam search in the stack decoding algorithm presented earlier. A* search allows risk-free pruning by the use of *admissible heuristics*, i.e. the estimate of the future cost of a partial hypothesis can never overestimate the true cost to completion. Note that the future cost estimate

used in beam search methods like (Koehn, 2004a) is not an admissible heuristic since it can under or over estimate the true completion cost.

Och et al. (2001) present an A* decoding algorithm for IBM Model 4. While A* search guarantees optimal decoding, it is usually much less efficient than beam search. A further reason why A* search is not prevalent is because coming up with admissible heuristics for decoding is hard.

2.4.1.4 Search Hypergraphs and N-best Lists

During decoding we are interested in finding the 1-best solution to the decision rule. Recall that stack decoding relies on hypothesis recombination for efficient search. If all we are interested in is finding the overall best solution, whenever two or more hypotheses are recombined, we can safely expand just the winning one, discarding traces of the "losing ones". However, we can also choose to keep track of the recombinations. If we do so, we end up with a data structure called a search or translation *lattice* which compactly represents the entire space explored during decoding. The search lattice is in the form of a weighted finite state machine (WFSM) consisting of states and transitions between them. Figure 2.2 shows an example of a translation lattice, where edge is annotated with its score and the target phrase is produces.

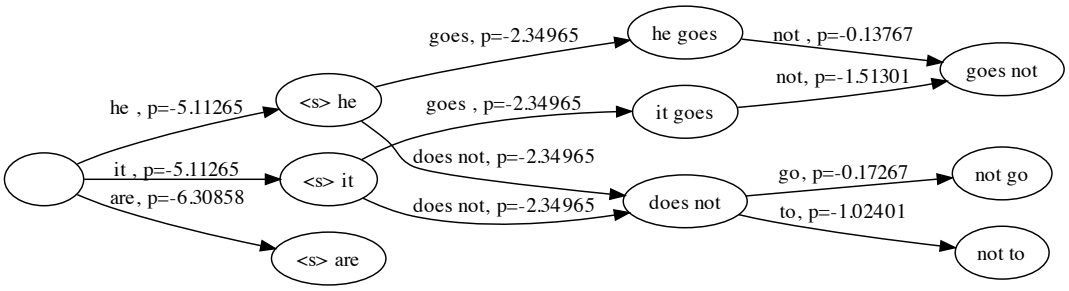


Figure 2.2: An example translation lattice. A trigram language model was used during search.

A similar compact representation of the search space arises as a by-product of grammar-based decoding. This representation is often called a translation *forest*, which formally is a weighted acyclic *hypergraph*. A hypergraph is a generalisation of a graph where an edge can connect any number of vertices. Formally, a hypergraph is a pair

$\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V}$ is a vertex set and $\mathcal{E}$ is a set of hyperedges, $\mathcal{E} \in \mathcal{V}^* \times \mathcal{V}$. Each hyperedge, $e \in \mathcal{E}$ connects a head vertex $h(e)$ with a sequence of tail vertices $T(e) = \{v_1, \cdots, v_n\}$. The *arity* of a hyperedge is the number of its tail vertices while the arity of a hypergraph is the maximum arity of its hyperedges. A hyperedge of arity 1 is a regular edge and a hypergraph of arity 1 is a regular graph or a lattice.

Since any finite-state automaton can also be encoded as a hypergraph, going forward we will refer to algorithms defined over lattices and forests as *hypergraph-based* algorithms. Hypergraphs encode the *exponential* number of derivations in the search space in *polynomial* space. Therefore, hypergraph-based algorithms can run in polynomial time (or equivalently in linear-time in the size of the hyperedges in the hypergraph).

A common application of MT hypergraphs is the extraction of n-best lists which are ranked lists of the n most probable derivations in the hypergraph. N-best extraction from a finite state machine or a hypergraph is a well studied problem for which efficient algorithms exist. N-best lists have many practical uses:

- They can be used during the *parameter estimation* step of the SMT pipeline.
- They can be used in a reranking step where features that are too expensive to compute during search can be applied (Shen et al., 2004).
- They can be used for applying decision rules that do not factorise over the search space (Kumar and Byrne, 2004).
- They can be used for model debugging purposes.

The last few years has seen an explosion of research interest in using hypergraphs instead of n-best lists in algorithms where n-best lists have hitherto proven to be useful. This is because by using the search hypergraph, it is possible get a more accurate view of the search space than with an n-best list.

2.4.2 Max Translation Decoding

All recently proposed SMT models exhibit *derivational ambiguity* i.e. there are multiple ways of yielding the same output string given an input string. In phrase-based models the ambiguity arises from the fact that different source-side segmentations can lead to the same translation string while in syntax-based models, different derivation

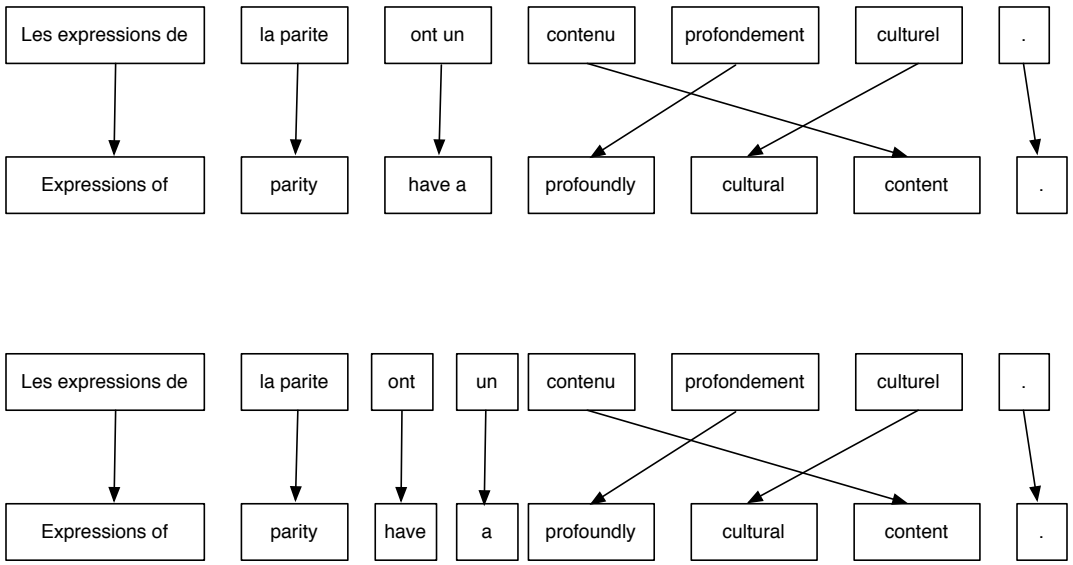


Figure 2.3: Derivational ambiguity: both derivations produce the same target string but using different segmentations of the source.

trees can produce the same string. An example of derivational ambiguity in phrase-based models is shown in Figure 2.3.

How bad is the derivational ambiguity problem? When decoding the Chinese-English NIST MT04 and MT05 test sets using Hiero, an implementation of a syntax-based model (Chiang, 2007), Li et al. (2009b) find that there are on average 115 derivations for each translation string in their model. On the other hand, for a similar hierarchical phrase-based system, Blunsom et al. (2008) find that the number of derivations yielding the same string is **exponential** in the length of the input.

We now review three algorithms proposed for approximate max translation decoding.

2.4.2.1 N-best Crunching

One simple algorithm for max translation decoding is n-best crunching (May and Knight, 2006) which works as follows. First the input sentence is decoded using a beam decoder. Then an n-best list of derivations is extracted with n set to a sufficiently large number (1,000 or 10,000). The n-best list is then *crunched*, i.e. the scores of derivations yielding the same translation string are summed together and the translation string with the greatest score is output as the max translation solution.

While simple to implement, n-best crunching is a very rough approximation to max translation decoding. This is because n-best lists only capture a small subset of

the space of all derivations in the search graph. The latter is exponential in the length of the input whereas typically, due to efficiency considerations, n-best list extraction is limited to around $n = 100,000$.

2.4.2.2 Beam Search

Blunsom et al. (2008) present a beam search approach to max translation decoding. They apply their method for decoding in a Hiero-like system but the approach can just as well be applied to phrase-based models. Their method modifies the dynamic program used for hypothesis recombination by using the entire partial target string generated rather than just the context needed for language model scoring. As a result, two hypotheses that recombine have to have generated the same output string. At each recombination step, instead of storing the score of the most probable hypothesis as is done during max derivation decoding, the scores are summed together. At the end of decoding, each hypothesis in the final stack contains the sum of the scores of all derivations yielding the same translation string. As a result of the more involved dynamic program, fewer recombinations take place. This leads to an explosion in the number of hypotheses to be expanded, both slowing down decoding and increasing the memory footprint dramatically. Therefore, Blunsom et al. are forced to resort to very aggressive pruning and restrict their experiments to sentences less than 10 word long.

2.4.2.3 Variational decoding

Li et al. (2009b) propose a novel algorithm for max translation decoding called *variational decoding* which, unlike the approach of Blunsom et al. (2008), is able to scale to large tasks. Variational decoding is an instantiation of a general class of approximate inference algorithms known as *variational inference* in which the original intractable distribution of interest p is approximated by a simpler distribution q which supports exact inference. The key therefore in variational inference is to come up with an approximation which is similar enough to the original distribution. This notion of similarity between the distributions is measured using the information theoretic measure of Kullback-Leibler divergence, KL $(p||q)$.

The variational decoding decision rule used by Li et al. (2009b) is :

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \left(\sum_n \theta_n \cdot \log q_n(\mathbf{e}) \right) + \theta_v \cdot \log \tilde{p}(\mathbf{e}|\mathbf{f}) \tag{2.16}$$

where:

- $q_n(\mathbf{e})$ is an n-gram based variational approximation of the true distribution $p(\mathbf{e}|\mathbf{f})$,
- θ_n weights the variational approximation $q_n(\mathbf{e})$ and
- θ_v is a parameter which weights $\log \tilde{p}(\mathbf{e}|\mathbf{f})$, the max derivation score of string $\mathbf{e}$

Each $q_n(\mathbf{e})$ is a distribution over the n-grams in the search hypergraph obtained when translating the source string $\mathbf{f}$ as target string $\mathbf{e}$. As n gets larger, the approximation q approaches p but on the other hand decoding with q becomes more complex. Li et al. explain that by setting $n \leq m$ where m is the order of the language model used for decoding, efficient variational inference can be performed.

Each $q_n(\mathbf{e})$ is of the form:

$$q_n(\mathbf{e}) = \prod_{w \in W_n} q(r(w)|h(w))^{c_w(\mathbf{e})} \tag{2.17}$$

where:

- W_n is the set of n-grams of order n in the translation hypergraph of $\mathbf{f}$,
- w is an n-gram occurring $c_w(\mathbf{e})$ times in $\mathbf{e}$,
- $h(w)$ are the first $n-1$ words in w and
- $r(w)$ is the last word in w.

The parameters of the model are the conditional probability distributions, $q(r(w)|h(w))$, of the n-grams w in the translation hypergraph. These parameters are estimated by computing the ratio of the expected count of every n-gram w and its history $h(w)$ in the translation hypergraph under the true distribution p:

$$q(r(w)|h(w)) = \frac{\sum_{\mathbf{d} \in D(\mathbf{f})} c_w(\mathbf{e}) p(\mathbf{e}, \mathbf{d}|\mathbf{f})}{\sum_{\mathbf{d} \in D(\mathbf{f})} c_{h(w)}(\mathbf{e}) p(\mathbf{e}, \mathbf{d}|\mathbf{f})} \tag{2.18}$$

where $\mathbf{e} = Y(\mathbf{d})$.

Computing the terms in Equation 2.18 requires a sum over an exponential number of derivations. However, the hypergraph represents the exponential space of derivations in polynomial space by collapsing together common subderivations. Li et al. take advantage of this fact to present an algorithm for computing Equation 2.18 in time linear in size to the number of hyperedges in the hypergraph.

Decoding with the decision rule in Equation 2.16 for $n \leq m$ consists of first decoding the input $\mathbf{f}$ with a max derivation decoder using a language model of order m then *rescoring* the edges of the resulting lattice or hypergraph with weights as per the decision rule. The yield of the rescored best scoring derivation is the max translation solution.

In experiments on a large scale Chinese to English translation task, Li et al. report significant improvements using variational decoding over max-derivation, MBR and n-best crunching (n=10000) decoding.

2.4.3 MBR Decoding

The MBR decision rule given in Equation 2.14 requires computing the expected loss of each translation string produced by the translation model with respect to every other translation string, a computation which is intractable to perform exactly due to the exponential number of translations.

There have been two main algorithms proposed for MBR decoding in SMT systems, both of which consist in rescoring an initial list of translations produced by a first-pass max derivation decoder. Since BLEU is the evaluation metric of choice in SMT, the algorithms use (1 - BLEU) as loss function or equivalently BLEU as gain function in which case the arg min term is replaced by an arg max.

2.4.3.1 *N*-best MBR

We begin by considering the first MBR decoding variant, *n*-best MBR decoding (Kumar and Byrne, 2004). As we saw in Equation 2.14, computing the MBR solution is an algorithm quadratic in the size of $T(\mathbf{f})$ with the inner loop of the algorithm performing the risk computation for each of the translations in the outer loop. The space of translation candidates over which the risk is computed is usually referred to as the *evidence space* of the algorithm and denoted by ε_E. Similarly, the translation minimising the risk is chosen from a space of candidates, ε_H denoted as the *hypothesis space* (Tromble et al., 2008).

In *n*-best MBR, both the hypothesis and evidence spaces of output strings are restricted to an *n*-best list of translations extracted from the lattice of the first-pass decoder. To ensure diversity of the translation strings, an *n*-best of *distinct* strings is usually extracted.

BLEU is a corpus level metric whereas MBR decoding is performed at a sentence level. As discussed in Section 2.2.1, BLEU is an inappropriate metric to use at the sentence level so instead the smoothed-BLEU implementation proposed in Lin and Och (2004) is used. We denote this variant of BLEU as SBLEU.

Note that the decision rule in (2.14) requires the conditional probability $p(\mathbf{e}|\mathbf{f})$ but most decoders return an unnormalised score $s(\mathbf{e},\mathbf{d},\mathbf{f})$ defined over derivations of the model. Converting this score to a probability requires exponentiating the score followed by two subsequent steps:

- marginalising over all the set of derivations $D(\mathbf{e},\mathbf{f})$ for each output string $\mathbf{e}$ and
- normalising by $Z(\mathbf{f})$, the partition function of the underlying log-linear model.

Both steps involve an expensive summation over an exponential number of derivations so in practice they are approximated; the score of each output string $\mathbf{e}$ is approximated by the score its most probable derivation (i.e we perform a max derivation approximation) whereas Z is computed over the derivations in the n-best list.

The resulting n-best MBR decision rule is given by:

$$\begin{aligned}\mathbf{e}^*_{\text{MBR}} &= \arg\min_{\mathbf{e}\in T(\mathbf{f})} \sum_{\mathbf{e}'\in T(\mathbf{f})} \ell(\mathbf{e},\mathbf{e}')p(\mathbf{e}'|\mathbf{f}) \\ &\approx \arg\max_{\mathbf{e}\in Y(\mathrm{N}(\mathbf{f}))} \sum_{\mathbf{d}'\in \mathrm{N}(\mathbf{f})} \text{SBLEU}(\mathbf{e},\mathbf{e}') \frac{\exp\left[s(\mathbf{e}',\mathbf{d}',\mathbf{f})\right]}{\sum_{\hat{\mathbf{d}}\in \mathrm{N}(\mathbf{f})} \exp\left[s(\hat{\mathbf{e}},\hat{\mathbf{d}},\mathbf{f})\right]}\end{aligned}$$

where $\mathrm{N}(\mathbf{f})$ represents the set of the n highest scoring derivations in $D(\mathbf{f})$, $\mathbf{e}' = Y(\mathbf{d}')$ and $\hat{\mathbf{e}} = Y(\hat{\mathbf{d}})$.

Typically, a scaling factor γ is introduced to control the shape of the estimated distribution. The value of γ is optimised for good MBR decoding performance using a grid-search on a held-out set.

The decision rule with the scaling factor is given by:

$$e^*_{\text{N-MBR}} = \arg\max_{\mathbf{e}\in Y(\mathrm{N}(\mathbf{f}))} \sum_{\mathbf{d}'\in \mathrm{N}(\mathbf{f})} \text{SBLEU}(\mathbf{e},\mathbf{e}') \frac{\exp\left[\gamma\cdot s(\mathbf{e}',\mathbf{d}',\mathbf{f})\right]}{\sum_{\hat{\mathbf{d}}\in \mathrm{N}(\mathbf{f})} \exp\left[\gamma\cdot s(\hat{\mathbf{e}},\hat{\mathbf{d}},\mathbf{f})\right]} \tag{2.19}$$

N-best MBR decoding has been shown to often give small improvements over max-derivation decoding and is supported by a large range of open source SMT decoders (Koehn et al., 2007; Li et al., 2009a).

2.4.3.2 Lattice and Hypergraph MBR

Tromble et al. (2008) introduce lattice MBR, an alternative to *n*-best MBR, which involves an exponential number of derivations in the MBR decision rule. In lattice MBR, both the evidence space and the hypothesis space can be defined over a translation lattice. Kumar et al. (2009) extend that work by presenting an MBR algorithm for the more general case involving a *hypergraph*.

The hypergraph MBR formulation uses a dynamic programming which requires that the gain function can be decomposed as a sum of of local gain functions over the hyperedges of the hypergraph. Such a gain function is introduced by Tromble et al. where they *approximate* $\log(\text{BLEU})$ as a linear function of *n*-gram matches and the length of the candidate translation.

Given reference and the candidate translations $\mathbf{e}$ and $\mathbf{e}'$ respectively, the linear approximation G is as follows:

$$G(\mathbf{e},\mathbf{e}') = \theta_0|\mathbf{e}'| + \sum_w \theta_{|w|}\#_w(\mathbf{e}')\delta_w(\mathbf{e}) \tag{2.20}$$

where w is an *n*-gram in either $\mathbf{e}$ or $\mathbf{e}'$, $\#_w(\mathbf{e})$ is the number of times w appears in $\mathbf{e}$, $\delta_w(\mathbf{e})$ is 1 if w in $\mathbf{e}$ and 0 otherwise, θ_0 is a weight associated with the candidate length and $\theta_{1\ldots N}$ are weights associated with *n*-grams of orders up to N.

With a linear function of this form, the MBR decision rule can be reformulated thus:

$$\mathbf{e}^*_{\text{MBR}} = \arg\max_{\mathbf{e}'\in\varepsilon} \left(\theta_0|\mathbf{e}'| + \sum_w \theta_{|w|}\#_w(\mathbf{e}')p(w|\varepsilon) \right) \tag{2.21}$$

where $p(w|\varepsilon)$, the posterior probability of the *n*-gram w in the lattice ε, is given by:

$$p(w|\varepsilon) = \sum_{\mathbf{e}\in\varepsilon} \delta_w(\mathbf{e}) \sum_{\mathbf{d}\in D(\mathbf{e},\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f}) \tag{2.22}$$

In contrast to the inner loop of the standard MBR decision rule which requires a summation over an exponential number of translations in the evidence space, the summation in the inner loop of Equation 2.21 is much more efficient since it is over the set of the *n*-grams that occur in the hypergraph, a set which is only linear in size to the number of hyperedges in the hypergraph.

Efficient hypergraph MBR is made possible by the use of the linear approximation to corpus BLEU. The parameters of this approximation are the θ terms whose values in Tromble et al. (2008) are given by the following two equations:

$$\theta_0 = -1 \tag{2.23}$$

$$\theta_n = \frac{1}{4p \cdot r^{n-1}} \tag{2.24}$$

where p is the BLEU unigram precision and r is the decay in BLEU n-gram precision for higher order n-grams, averaged over multiple decoding runs of the baseline decoder.

Both Tromble et al. (2008) and Kumar et al. (2009) find that hypergraph MBR significantly outperforms n-best MBR. Their experiments show that the improvement in performance comes from a more accurate estimation of risk brought about by using a much larger evidence space. Note though that the hypergraph does not encode the *entire space* of possible translations since substantial pruning is required during decoding. As far as the hypothesis space is concerned, using a 1000-best list is as good as using the entire hypergraph. This means that the MBR solution is almost always a highly probable solution as per the base decision rule.

Since the hypergraphs can be very large, efficient hypergraph MBR decoding requires additional pruning. This is done using Forward-Backward pruning (Sixtus and Ortmanns, 1999), an algorithm to limit the average number of hyperedges per word (the hypergraph density) to a configurable parameter optimised based on MBR performance on a held-out set.

2.4.4 Consensus Decoding

While hypergraph MBR applies the MBR decision rule using a similarity metric which is an approximation to BLEU, the fast consensus decoding of DeNero et al. (2009) applies an alternate hypergraph-based decision rule using a similarity metric which is BLEU itself.

The consensus decoding objective, with BLEU as similarity metric, is given by:

$$\mathbf{e}^*_{\text{Fast-CON}} = \arg\max_{\mathbf{e} \in Y(\mathrm{N}(\mathbf{f}))} \text{BLEU}(\mathbf{e}, \mathbb{E}_{P(\mathbf{e}'|\mathbf{f})}[\phi(\mathbf{e}')]) \tag{2.25}$$

where $\phi(\mathbf{e}')$ maps the string $\mathbf{e}'$ to a feature-based representation. Note that in this instance, the definition of the BLEU metric is overloaded since the second argument which is usually a string (the reference sentence) is in this case a feature vector.

Consensus decoding is a fast linear time alternative to MBR. It involves a first pass in which the expectations of the features of the similarity measure under the model

distribution, $\mathbb{E}_{P(\mathbf{e}'|\mathbf{f})}[\phi(\mathbf{e}')]$ are computed, followed by the application of the similarity measure to every candidate translation $\mathbf{e}$ extracted from an n-best list of translations.

The features in BLEU-based consensus decoding are n-gram counts for n from 1 up to 4. DeNero et al. present an algorithm to compute the expectation of these features over a hypergraph in time linear to the number of hyperedges. The feature expectation algorithm is very similar to that presented in both Kumar et al. (2009) and Li et al. (2009b) underlying the similarities between all three approaches to decoding. In hypergraph MBR, expectations are computed for n-gram indicator functions $\delta_w(\mathbf{e})$ whereas in both consensus decoding and variational decoding expectations for n-gram counts $\#_w(\mathbf{e})$ are computed over the hypergraph.

In experiments, DeNero et al. find that consensus decoding is on average 80 times faster than MBR while having the same BLEU performance, when applying both algorithms on evidence spaces consisting of 1,000-best lists. They also find that forest-based consensus decoding always outperforms 10,000-best consensus decoding, a result in line with the findings of Tromble et al. (2008) when comparing lattice to n-best MBR.

2.4.5 Summary

In this section, we looked at search algorithms focusing on phrase-based models. These search algorithms attempt to find the solution of the chosen decision rule. Finding the max derivation solution in phrase-based models is intractable since an exponential number of derivations need to be considered. The most popular search algorithm for this task is stack decoding with beam search. Here, the search space is carefully organised so as to be able to use dynamic programming algorithms. Dynamic programming allows the sharing of common partial derivations among many derivations, rendering search more effective. By pruning partial derivations expected to be low scoring were they expanded to completion, the search space can be further reduced. However pruning is a risky strategy as the max derivation solution might be discarded were it to have a low partial score. This is referred to as a search error. By carefully designing the pruning mechanism, the risk of search errors can be minimised and a substantial improvement in translation speed can be gained.

We also looked at phrase-based decoding algorithms that guarantee to be free of search errors. However, they are usually too slow for decoding long sentences. Another alternative is greedy decoding which is suitable when dynamic programming methods

cannot be used. This is the case when the features of the model are global, that is they take into account characteristics of the entire source and target sentences. However, in most models of sufficient complexity, greedy decoding is likely to make more search errors than other decoding algorithms.

We next considered search algorithms for decision rules which take into account the whole distribution. Examples of these decision rules are max translation decoding, MBR decoding and fast consensus decoding. Blunsom et al. (2008) proposed a beam search approximation for performing max translation decoding. However, their method requires aggressive pruning and can only be applied to short sentences. Most approaches to these problems adopt a 2-pass strategy. First, approximate dynamic programming based max derivation search is performed. A by-product of this decoding algorithm is a translation hypergraph which is compact representation of the pruned search space.

The space of the true distribution can then be approximated using either an n-best list of translations extracted from the hypergraph or by using the entire hypergraph. The latter method requires designing clever dynamic programming algorithms that can exploit the characteristics of the data structure. Recent results have shown that search algorithms which use a hypergraph for estimating the true distribution outperform methods using n-best lists for the same purpose.

2.5 Parameter Estimation for Log-linear Models

Following Och and Ney (2002), most SMT models adopt a log-linear formulation of the translation task. The log-linear model is attractive since it allows tremendous flexibility for integrating knowledge sources in the model. These knowledge sources are usually referred to as *features*. However, the choice of features needs to be balanced by the need for having efficient decoding algorithms. For example, the most popular decoding algorithm for phrase-based models is beam decoding which relies on local or near-local features for efficient search. If the model contains features that take into account long distance interactions in the source and/or target, also known as global features, then alternate decoding algorithms are required.

Parameter estimation consists of finding appropriate weights for the features in the model. It is an optimisation problem in that it tries to find weights that maximise a given *objective function*. The parameter estimation phase is often referred as the log-linear model training step or simply the tuning step in the SMT pipeline. Numerous

objective functions have been proposed for SMT and in this section, we will go over some of the more salient ones. The training methods we describe are all examples of *discriminative training* algorithms as they are designed to discriminate the right translation against the incorrect translations. An alternative to discriminative training is generative training which learns a model of the joint probability of the source and target sentences in the training set and then uses Bayes rule to make predictions at test time. Since discriminative training algorithms typically outperform generative training ones as far as test time performance is concerned, they are the preferred techniques for parameter estimation in SMT.

As a rule of thumb, in order to get good performance at test time, the objective function during parameter estimation should match the test time decision rule. For example, Blunsom et al. (2008) find when performing max translation decoding that they get better translation performance if their model had been tuned using an objective function that accounts for derivational ambiguities rather than one which only considers the most likely derivation.

The most popular tuning algorithm for SMT models is minimum error rate training (MERT; Och (2003)) which can be used to directly optimise translation quality as evaluated by a given metric such as WER and BLEU. MERT is especially popular because it is very efficient at optimising corpus BLEU, the metric by which most SMT systems are evaluated. The main drawback of MERT however is that its optimisation algorithm can only be used to tune a dozen or so features. A particularly active research area currently is devising tuning algorithms that can scale to millions of features.

2.5.1 Maximum Likelihood Training

Och and Ney (2002) train their log-linear model by finding weights which optimise the likelihood of a given training dataset.

Sentence-level Training

Maximum likelihood (ML) estimation aims to maximise the joint likelihood of the model parameters and the training data. In the case of log-linear models, the ML objective function corresponds to the standard objective function for *maximum entropy* models given by:

$$\lambda_{ml} = argmax_{\lambda} \left\{ \sum_{c=1...C} \log p(\mathbf{e}_c|\mathbf{f}_c;\lambda) \right\} \tag{2.26}$$

where the set $\{\mathbf{e}_c, \mathbf{f}_c\}_1^C$ denotes a training corpus of C sentence pairs. Equation 2.26 aims to find the optimal weight vector λ_{ml} which maximises the conditional log-likelihood of the training data. The objective function in (2.26) has a very desirable property: it is *convex* with a single optimum. This optimum can be found using standard numerical optimisation packages. However, the training algorithm is not so straightforward in the case of SMT.

As we have seen earlier, computing the conditional probability requires two terms: a numerator which sums over $D(\mathbf{e}_c, \mathbf{f}_c)$ and a denominator which sums over $D(\mathbf{f}_c)$. Both summations are expensive so instead the denominator is computed over an n-best list and a max derivation approximation is used for the numerator. Additionally, the n-best list might not contain any derivations which yield the reference, necessitating the use of a *surrogate reference* derivation.

As surrogate, Och (2003) chooses the derivation in the n-best list that minimises word error rate with respect to any of the reference translations whereas Zens et al. (2007) choose the derivation that maximises a sentence-level approximation to BLEU.

Och notes that since the ML training objective function does not directly take into account the evaluation metric used to measure test time performance, there is no guarantee that the values learnt for the model weights are optimal with respect to the metric. Another criticism of the ML objective is that is too harsh a training criterion since it does not distinguish between translations that are close to the reference and ones that are far from the reference, penalising all of them equally.

Nevertheless, Och and Ney find that parameters obtained from ML training improve performance dramatically compared to a baseline where the parameters are set arbitrarily. This finding motivates the need for a parameter estimation phase in the SMT pipeline.

N-gram-level Training

An alternative ML training criterion which addresses some of the limitations of sentence-level ML training is presented by Zens et al. (2007). Their objective function is defined over the posterior probabilities of the n-grams appearing in the target side

of the training set (they make a conditional independence assumption between the n-grams) where the normalisation term for the probability distribution is still computed over an n-best list.

This training criterion has two main advantages over the sentence-level variant. Firstly, it does not require the use of surrogate references since the posterior probabilities of the n-grams in the true reference can be easily computed (smoothing is required to account for n-grams that might not appear in the n-best list). Secondly, it is less harsh than the sentence-level ML training as by optimising n-gram posteriors, it rewards partial matches. Zens et al. find that the n-gram level MLE outperforms the sentence level variant when performing both max derivation and MBR decoding, attributing this improvement in performance to the two reasons mentioned.

Another advantage of this objective function, one which Zens et al. omit to mention, is that by summing up over reference translations n-grams in the n-best list, it is able to account for some derivational ambiguity.

2.5.2 Maximum A Posteriori Training

(Blunsom et al., 2008; Blunsom and Osborne, 2008) argue that an unregularised training criterion such as the one used in Equation 2.26 leads to parameters that overfit the training data. To address overfitting, they add a Gaussian prior term to the objective function. The prior regularises the model by penalising the objective function when the model parameters deviate too far away from the mean of the Gaussian. This modified training regime is referred to as Maximum A Posteriori (MAP) training (Gauvain and Lee, 1994).

MAP training as implemented by (Blunsom et al., 2008; Blunsom and Osborne, 2008) attempts to address some of the other limitations of sentence-level ML training too. Most notably, Blunsom et al. propose a method that accounts for derivational ambiguities by marginalising over them. The underlying model used in (Blunsom et al., 2008; Blunsom and Osborne, 2008) is a syntax-based model; however, no language model is used. Due to this lack of a language model, for each training instance, they are able to produce a full unpruned translation forest from which they can exactly compute the partition function of the log-linear model. They do so using the *inside-outside* algorithm, a dynamic programming algorithm for tree-based models. They run the decoder a second time but this time in *constrained* mode, i.e. the target

side is fixed to be the reference translation. The numerator is computed by running the inside outside algorithm on the resultant forest.

Blunsom et al. (2008)'s parameter estimation step requires that the reference translation be in the hypothesis space of (or be reachable by) the decoder. Often this is not the case. In phrase-based models, discounting the case where the reference might have been pruned away during search, there are two additional circumstances in which this situation can occur:

- A phrase-pair needed to reach the reference is not in the phrase-table. This is possible because of heuristics used to extract the phrase table.

- In order to make decoding tractable, phrase-based decoders enforce distortion limits i.e the start position of the source phrase to be decoded can be at a distance of most d words from the end position of the previously decoded source phrase. Typically, d is set to between 4 and 8. If, in order to reach the reference, a reordering of more than d words is required, the decoder is unable to generate such a derivation.

In order to have a training corpus of reachable sentences, Blunsom et al. find that they have to throw away 24% of their overall training set. In contrast to most SMT models where the parameter estimation step consists of finding weights for a handful of non-sparse features such as language model score or translation model score, Blunsom et al. learn weights for a more than a million sparse binary translation model features. Due to the computational cost of creating unpruned forests during training, the training set is restricted to only short sentences. Experimental results show that the best test time performance is obtained when both training and testing account for derivational ambiguity thus demonstrating the benefits of marginalising the latent variables of the model.

Blunsom and Osborne (2008) extend their previous work by introducing a language model in their model. Adding the language model renders the exact computation of Z intractable so they experiment with two approximations. In the first instance, the input is decoded with the help of beam pruning and the resulting forest is used to compute Z. Blunsom and Osborne postulate that such a forest, where low probability derivations have been pruned away, may provide a biased estimate of the true distribution. They argue that the presence of low probability derivations during training increases the discriminating power of the model. Therefore, they propose a new Monte Carlo

sampling based algorithm which takes as input the full distribution obtained when decoding without a language model and then draws samples from it. The resulting forest contains derivations which are mostly high scoring but also some low scoring ones with respect to the language model enriched distribution. Experiments show that at test time, performance using the latter approximation outperforms the former suggesting that having a broad view of the probability space is advantageous.

2.5.3 Minimum Error Rate Training

By far the most popular parameter estimation technique for log-linear models in SMT is Minimum Error Rate training (MERT) (Och, 2003). The main characteristic of this algorithm is that rather than maximising the conditional log-likelihood of the data, it directly optimises the error rate of the model on a held-out set as measured by an evaluation metric. MERT performs best if the metric used during tuning matches the one used at test time.

Formally, the MERT objective function seeks to minimise the error rate as follows:

$$\begin{aligned}
\lambda_{\text{MERT}} &= \arg\min_{\lambda} \sum_{c=1\ldots C} \text{Loss}\left(\arg\max_{\mathbf{e}\in T(\mathbf{f}_c)} p(\mathbf{e}|\mathbf{f}_c;\lambda), \mathbf{e}_c\right) && (2.27)\\
&\approx \arg\min_{\lambda} \sum_{c=1\ldots C} \text{Loss}\left(\arg\max_{\mathbf{d}\in D(\mathbf{f}_c)} p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda), \mathbf{e}_c\right) && (2.28)\\
&= \arg\min_{\lambda} \sum_{c=1\ldots C} \text{Loss}\left(\arg\max_{\mathbf{d}\in D(\mathbf{f}_c)} s(\mathbf{e},\mathbf{d},\mathbf{f}_c;\lambda), \mathbf{e}_c\right) && (2.29)
\end{aligned}$$

where $\text{Loss}(\mathbf{e},\mathbf{e}_c)$ quantifies the error in hypothesising translation $\mathbf{e}$ when the reference translation is $\mathbf{e}_c$. If a gain function such as BLEU is used instead of a Loss function, then the arg min is replaced by an arg max and the objective function is often referred to as the max BLEU objective.

Note that the objective function in Equation 2.27 supposes a loss function defined at the sentence level. The loss function can be defined at the corpus level too. A case in point is BLEU. For such a loss function, the MERT objective function can be modified to apply on statistics collected at the corpus level.

Equation 2.27 involves a search for the MAP solution which as we have seen in Section 2.3.1 is intractable. It is therefore approximated using the max derivation approximation (2.28) for which efficient solutions exist contingent on the locality of the features in the model. Since the objective function is only interested in the arg max solution and normalising model scores so to obtain a probability distribution

is an expensive operation, a further approximation can be performed by dropping the normalisation term. The objective function as given by (2.29) therefore optimises an *unnormalised linear model* rather than a *normalised log-linear* one. As a result of the training objective of (2.29), MERT finds weights that strongly favour a few derivations in the merged n-best list, where these derivations are the ones that lead to a reduced error rate. Consequently, MERT trained weights are best suited for the max derivation decoding decision rule.

The optimisation problem in (2.29) is hard since the objective function contains an arg max operation that precludes the use of gradient-based methods. It is also non-convex with many local optima. Och's proposed training algorithm works as follows:

- Regard $\text{Loss}(\arg\max_{\mathbf{d}\in D(\mathbf{f}_c)} s(\mathbf{e},\mathbf{d},\mathbf{f};\lambda))$ as a function of the parameter vector λ being optimised. Use the initial weight setting λ^0 to create an n-best list from which to select $\arg\max_{\mathbf{d}\in D(\mathbf{f}_c)} s(\mathbf{e},\mathbf{d},\mathbf{f};\lambda)$

- The error surface defined by Loss (as a function of λ) is piecewise linear with respect to a single parameter λ_m, hence one can determine precisely where it would be useful (values that change the result of the *argmax)* to evaluate λ_m for a given sentence using a simple line intersection method.

- Combine the list of useful evaluation points for λ_m and evaluate the corpus level $\text{Loss} = \sum_{c=1\ldots C} \text{Loss}(\arg\max_{\mathbf{d}\in D(\mathbf{f}_c)} s(\mathbf{e},\mathbf{d},\mathbf{f}_c;\lambda),\mathbf{e}_c)$ at each one.

- Select the model parameter that represents the lowest corpus level Loss as m varies, set λ_m and consider the parameter λ_j for the subsequent dimension j.

This training algorithm, referred to as minimum error rate training (MERT) is a greedy search in each dimension of λ, made efficient by realising that within each dimension, we can compute the points at which changes in λ actually have an impact on Loss. It is an iterative algorithm where the weights obtained at the end of the optimisation procedure are used to generate a new n-best list. This n-best list is merged with the previous n-best lists and the optimisation algorithm is run over the merged n-best lists until the n-best list does not change. To avoid being stuck in a low optimum due to an unfortunate initialisation, the algorithm is run with multiple random starting points.

Since the error is computed at the corpus level, MERT is particular suited to be used in conjunction with BLEU. The results obtained using MERT with BLEU as gain

function significantly outperform the results obtained using the standard maximum entropy training criteria. As a result, MERT is the standard training algorithm for SMT systems.

In contrast to Och's MERT in which the error surface for each feature component is computed over an n-best list of translation candidates, lattice MERT introduced by Macherey et al. (2008) leverages the exponential number of translations represented in a translation lattice for the same purpose, yielding faster convergence.

MERT is the most widely used parameter estimation technique for SMT systems since it is very good at optimising for BLEU and the outer loop which involves decoding the held-out set with the current weight sets is trivial to parallelise. It also has a few drawbacks. The main criticism of MERT is that due to its one dimension at a time line-search nature, it cannot be used to train models with more than a few (around 15) feature components. The algorithm either does not converge or converges very early with not very satisfactory weight settings. Additionally, due to the use of multiple random initialisation points, MERT in general is quite unstable, giving varying scores across different training runs.

A further drawback of MERT is that since it optimises an unnormalised linear model, the probabilistic interpretation of the model is compromised. A probability distribution can be obtained by normalising the model scores by the sum of scores of derivations in an n-best list or a lattice, but this distribution is arbitrarily shaped. Typically, downstream tasks such as MBR decoding scale the distribution until it has a shape appropriate for the task in question.

2.5.4 Minimum Risk Training

An alternate training objective function is minimum risk training also known as expected BLEU training when BLEU is the evaluation metric being used. Minimum risk training was first proposed for SMT by Smith and Eisner (2006) and has subsequently been used by Zens et al. (2007) and Li et al. (2009b). Formally, the objective function is:

$$\begin{aligned}\lambda_{mr} &= \arg\min \frac{1}{C} \sum_{c=1\cdots C} \sum_{\mathbf{d}\in D(\mathbf{f}_c)} p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)\ell(\mathbf{e},\mathbf{e}_c) \\ &= \arg\max \frac{1}{C} \sum_{c=1\cdots C} \sum_{\mathbf{d}\in D(\mathbf{f}_c)} p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)\mathrm{BLEU}(\mathbf{e},\mathbf{e}_c) \qquad (2.30)\end{aligned}$$

where the set $\{\mathbf{e}_c, \mathbf{f}_c\}_1^C$ denotes a training corpus of sentence pairs and $\mathbf{e} = Y(\mathbf{d})$. It is an appealing objective because it is continuous and differentiable, making it amenable to the use of gradient-based optimisation techniques.

However, it is *non-convex* so gradient-based methods are not guaranteed to converge to the global optimum. Another drawback is that assuming that the test time decision rule is MAP decoding then the training objective no longer matches to the test condition. Nevertheless, Och (2003) find that a minimum risk trained model does just as well as a MERT trained one on max derivation decoding.

Smith and Eisner (2006) use minimum risk training as a drop-in replacement to the MERT optimisation criterion in the standard MERT iterative training scheme. The risk, or expected loss, is computed for each sentence over an n-best list of derivations. Since the optimisation is liable to get trapped in a local optima, they use *deterministic annealing* (Rose, 1998), a procedure well suited for non-convex optimisation.

We will look at deterministic annealing in more detail in Chapter 7 but for now only give a cursory explanation. Deterministic annealing works by adding an entropic prior to the minimum loss objective. The prior is weighted by a temperature factor which is set to a high value at the start of the optimisation. At high temperature settings, the resulting annealed objective function is smooth making optimisation easier. As the temperature is gradually decreased, the impact of the prior diminishes and that of the minimum risk criterion increases so that at the end of deterministic annealing, when the temperature is close to 0, the original minimum risk objective is recovered.

In an attempt to reduce the mismatch between the minimum risk training objective and the max derivation MAP decision rule at test time, Smith and Eisner also introduce in the log-linear model a scaling factor γ which alters the shape of the distribution similarly to what is done during MBR (refer to Equation 2.19).

Setting γ to 0 results in a uniform distribution while as $\gamma \to \infty$, the distribution gets peaked toward its mode, the max derivation solution. In their implementation of minimum risk training, in addition to slowly cooling the annealing temperature, Smith and Eisner also raise γ according to a *quenching* schedule. By this process of quenching, they are able to recover at the end of the optimisation a low-entropy distribution which is peaked toward its mode and gives good performance when used for 1-best style decoding.

Another interesting aspect of Smith and Eisner's minimum risk implementation is their computation of the risk or expected loss. As discussed in Section 2.2.1, BLEU is a corpus level metric which does not decompose at a sentence level, in other words BLEU

is a *non-linear* metric. The computation of expected loss as per Equation 2.30 on the other hand involves a sentence-level decomposition. A sentence-level approximation of BLEU can and is used e.g by Zens et al. (2007) but is not guaranteed to match corpus BLEU.

To get around this potentially inexact approximation, Smith and Eisner propose to instead optimise expected $-\log$ corpus BLEU, an objective which tends to expected corpus BLEU as $\gamma \to \infty$. This new objective is approximated using a second-order Taylor series expansion. The resulting approximation is computed using the sentence-level means and variances of BLEU's sufficient statistics. The sufficient statistics correspond to n-gram precisions and hypotheses lengths. Smith and Eisner report that performance on test data is significantly better using the $-\log$ corpus BLEU approximation compared to sentence-level BLEU approximation. They also report test-time performance improvements over MERT trained models on three different datasets.

N-best list minimum risk training is performed by Zens et al. (2007) too. Similar to their maximum likelihood training methods we encountered in Section 2.5.1, they define a sentence-level and an n-gram level decomposition of the objective function. In the former, in contrast to Smith and Eisner, Zens et al. use a sentence-level variant of BLEU to compute the expected gain. For the n-gram level computation, a conditional independence assumption is made between the brevity penalty and the n-gram precisions. Zens et al. evaluate the quality of their estimated model parameters both using max-derivation and n-best MBR decoding. They find that minimum risk training outperforms both MERT and ML training, with the n-gram level decomposition performing best and that for the minimum risk trained models, MBR decoding is never worse than max-derivation decoding.

An algorithm for performing minimum risk training on a translation forest, thus leveraging information from a translation space orders of magnitude greater than that of an n-best list, is described by Li and Eisner (2009). To compute the risk, their dynamic programming algorithm requires a loss function which can factor over the edges of the translation hypergraph so they use the linear approximation to BLEU of Tromble et al. (2008). They ran experiments on a small scale data set to compare various parameter estimation algorithms when used in conjunction with max derivation decoding. They find that n-best based minimum risk training (using approximated linear BLEU) both with and without deterministic annealing performs just as well as

n-best MERT. However, hypergraph-based minimum risk training outperforms all the *n*-best methods.

In contrast to the MERT optimisation algorithm, minimum risk training can be used to train a large number of features. Li and Eisner (2009) show further improvements when they train a model with 20,000 additional sparse features.

2.5.5 Consensus Training

Pauls et al. (2009) present a novel objective function for parameter estimation called *consensus* BLEU or CoBLEU. This function aims to maximise expected counts of the *n*-grams appearing in the reference translations, the same terms involved in DeNero et al. (2009)'s consensus decoding decision rule (Section 2.4.4). Additionally, the CoBLEU objective function precisely matches the consensus decoding decision rule therefore maintaining a consistent objective through the translation pipeline.

The CoBLEU function is continuous and mostly differentiable, therefore making it amenable to gradient ascent. Pauls et al. (2009) present a dynamic programming algorithm similar to that of Li and Eisner (2009) for computing the objective function and its gradient over a *translation forest*.

They report mixed results when comparing CoBLEU tuning with MERT tuning on consensus decoding. When both tuning algorithms are initialised uniformly, CoBLEU outperforms MERT on one dataset whereas the opposite result is obtained on the second dataset. However, by initialising CoBLEU with MERT trained weights, CoBLEU gives performance improvements on both datasets. They attribute this effect of initialisation to the fact that since CoBLEU is non-convex, gradient ascent is liable to get stuck in local optima.

2.5.6 Summary

In this section, we presented a number of parameter estimation techniques proposed for log-linear SMT models. The ground we covered is by no means exhaustive. Amongst the most notable tuning techniques we omitted are perceptron-based (Liang et al., 2006) and margin-based methods (Watanabe et al., 2007; Arun and Koehn, 2007; Chiang et al., 2008b) which optimise linear models and are particularly suitable for models with a large number of features.

We began by discussing tuning methods that aim to maximise the likelihood of the training data. We subsequently noted that better test-time performance can be obtained

when using tuning objective functions that incorporate the evaluation metric to be used at test time and highlighted a few of these tuning methods. A recent trend in the SMT community is to move away from the limited information provided by n-best lists and move towards translation lattices and forests which are compact representations of the exponential number of translations processed during decoding. By leveraging this much larger space of translations, more accurate and stable parameters can be learnt. Another recent trend is the use of consensus decoding methods at test time. These methods choose a translation that is informed by the entire model distribution rather than simply picking the top scoring translation. We reviewed tuning methods that aim to optimise the terms that are involved in such consensus decoding algorithms.

2.6 Summary

In this chapter, we presented a comprehensive description of an end-to-end statistical machine translation system, in particular, a system using a phrase-based log-linear model. We focused our exposition on three parts of the SMT pipeline, namely the parameter estimation, decision rule and decoding steps. A common characteristic of these three steps is that while the traditional way they are implemented focuses on the 1-best derivation of the model, recent research has looked at utilising the predictive power of the entire distribution and has managed, in doing so, to improve translation performance.

However, significant algorithmic challenges have to be met in order to exploit the predictive power of the entire distribution. This is as a result of the complexity of the distribution which arises by the fact that its *support*, the space over which it is defined, is exponentially large. We reviewed some of the algorithms that have been proposed to tackle this challenge. The majority of these algorithms rely on approximate dynamic programming based methods.

In the next chapter, we discuss sampling methods in particular Markov Chain Monte Carlo (MCMC) techniques. These techniques have been found very useful for performing approximate inference in a principled manner in probabilistic models where the distribution is particularly unwieldy.

Chapter 3

Sampling Methods

In the previous chapter, we saw that the machine translation task can be formulated in a statistical framework allowing practitioners to exploit the rich and theoretically sound techniques of statistical learning for correct and efficient modeling of the task. Casting the translation task in terms of a log-linear model allows the principled integration of informative features to guide translation and at the same time provides techniques for estimating the weights of these features. Moreover, we can draw upon elements of statistical decision theory to help us select the optimal solution from within the exponential space of possible translations given the input sentence.

As we saw in Chapter 2, the log-linear models used at decoding time are *probabilistic models*. Given a probabilistic model, *probabilistic inference* is the act of drawing conclusions about quantities that are not observed. Examples of inference tasks are: a) computing the partition function of log-linear models b) computing the expectation of the features of a log-linear model c) computing the most likely output in the model given an input (also known as decoding).

For many probabilistic models of interest, exact inference is *intractable*. This could be because the space of all possible assignments of the variables in the model is too large. This is the case for the decoding problem in SMT, where a search over an exponential number of translations is required. Exact inference can also be intractable if the distribution of interest has a particularly complicated form.

For tractable inference, one has to resort to *approximations*. Approximations can be in the form of *heuristics* tailored to the task under consideration. For example, the intractable decoding problem in SMT is usually approximated by substituting the search for the most likely translation string with a search for the most likely derivation.

Even computing this solution exactly is intractable in most models, so a heuristic-based search algorithm, like beam search, is usually employed.

Heuristic-based approximations have the advantage that by exploiting problem-specific knowledge, one can usually come up with an efficient solution. On the other hand, even ignoring the fact that problem-specific knowledge is not easy to acquire, heuristic-based approximations often lack theoretical guarantees and do not generalise to other problems.

As alternatives to heuristic inference, there exist two main classes of general approximate inference algorithms. One is variational inference which is based on deterministic approximations. In variational inference, the intractable distribution of interest is approximated using a simpler distribution for which exact inference is possible. The downside of variational inference is obvious: finding a suitable close enough variational approximation might not be easy.

A second class of approximate inference algorithms are based on sampling methods which have the property that given infinite computational resources, they converge to the exact results. While sampling methods tend to be slower than variational inference, they are usually easy to implement. Gibbs sampling (Geman and Geman, 1984) is such an example: sampling from the joint distribution over all variables in the model is achieved by simply successively sampling each variable from its conditional distribution. Gibbs sampling is an example of a Markov Chain Monte Carlo (MCMC) algorithm.

This thesis applies sampling methods for solving various intractable inference problems in SMT. In this chapter we present background information about sampling, focusing on MCMC algorithms, demonstrating their use for two inference problems: learning and optimisation. Finally, we highlight previous applications of MCMC methods in Natural Language Processing (NLP) and in SMT.

3.1 Monte Carlo Sampling

An often essential component of many stochastic scientific problems is the evaluation of the expectation, $\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$ of function $f(\mathbf{x})$ of a multidimensional variable $\mathbf{x}$ over the probability distribution $p(\mathbf{x})$, where the distribution is defined over a space $\mathbb{S}$. If the distribution is discrete, the expectation is given by:

$$\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] = \sum_{\mathbf{x}\in\mathbb{S}} f(\mathbf{x})p(\mathbf{x}) \tag{3.1}$$

For continuous distributions, the sum in Equation 3.1 is replaced by an integral. In cases where $\mathbb{S}$ is a high-dimensional space, calculating $\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$ can be intractable.

However, if we can draw independent and identically distributed (i.i.d) random samples $\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^m$ from the distribution $p(\mathbf{x})$, these samples can be used to approximate Equation 3.1.

$$\begin{aligned} \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] &= \sum_{\mathbf{x}\in\mathbb{S}} f(\mathbf{x})p(\mathbf{x}) \\ &\approx \frac{1}{m}\sum_{j=1}^{m} f(\mathbf{x}^j), \qquad \mathbf{x}^j \sim p(\mathbf{x}) \end{aligned} \tag{3.2}$$

This procedure of approximating expectations by statistical sampling is known as *Monte Carlo sampling* (Metropolis and Ulam, 1949; Metropolis et al., 1953). The Monte Carlo estimate has three key properties:

- It is asymptotically *unbiased*.
- Its *variance* is inversely proportional to the size of the sample set.
- Its variance does not depend on the dimensionality of space being sampled.

The latter observation makes Monte Carlo sampling an especially attractive algorithm for high dimensional distributions. However, drawing samples from the desired distribution $p(\mathbf{x})$ can be hard. There are 2 main reasons for this :

1. The number of possible states in $\mathbb{S}$, the space over which the distribution is defined, can be very large. As a result, it is often unclear how to explore this state space efficiently.
2. Often the normalisation factor, $Z(\mathbf{x})$, required so that $p(\mathbf{x})$ is a well-formed probability distribution, cannot easily be computed. This is common for many of the log-linear models that are widespread in NLP and as we saw in Chapter 2, especially for the models used in Statistical Machine Translation.

Two common Monte Carlo sampling techniques for calculating expectations are importance sampling and rejection sampling. In both these techniques, samples are

drawn from a simpler *proposal distribution* instead of the more complicated target distribution. These samples are then used to calculate expectations of functions of interest, after an appropriate correction has been made to account for the difference between the proposal and target distributions. Both these methods are simple to implement but scale badly with the dimensionality of the problem. Also, for these methods to work well, the chosen proposal distribution has to be similar to the target distribution. Usually, finding such a proposal distribution can be hard.

3.2 Markov Chain Monte Carlo Sampling

Markov chain Monte Carlo (MCMC) is a very powerful framework which allows sampling from a large class of distributions, including those for which the normalisation term Z is unknown, and which is able to handle high-dimensional sample spaces. MCMC algorithms work by generating samples from a simpler proposal distribution. The probability of generating each sample is conditioned on the previous sample, forming a Markov chain. Eventually, the chain converges to the desired target distribution as its *equilibrium* distribution. Once the chain is at or close to equilibrium, independent samples can be drawn from it. However, since in practice it is hard to diagnose when the chain has converged, we can start drawing samples after allowing an initial settling in period (usually called *burn-in*). Therefore, in contrast to Monte Carlo sampling, MCMC samples tend to be correlated to each other.

Before introducing the Metropolis-Hastings and the Gibbs sampling algorithms, two of the most popular MCMC techniques and methods that we employ in this thesis, we first give a brief overview of the theory underpinning MCMC methods.

Overview

Let $\mathbf{x}^t$ denote the value of a multidimensional random variable $\mathbf{x}$ at time t and let the state space refer to the range of the possible values $\mathbf{x}$ can take. A *Markov chain* refers to a correlated sequence of states $(\mathbf{x}^0, \cdots, \mathbf{x}^n)$ generated by a Markov process. A *Markov process* is a stochastic process where the probability that the process at state $\mathbf{x}$ moves to state $\mathbf{x}'$ in a single time step is given by a transition kernel $K(\mathbf{x}' \leftarrow \mathbf{x})$ which satisfies the *Markov property*: the transition probabilities depend only on the current state $\mathbf{x}$.

The aim of MCMC methods is to construct a chain such that the desired distribution $p(\mathbf{x})$ is an *invariant distribution* of the chain. In other words, given a sample from $p(\mathbf{x})$, the marginal distribution over the next state in the chain is also the desired distribution:

$$p(\mathbf{x}') \quad = \sum_{\mathbf{x}} K(\mathbf{x}' \leftarrow \mathbf{x}) p(\mathbf{x}) \quad \text{for all } \mathbf{x}' \tag{3.3}$$

This can be achieved as long as the transition kernel K satisfies the following two conditions:

1. Irreducibility : There is a positive probability of visiting all other states starting from a given state of the Markov chain, i.e., the transition graph must be connected.

2. Aperiodicity: The chain should not get trapped in cycles, since otherwise it might never settle to an invariant distribution.

A chain that satisfies these two properties is called an *ergodic chain*. MCMC samplers are ergodic Markov chains that have the target distribution as the invariant distribution. The transition kernel is also said to leave the target distribution *stationary*. Therefore, if such a chain is run long enough (say, after a burn-in period of n steps), the samples $\mathbf{x}^{n+1}, \mathbf{x}^{n+2}, \cdots$ produced by the chain can be regarded as approximately following the target distribution. These samples remain, however, correlated to each other.

The transition kernel $K(\mathbf{x}' \leftarrow \mathbf{x})$ is usually constructed by the concatenation of simpler transition operators $O(\mathbf{x}' \leftarrow \mathbf{x})$. These base operators should all have the desired density as an invariant distribution but they do not individually have to be ergodic.

Detailed Balanced Equations

A *sufficient condition* to ensure that $p(\mathbf{x})$ is the desired stationary distribution is the detailed balance condition:

$$K(\mathbf{x}' \leftarrow \mathbf{x}) p(\mathbf{x}) = K(\mathbf{x} \leftarrow \mathbf{x}') p(\mathbf{x}') \quad \text{for all } \mathbf{x}, \mathbf{x}' \tag{3.4}$$

This states that if one starts from the stationary distribution, then a transition under K has the same probability going in one direction ($\mathbf{x} \rightarrow \mathbf{x}'$) and in the opposite direction ($\mathbf{x}' \rightarrow \mathbf{x}$). Proving detailed balance only requires considering each pair of states in

isolation, there is no sum over all states as in Equation (3.3). Summing over $\mathbf{x}$ on both sides recovers the invariant distribution requirement of Equation (3.3).

We will be using detailed balance to prove the correctness of our sampler in Chapter 5.

3.3 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm (Hastings, 1970) is the canonical and most popular MCMC algorithm. While we do not directly make use of the Metropolis-Hastings algorithm in this thesis, it is an important algorithm and can be viewed as a generalisation of Gibbs sampling, which we do use. It is also key for the understanding of the novel algorithms we propose as possible extensions to this thesis (Chapter 8). We therefore give a brief overview of this algorithm in this section.

The Metropolis-Hastings algorithm simulates a Markov chain with stationary distribution $p(\mathbf{x})$ by making use of a proposal distribution q which depends on the current state $\mathbf{x}$. This distribution can be any distribution as long as samples can be drawn from it. Unlike importance and rejection sampling, the proposal distribution *need not be similar* to the target distribution.

The Metropolis-Hastings algorithm has the convenient characteristic that is requires the target distribution be known only up to a normalisation term. In other words, assuming $p(\mathbf{x})$ can be rewritten as $\tilde{p}(\mathbf{x})/Z_p(\mathbf{x})$, the algorithm can be used as long as $\tilde{p}(\mathbf{x})$ can be easily computed for any value of $\mathbf{x}$, without requiring the potentially expensive calculation of $Z_p(\mathbf{x})$.

Algorithm 3.1 presents a pseudo-code of Metropolis-Hastings sampling. At line 3, a tentative new state $\mathbf{x}'$ is generated from the proposal distribution, conditioned only on the previous state. To decide whether to accept this new state, an acceptance ratio a is computed in line 4. Observe that the calculation of this acceptance criterion *does not* require knowledge of Z_p, the normalisation term of the target distribution. If the acceptance ratio is greater than 1, the new state is accepted (line 5) otherwise it is accepted with a probability a (line 9).

The value of $\mathbf{x}$ at the end of each iteration is retained as a *sample* from the target distribution. If the condition in line 9 is not satisfied, then the previous sample is retained meaning that adjacent samples are identical. The sequence of samples form a Markov chain, as the transition probability for going from one state to the other depends only on the previous state. However, these samples are *not independent*.

Algorithm 3.1 Metropolis-Hastings algorithm

Input: initial setting $\mathbf{x}$, number of iterations S
for $s = 1 \ldots S$ **do**
  Sample $\mathbf{x}' \sim q(\mathbf{x}' \leftarrow \mathbf{x})$
  Compute $a = \frac{\tilde{p}(\mathbf{x}')q(\mathbf{x} \leftarrow \mathbf{x}')}{\tilde{p}(\mathbf{x})q(\mathbf{x}' \leftarrow \mathbf{x})}$
  if $a \geq 1$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}'$
  else
    Sample $u \sim \text{Uniform}[0,1]$
    if $u < a$ **then**
      $\mathbf{x} \leftarrow \mathbf{x}'$
    else
      $\mathbf{x} \leftarrow \mathbf{x}$
    end if
  end if
end for

It is straightforward to prove that the Metropolis-Hastings algorithm satisfies detailed balance and that therefore the desired distribution $p(\mathbf{x})$ is invariant under the algorithm. Interested readers should consult Bishop (2007) for details.

3.4 Gibbs Sampling

Gibbs sampling (Geman and Geman, 1984) is an MCMC algorithm suitable for problems in which a joint distribution can be efficiently decomposed into a sequence of conditional distributions. Consider the n-dimensional distribution $p(\mathbf{x})$. This distribution can be rewritten as a *joint distribution* over its constituent components: $p(\mathbf{x}) = p(x_1, x_2, \ldots, x_n)$. Then Gibbs sampling resamples each component x_i of the multivariate quantity $\mathbf{x}$ by a value drawn from its conditional distribution $p(x_i|\mathbf{x}_{\setminus i})$ where $\mathbf{x}_{\setminus i}$ denotes all the dimensions $x_1, x_2, \ldots, x_n$ excepting x_i. One iteration over all the components x_i of the distribution is referred to a *scan* of the sampler. A scan can either be *deterministic* whereby all the components are visited in a fixed order or it can be *random* in which case the next component to be sampled is chosen randomly according to some distribution.

Gibbs sampling can be viewed as a special case of the Metropolis-Hastings algorithm where the conditional distributions are the proposal distributions and the acceptance ratio is always 1. It is is a popular algorithm due to its ease of implementation and the fact that it does not have any free parameters requiring tuning. For example, in cases where the target distribution is discrete and each variable can take a limited number on values, the conditional distributions can be explicitly computed:

$$p(x_i|\mathbf{x}_{\setminus i}) \propto p(x_i, \mathbf{x}_{\setminus i}) = \frac{p(x_i, \mathbf{x}_{\setminus i})}{\sum_{x_i'} p(x_i', \mathbf{x}_{\setminus i})} \tag{3.5}$$

It can be shown that Gibbs sampling draws samples from the target distribution by noting that the each sampling step leaves the joint distribution of interest invariant (Robert and Casella, 2005). For a given Gibbs sampler, if none of the conditional distributions are zero anywhere, then the sampler also satisfies the ergodicity requirement and is therefore sampling from the correct distribution.

Like the Metropolis-Hastings algorithm, the Gibbs sampler can be used to sample from distributions known only up to a normalisation constant.

3.5 Practical Considerations of MCMC

In the previous sections, we showed the theory behind the two most popular MCMC sampling algorithms. In practice, sampling can be quite tricky and getting a good sampling procedure for one's problem is akin to an art. A sampler might take too long to converge to the stationary distribution, or it might get stuck in one part of the distribution not being able to explore the rest of it. Also, some variables in the distribution might be strongly correlated, requiring careful design of proposal distributions. Practitioners also worry about the number of chains of the sampler that need to be run and ways to reduce the variance of their sampler-based estimates. In this section, we look at these issues in more detail.

Proposal Distributions The success or failure of the Metropolis-Hastings algorithm often depends on the choice of the proposal distribution. Typical Metropolis-Hasting proposal distributions are local in nature, that is they restrict moves to a small *neighbourhood* around the current state. A reason for choosing local moves is that for most high-dimensional problems, a large jump from a point in the state space is liable to end in a region of low-probability and thus be rejected leading to high correlation

between samples. On the other hand, by restricting the neighbourhood too much, the convergence of the Markov chain might be too slow. Therefore, a good proposal distribution is one whose neighbourhood trades off the benefits of high acceptance rate with those of rapid exploration of the state space.

Proposal distributions in component-wise Gibbs sampling are simple: they are simply the conditional distribution of that component given all other components. However, in some cases of Gibbs sampling such as block sampling, introduced next, the proposal distributions can be more involved.

Block Sampling A valid sampler is one whose Markov chain satisfies the ergodicity requirement. If all the conditional probability distributions in a Gibbs sampler are non-zero anywhere, then the sampler is ergodic. However, there could be cases where certain variables of the distribution are tied together: changing only one at a time could lead to a zero-probability state and thus, a non-ergodic chain. An appropriate strategy in such cases is to use a *block sampling* (Jensen et al., 1993) algorithm in which each proposal distribution samples a block of variables conditioned on the remaining variables. This strategy ensures that the sampler is ergodic while still maintaining detailed balance. In addition to being a way of satisfying ergodicity, block sampling highly correlated variables can also accelerate convergence of a Markov chain.

Mixing A key consideration in the implementation of an MCMC sampler is the number of iterations required for the chain to approach the stationary distribution. This is referred to as the *burn-in* time of the sampler. As we shall see in Section 3.5, assessing convergence of a sampler is difficult. Typically, MCMC practitioners initialise their sampler from a starting point that they deem reasonable and discard a configurable number of initial samples until they feel that the sampler has converged.

A poor choice of starting point and/or proposal distribution can greatly increase this burn-in time. A Markov chain's *mixing* rate or its *mobility* denotes how well the chain is able to explore the target distribution given the initialisation and the proposal distribution. A *poorly mixing* chain is one that stays in small regions of the state space for long periods of time while a *well mixing* one is able to explore the state space efficiently.

A chain can mix poorly if the target distribution is *multimodal*. Depending on the initialisation point, the chain can find itself stuck in one mode of the distribution and unable to move the other modes. This can happen if a move to the other mode has to go

through regions of low probability; such a move is unlikely to be seen in a simulation run of limited length.

Block sampling is one way to circumvent this problem. Another way is to run multiple Markov chains initialised with different starting points (Gelman and Rubin, 1992). Yet another way is to use annealing methods, which we discuss in Section 3.6.

Convergence Diagnostics While many graphical and statistical tests exist to assess convergence, none of these provide entirely satisfactory diagnostics (Andrieu et al., 2003; Robert and Casella, 2005). An arbitrarily chosen initial state is usually very improbable under the target distribution. Reaching a mode in high dimensions can take a long time with sophisticated optimisers, let alone a Markov chain simulation. Analyzing the chain can identify when the statistics of the sampler settle down, allowing the initial exploratory phase to be discarded. Such diagnostics could be severely misleading if there are multiple modes.

In general applications there is no way of knowing if a Markov chain has yet to find the most important regions of a probability distribution. An often employed sanity check is to run the sampler on a cut-down version of the problem where exact results are available.

Correlation Samples generated by an MCMC method are likely to be highly correlated. This effect is known as *autocorrelation*, since the correlation is between successive values in the sample set. The theory of time series analysis states if the samples are from a stationary distribution, correlated samples still provide an unbiased picture of the distribution *provided the sample size is sufficiently large* (Walsh, 2004). However, a high autocorrelation can inflate the sampling variance.

One straightforward way of reducing autocorrelation is by collecting only every m-th sample. This procedure is normally known as *thinning* and the frequency of collection is referred to as the *lag*.

Chains Given a finite amount of computing time, it is important to know how best to run the sampler. Two main strategies exist: one of them is to run one long *chain* of the sampler and the other is to run multiple short chains starting off with different initialisations points.

Both of these strategies have merits. Geyer (1992) shows that theoretically a longer chain is to be preferred. If the chains have high auto-correlations, then using a number

of small chains may result in them not being long enough to be useful. On the other hand, multiple runs do have some diagnostic value: differing results would reveal a failure to mix that might not be diagnosed from a single chain. But as Geyer (1992) argues, agreement between different runs does not provide a guarantee of good mixing.

Variance While the Monte Carlo estimate of any function of interest defined over the target distribution is *unbiased*, since it relies on a stochastic method, there is some variance associated with this estimate. One straightforward way of reducing this variance is to increase the length of the chain, since as we saw in Section 3.1, the variance decreases linearly with the size of the sample set. If the autocorrelation of the sampler is high, then thinning is a simple and effecive way to reduce variance.

3.6 Sampling for Optimisation

In Section 3.1, we motivated the use of sampling methods as a means of computing expectations on complex probability distributions. Sometimes, an *optimisation* problem can also be formulated as a Monte Carlo sampling problem. Let us imagine that instead of wanting to approximate a distribution $p(\mathbf{x})$, we want to find its global optimum or mode of the distribution. For example, if $p(\mathbf{x})$ is the posterior distribution then we often are interested in finding the maximum a posteriori (MAP) solution.

This can be done by simply running a Markov chain with $p(\mathbf{x})$ as its stationary distribution and then estimate the mode, $\hat{\mathbf{x}}$ using the drawn samples, $\mathbf{x}^i$:

$$\hat{\mathbf{x}} = \arg\max_{i=1\ldots n} p(\mathbf{x}^i) \tag{3.6}$$

It should be clear that this is a very inefficient approach. Unless most of the probability mass is centered in the vicinity of the mode, samples will only rarely be drawn from around the mode, and the sampler will waste time exploring areas of the state space of no interest. This problem can sometimes be overcome using *simulated annealing*.

Simulated annealing (Kirkpatrick et al., 1983) is a heuristic algorithm for optimising a function with local optima where standard hill-climbing approaches may trap the algorithm at a less than optimal peak. Instead of sampling from the true distribution $p(\mathbf{x})$, simulated annealing, at iteration i, samples from a distribution $p_i(\mathbf{x}) \propto p^{1/T_i}(\mathbf{x})$ where T_i is a cooling schedule. At early iterations, the temperature is set to a high value which effectively smooths the distribution and allows the Markov chain high

mobility in the state space. At T = 1, simulated annealing samples from the base distribution. As the temperature is progressively cooled to approach 0, the probability mass concentrates around the mode. The probability that the algorithm finds the global optimum approaches 1 as the annealing schedule is extended (Aarts and Laarhoven, 1987), however there is no guarantee that this optimum will be reached fast enough to make the algorithm practical. Nevertheless, simulated annealing has proved useful in many applications.

Annealing can be used with all Markov chain sampling algorithms. To sample at a temperature T only requires altering the probabilities of all states by raising them to the power $1/T$ and then renormalising to ensure the probabilities sum up to one.

3.7 Sampling in Natural Language Processing

The use of sampling techniques in Natural Language Processing (NLP) research has become prevalent in the recent past, especially with the widespread adoption of non-parametric Bayesian methods (Teh et al., 2006) as ways of modeling unsupervised learning tasks. Nonparametric models are probabilistic models in which the number of parameters is not specified a priori, allowing the number of inferred parameters to grow with the size of the data. Bayesian methods manage uncertainty in two specific ways:

- by the use of informative priors which represent beliefs of what the model should look like. The priors usually are set so as to prefer sparse solutions.
- by taking into account the entire distribution of parameters given observation data during inference.

Some examples of sampling-based nonparametric Bayesian methods applied to unsupervised learning in NLP include work on grammar induction (Johnson et al., 2007b; Cohn et al., 2009), on modeling word segmentation (Goldwater et al., 2006) and on part-of-speech tagging (Goldwater and Griffiths, 2007).

These models infer a posterior distribution of hidden variables given observed data by marginalising over parameters of the model. Integrating over all the possible values of the model parameters is intractable and is therefore approximated using MCMC methods. While this posterior distribution is often of great interest, e.g. in case of a pipeline structure where the distribution can be integrated with other probabilistic

components (Finkel et al., 2006), one can also be interested in finding the mode of this distribution. i.e., the most probable instantiation of hidden variables given the observed data. In such cases, Gibbs sampling is used as a stochastic search procedure in conjunction with simulated annealing for accelerated convergence (Goldwater et al., 2006; Goldwater and Griffiths, 2007).

Gibbs sampling as a stochastic search procedure was introduced in the NLP community by Finkel et al. (2005). In this work, non-local features are incorporated in a conditional random field (CRF) (Lafferty et al., 2001), a state of the art sequence model. Due to the use of global features in the model, exact inference using dynamic programming is no longer possible, so simulated annealing is used instead.

3.8 Sampling in Statistical Machine Translation

3.8.1 Word-based Models

The use of sampling techniques in Statistical Machine Translation (SMT) has so far been restricted to inducing translation models from parallel corpora. The earliest example of sampling in SMT can be found in the so-called IBM models for machine translation (Brown et al., 1993) which induce word alignments from a parallel corpora using the expectation-maximisation (EM) algorithm. In the case of IBM Models 1 and 2, the models are simple enough to allow for exact inference but the addition of a fertility model (Model 3) and a more sophisticated reordering model (Model 4) makes exact inference intractable.

Instead, the exponential space of possible alignments is *sampled* for highly probable alignments which are then used during EM training. Starting off with the most probable alignment under Model 2 (which can be computed exactly), a greedy hill-climbing strategy is used to find the most probable alignment under Model 3 or Model 4. This hill-climbing algorithm inspects *neighbouring* alignments to find a better alignment than the current best, where the neighbourhood consists of those alignments that differ from the current best by only a local perturbation. Since the hill-climbing strategy is a greedy algorithm, it is liable to get stuck in a local optima. Model 3/4 training therefore uses multiple initialisation points. These initialisation points are obtained by successively fixing one alignment in the sentence-pair and using Model 2 to find the most likely alignment for the remaining unaligned part of the sentence pair (this procedure is referred to as *pegging*). For each starting point, the best alignment

found using the hill-climbing heuristic as well as all its neighbouring alignments are collected for EM training.

Note that while there do exist a number of similarities between this procedure and an MCMC algorithm, it is *not* a Monte Carlo sampling algorithm. In the hill-climbing procedure, the best possible move as per the proposal distribution (the set of neighbouring alignments) is always accepted whereas in an MCMC algorithm, it will be accepted stochastically.

3.8.2 Phrase-based and Syntax-based Models

In the phrase-based model of Koehn et al. (2003), heuristics are used to induce a many word to many word alignment (also known as phrase alignment) from the word alignments obtained from training the IBM models. Marcu and Wong (2002) proposed a joint generative model trained using EM to induce phrasal alignment directly from parallel corpora without detouring via word alignments. As in the higher order IBM models, exact inference is intractable so is approximated using a hill-climbing strategy which finds high probability alignments by a sequence of local moves.

DeNero et al. (2008) revisit the joint phrasal alignment model of Marcu and Wong (2002). The latter suffers from a deficiency in that the expectations of aligned phrase-pairs under the current model, required for the E step in EM, as estimated by hill-climbing are biased. DeNero et al. (2008) propose a Gibbs sampling algorithm that explores the alignment space and computes unbiased aligned phrase-pair expectations. Note that, while previous work in NLP has used sampling so as to be able to incorporate Bayesian priors, DeNero et al. (2008)'s use of sampling is motivated by the fact that computing expectations for phrase alignments exactly is intractable. Additionally, in order to avoid degenerate solutions for their model which is likely with EM (Marcu and Wong, 2002), DeNero et al. (2008) extend their model with the addition of non-parametric Bayesian priors. Inference in this model which requires marginalising over the model parameters is done using the Gibbs sampler.

A closely related work is that of Blunsom et al. (2009) which model alignments using a Synchronous Context Free Grammar formalism. A SCFG (Lewis and Stearns, 1966) generalises context-free grammars to generate strings simultaneously in two (or more) languages. A string pair is generated by applying a series of paired rewrite rules of the form, $X \rightarrow \langle \mathbf{e}, \mathbf{f}, \mathbf{a} \rangle$ where X is a non-terminal, **e** and **f** are strings of terminals and non-terminals and **a** specfies a one-to-one alignment between non-terminals in **e** and **f**.

By assigning the source and target languages to the respective sides of a probabilistic SCFG, translation can be described as the process of parsing the source sentence, while inducing a parallel tree structure and translation in the target language (Chiang, 2007).

Blunsom et al. present a Gibbs sampler to perform inference over the latent synchronous derivation trees for the training data. Their instantiation of the SCFG formalism is a restricted one allowing only binary or ternary branching rules and disallowing rules to mix terminals and nonterminal. By restricting the expressivity of their formalism, they are able to constrain the space of possible derivation trees to be polynomial in the input rather than the exponential space of full phrasal alignments of (Marcu and Wong, 2002; DeNero et al., 2008) and therefore able to scale their model to longer sentences. The sampled alignments are used to induce a phrase-based translation model and a hierarchical phrase-based translation model which are then used for decoding test sentences.

Another Bayesian model for inducing a translation model directly from parallel corpora is presented by Cohn and Blunsom (2009). In contrast to (DeNero et al., 2008; Blunsom et al., 2009), Cohn and Blunsom (2009)'s model uses a synchronous tree substitution grammar formalism (STSG) (Eisner, 2003) for grammar induction.

3.9 Summary

In this chapter, we introduced Monte Carlo sampling as a general class of approximate inference algorithm. Sampling methods can be used to provide unbiased estimates of expectations of functions defined over the distributions of interest, even when exact inference is intractable. We studied one particular class of sampling algorithms, Markov chain Monte Carlo sampling, and showed how MCMC methods can be used to efficiently sample from any desired distribution, irrespective of the dimensionality of the space, even when the distribution is only known up to a normalising constant. We also showed how unconstrained optimisation can be formulated as a sampling problem, highlighting the use of annealing type methods in such a case.

MCMC sampling underpins the work presented in this thesis so in addition to its theoretical aspects, we addressed the practical considerations of sampling, especially ways of dealing with correlated variables and accelerating convergence of the sampler to the desired distribution.

In Chapter 5, we apply the lessons learnt about MCMC sampling to construct a sampler that can efficiently explore the state-space defined by phrase-based SMT translation model.

Chapter 4

Experimental Setup

This thesis presents a novel sampling-based framework for Statistical Machine Translation. To demonstrate the correctness of the proposed framework and to evaluate its performance in comparison it with alternate methodologies, we run translation experiments.

Before delving into an exposition of the proposed architecture, in this chapter, we describe the conditions used for the experiments conducted during the course of this thesis. By doing so, we aim to make the results presented in this thesis reproducible. The experiments and the results themselves are covered in more detail in the next three chapters.

This chapter covers the following:

- The baseline model. (Section 4.1)
- A description of all the corpora used for the experiments in this thesis (Section 4.2)

4.1 Baseline Model

The baseline model used in this thesis is a phrase-based log-linear model (Koehn et al., 2003; Koehn, 2004a). While in recent years, syntax-based models have outperformed phrase-based models in the translation of Chinese to English where significant long-distance reordering is prevalent, phrase-based models remain the best performing models for the three language-pairs we consider in this thesis (Zollmann et al., 2008; Callison-Burch et al., 2009). Additionally, the phrase-based formalism is a mature and extensively studied one for which many freely available implementations exist. In this

thesis, we use the most popular of these: Moses (Koehn et al., 2007), an open source implementation of an end to end SMT system. Moses implements the whole SMT pipeline using:

1. Word alignment and phrase extraction tools for extracting and scoring a phrase table from parallel corpora.
2. Scripts implementing *n*-best MERT for discriminative training of feature weights.
3. A phrase-based beam decoder which implements the max derivation, *n*-best MBR and lattice MBR decision rules.

Features

Moses uses a log-linear formulation of the statistical machine translation model. This formulation breaks down the modeling of the translation task in terms of a weighted combination of features of the model. The features in our baseline model are:

- A **distortion feature**, also known as linear distortion. In the decoding algorithm proposed in (Koehn, 2004a), the target string is generated from left to right while source side phrases can be reordered. The distortion feature is a mechanism to control the amount of reordering performed during decoding. The linear distortion model is a weak reordering model in that it only looks at the distance between the phrases being reordered while remaining agnostic about the identity of the phrases involved.

  In a beam decoding algorithm, under a log-linear or a linear model, the distortion feature *decomposes* over the derivation. In other words, the distortion associated with a completed translation is the sum of the distortions computed at each hypothesis expansion. Each time a hypothesis is expanded, it incurs a exponential penalty based on the reordering distance, which is the number of words skipped on the source side between the source phrase being translated and the preceding one. The penalty is then weighted by the weight associated with the linear distortion feature.

  The distortion feature is an example of a *near-local* feature. It is not a local feature since its value cannot be computed based solely on information provided by the source and the target phrase involved in the hypothesis expansion. As the additional information required can be obtained by simply keeping track of the

previously translated source phrase, we call this feature *near-local*. Non-local features impose constraints on the search algorithm's hypothesis recombination strategy (as described in Section 2.4.1.1). In the case of distortion feature, if two hypotheses differ with respect to the source position of their previously translated source phrase, they cannot be recombined.

- A **language model** feature. As we saw in Section 2.1.2, the language model provides a measure of how well formed the target sentence **e** is. An n-gram language model does so by assigning to each string a probability solely based on the words comprising it. For a string $\mathbf{e} = e_1 e_2 e_3 \cdots e_n$, this probability is mathematically given by:

$$\begin{aligned} p(\mathbf{e}) &= p(e_1 e_2 e_3 \cdots e_n) \\ &= p(e_1)p(e_2|e_1)p(e_3|e_1e_2)p(e_4|e_1e_2e_3)\cdots p(e_n|e_1e_2\cdots e_{n-1}) \end{aligned} \quad (4.1)$$

The formulation in (4.1) casts language modeling as the task of computing the probability of a word given all the words that precede it in the string. The predecessor words are also known as the *history* or the *context*. The parameters in an n-gram language model are therefore a pair composed of the context and the word whose probability is being estimated. In order to have better estimates of the model parameters, n-gram language models typically make a *Markov assumption* that only the last few words of the context affect the next word. In a language model of *order n*, the length of the context is reduced to the last n-1 words.

The probability assigned by a trigram language model is given by:

$$\begin{aligned} p(\mathbf{e}) &= p(e_1 e_2 e_3 \cdots e_n) \\ &= p(e_1)p(e_2|e_1)p(e_3|e_1e_2)p(e_4|e_1e_2e_3)\cdots p(e_n|e_1e_2\cdots e_{n-1}) \\ &\approx p(e_1)p(e_2|e_1)p(e_3|e_1e_2)p(e_4|e_2e_3)\cdots p(e_n|e_{n-2}e_{n-1}) \end{aligned}$$

In order to account for unseen or rarely seen contexts, n-gram language model parameters are typically smoothed. In our experiments, unless otherwise stated, we train trigram language models on the target side sentences of our parallel

corpora with Kneser-Ney smoothing (Kneser and Ney, 1995) using the SRILM toolkit (Stolcke, 2002).

The language model feature assigns to each target side string a score equal to the *log probability* of the string under the language model. The language model feature is a *near-local* feature since a history of the last n-1 words generated words is required to compute its value.

As an aside, we note that as the order of the language model increases, the locality of the feature becomes more global. This has a negative repercussion on the efficiency of the underlying search algorithm. Recall that a necessary condition for recombining two hypotheses is that they share the same language model context. When decoding with high order language models, fewer hypotheses recombine causing a blow-up in the search space. In order to maintain efficient decoding, more aggressive pruning is required thus increasing the chances of search errors.

The max translation decoding algorithm of Blunsom et al. (2008), discussed in Section 2.4.2.2, stores the entire generated string as context and thus is equivalent to decoding with a language model of order $n = \infty$.

- A **word penalty** and a **phrase penalty** feature. Both features are local count features indicating the number of target words generated and the number of phrase-pairs used in the derivation respectively.

  A positive word penalty feature weight biases the model to produce shorter translations while longer translations are produced when the feature weight is negative. When the phrase penalty feature weight is positive, the model prefers translations made of a large number of short phrases whereas a small number of long phrases is preferred if the feature weight is negative.

- **Translation model** features. Four translation model features are used. All four features are local to the phrase-pair being considered and consist of:

  1. The log-probability score from a probability distribution, $p(\bar{e}|\bar{f})$, mapping source phrases to target phrases. We refer to this distribution as the forward phrasal translation distribution.
  2. The log-probability score from a probability distribution, $p(\bar{f}|\bar{e})$, mapping target phrases to source phrases. We refer to this distribution as the reverse phrasal translation distribution.

3. The log-probability score from a forward lexical translation probability distribution $p_{\text{lex}}(\bar{e}|\bar{f})$.

4. The log-probability score from a reverse lexical translation probability distribution $p_{\text{lex}}(\bar{f}|\bar{e})$.

All four distributions rely on phrasal and word alignments and are computed following the procedure detailed in Koehn et al. (2003). These alignments are not usually annotated in available parallel corpora, instead, they need to be automatically induced. The IBM models presented in Section 2.1.1 produce word alignments between source and target sentences as a by-product of the modeling task. These word alignments can then be used in conjunction with heuristics to produce phrasal alignments from which phrasal distributions can be induced using simple relative frequency estimates.

The phrase extraction heuristics can be inexact and can sometimes align phrases erroneously. Since the phrase translation probabilities are calculated using maximum likelihood estimation, some distributions might be falsely peaked in the presence of rare events. A common way to smooth the effect of these distributions is to introduce lexical translation distributions (Koehn et al., 2003). In these distributions, the lexical probability of a phrase will be low if the words that comprise the source phrase are not good translations of the words in the target phrase.

The automatic word alignments were created using the GIZA++ toolkit (Och and Ney, 2003) and phrase-pairs extraction was done using the scripts provided with Moses.

Phrase Table Pruning

In phrase-based models, the translation model is in the form of a phrase table. A phrase table consists of a list of source phrases and all the possible target phrases each source phrase can translate to. A pair of source and target phrase is referred to as a phrase-pair. We sometimes refer to a phrase-pair in a phrase table as a phrase table entry. Each phrase-pair is annotated with the translation model feature values associated with it. Figure 4.1 is an example of a phrase table.

FOREIGN	ENGLISH	P(elf)	P(fle)	P_{lex}(elf)	P_{lex}(fle)
a atteint	has stretched	0.01	1	0.05	0.003
a atteint	achieved	0.30	0.10	0.21	0.002
a atteint	amounted	0.29	0.15	0.40	0.008
a atteint	rose to	0.40	0.30	0.12	0.012
a aussi	has also	0.65	0.13	0.11	0.023
a aussi	have also	0.10	0.03	0.09	0.015

Figure 4.1: Example of a phrase table. Each row in the table consists of a source phrase with its associated target phrase and translation model features.

Due to the extraction heuristics used, phrase tables can become very large thus slowing decoding. We use the following methods to reduce the size of the phrase table:

- We set the maximum source length of phrase pairs in the phrase table to be 5.
- For a given source phrase $\bar{f}$, we only include the top 20 most probable target phrases $\bar{e}$ based on the $p(\bar{e}|\bar{f})$ distribution. This is the default setup in Moses.
- The phrase table is pruned using the associated score filtering technique of Johnson et al. (2007a). The intuition behind this pruning technique is that not all of the phrase pairs in the phrase table are reliably supported by the data and that some of them might be included just as an artifact of the extraction heuristic. Johnson et al. (2007a) present an algorithm to identify and eliminate such phrase-pairs, leading to a drastic reduction (of almost 90%) of the size of the phrase table without any deterioration in BLEU.

Reordering Limit

In Section 2.4.1, we discussed that, in order to make stack decoding tractable, the amount of reordering allowed by the decoder has to be limited. Imposing a reordering limit reduces the complexity of the search space $O(I^2 2^I)$ which is exponential in the sentence length I to $O(\Lambda^2 2^\Lambda I)$ which is exponential in the reordering limit Λ but linear in I.

In preliminary experiments, we found that a Λ value of 6 gave the best trade-off between speed and accuracy. Therefore, for the experiments in thesis we set Λ to 6.

4.2 Corpora

The translation experiments in the thesis are performed on the following three language pairs:

- Arabic to English.
- French to English.
- German to English.

In a SMT experimental setup, we can distinguish between five types of datasets. These are:

- The *parallel training corpus* used to train the *translation model*.
- The *monolingual training corpus* used to train the target side *language model*.
- The *parallel tuning corpus* used for *parameter estimation*.
- Optionally, a *parallel held-out corpus* which is used to evaluate the quality of estimated model parameters.
- One or many test *parallel test sets* used to evaluate the final translation performance of the SMT system.

The parallel training corpora consists of one target sentence for each source sentence. However, the tuning, held-out and test corpora can have multiple target sentences, also known as *references*, for each source sentence. We indicate the number of references available for each of the corpora used.

The Arabic-English training corpus is almost a third in size to the corpora used for the other two language pairs. The use of this smaller corpus is to allow us a rapid experimental life-cycle and also to ascertain whether the techniques presented in this thesis work both for small scale, in the case of Arabic-English, and large scale, in the case of French-English and German-English, data sets.

Corpus	Sentence pairs	Source words	Average Target words	Source vocab size	Target vocab size
Training	288,093	8,478,652	9,280,345	143,397	87,143
Tuning - MT02	1,043	28,553	33,503	6,474	10,443
Held-out - MT03	663	17,815	19,821	4,684	5,478
Test - MT05	1,056	31,375	36,173	6,950	7,081

Table 4.1: Corpus statistics for Arabic to English translation task in terms of number of sentence pairs, number of source words and the size of the source vocabulary for each data set. Average Target words is the number of target words averaged over all references in case of multi-reference data sets. The target vocabulary size is computed by aggregating all references.

4.2.1 Arabic to English

We present statistics for the corpora used for Arabic to English translation experiments in this section. We refer to this language pair as *AR-EN*. The data used in these experiments are obtained from the Linguistic Data Consortium (LDC)[1] and from the annual machine translation evaluation workshops organised by the National Institute of Standards and Technology (NIST)[2].

The training corpus is a subset of 288,093 sentence-pairs drawn from the training data made available for the NIST workshop. It consists of the eTIRR corpus (LDC2004E72), the Arabic news corpus (LDC2004T17), the Ummah corpus (LDC2004T18), and the sentences with confidence $c > 0.995$ in the ISI[3] automatically extracted web parallel corpus (LDC2006T02).

The tuning corpus is NIST's MT02 dataset. It consists of 1,043 Arabic sentences with 10 English reference translations for each source sentence. The 663 sentence long MT03 dataset is used as held-out for some experiments and as test set for some other experiments. It includes 4 reference translations per source sentence. The blind test set comes from the NIST's MT05 dataset. It is a 4 reference, 1,056 sentence long corpus.

In Table 4.1, we provide statistics for each of the datasets detailing the number of sentence pairs in each, as well as the number of source words, target words, source

[1] www.ldc.upenn.edu
[2] www.itl.nist.gov/iad/mig/tests/mt/
[3] www.isi.edu

tokens and target tokens in each corpus. For the target side statistics of the multi-reference test sets, we present the average number of target words per reference set (Average Target words) and the number of distinct target words aggregated over all the references (Target vocab size).

4.2.2 French/German to English

For the French to English (*FR-EN*) and German to English (*DE-EN*) translation experiments, we use data made available for the WMT09 shared translation task (Callison-Burch et al., 2009). This consists of data from the proceedings of the European Parliament. These proceedings are commonly known as *Europarl*. We use the training corpus from WMT09 to train the translation model and use its target side for training a trigram language model. Tuning is performed using the DEV2006 set. We use TEST2007A as test set for some preliminary experiments and use it as held-out set for further experiments. TEST2008A is used as blind test set for final experiments.

Since in most realistic applications of MT, the test data comes from a domain different to that of the training data, the WMT09 shared task also provides an additional test set composed of data drawn from a domain different to parliamentary proceedings. This *out-of-domain* test set consists of news stories taken from major news outlets as the BBC, Der Spiegel, Le Monde, etc. during the time period of November-December 2007. The purpose of this data set is to evaluate the generalisation capacity of the SMT model when made to translate data from a domain different to that which was used during training and tuning. We use a subset of the out-of-domain data as an additional test set. This test set is referred to as NEWSDEV2009B. All the data sets used only contain one reference translation per source sentence.

We present detailed statistics for *FR-EN* and *DE-EN* in Table 4.2. These statistics also include the percentage of out of vocabulary (OOV) words in each data set. The OOV rate is computed by dividing the number of word types in a given data set which are not in the training data by the total number of word types in the data set. We expect that the OOV rate will be lower for *in-domain* data sets (indicated as Test-In in the Table 4.2), since both the training and test sets are drawn from parliamentary proceedings, and higher for out-of-domain data sets (indicated as Test-Out in the Table 4.2). This is borne out by the figures in Table 4.2.

Corpus	Sentence pairs	Source words	Target words	Source vocab size	Target vocab size	% OOV
Training	1,393,452	41,722,058	37,457,241	123,732	102,418	NA
Tuning	2,000	63,265	58,055	7,289	6,116	1.0
Held-out	1,000	31,981	29,493	4,789	4,019	1.0
Test - In	1,000	33,931	31,124	5,007	4,221	0.8
Test - Out	1,026	27,929	25,049	5,936	5,362	9.3

Corpus	Sentence pairs	Source words	Target words	Source vocab size	Target vocab size	% OOV
Training	1,388,758	35,755,222	37,742,299	310,753	100,565	NA
Tuning	2,000	55,118	58,761	8,794	6,116	3.1
Held-out	1,000	27,675	29,493	5,226	4,019	2.0
Test - In	1,000	29,059	31,124	5,797	4,221	2.1
Test - Out	1,026	23,931	25,049	6,701	5,362	14.9

Table 4.2: Corpus statistics for French to English (top) and German to English (bottom) translation task in terms of number of sentence pairs, number of source words, number of target words, size of the source vocabulary, size of target vocabulary and % OOV for each data set. For each data set, out of vocabulary statistics are computed as the ratio of word types in the data set which are not in the training data divided by the total number of word types in the data set. All data sets are single reference sets.

Chapter 5

A Gibbs Sampler for Phrase-based Statistical Machine Translation

In Chapter 3, we saw that Markov Chain Monte Carlo sampling (MCMC) sampling techniques can be used for theoretically sound approximate probabilistic inference in distributions where exact inference is not practical. For a number of tasks in all but the most trivial of Statistical Machine Translation (SMT) models, exact inference is intractable too. Such intractable tasks include MAP and MBR decoding (Section 2.3) as well as minimum risk training and conditional log-likelihood training (Section 2.5).

Most translation equivalence models define multiple derivations for each translation. Reasoning over a distribution over translations rather than derivations, as required by the above mentioned tasks, requires marginalising out these derivations, a procedure too slow and too memory intensive to be practical. Instead, a common approximation is to just use the distribution over derivations. The implication of this approximation is that the probability (or score in case of an unnormalised model) of a string is approximated by the probability (or score) of the most probable derivation that yields the string.

This so called "max derivation approximation" remains intractable but can be computed in polynomial time via approximate dynamic programming (DP) methods. While fast and effective for many problems, it has two serious drawbacks for probabilistic inference. First, in the unnormalised probabilistic models common in SMT, the error incurred by using the score of the max derivation approximation instead of the score of the translation, is unbounded. Second, the DP solution requires substantial pruning and restricts the use of non-local features. The latter problem persists even in the variational approximations of Li et al. (2009b), who attempt to solve the former.

In this chapter, we propose MCMC sampling as a way of addressing these problems. In Section 5.3, we describe a novel Gibbs sampling algorithm that draws samples from the posterior distribution over derivations, $p(\mathbf{e},\mathbf{d}|\mathbf{f})$, of a phrase-based translation model. The generated samples can then be used to provide theoretically sound solutions for a number of inference tasks. We then present formal (Section 5.4) and experimental (Section 7.1.1) evidence to confirm that the sampler is able to effectively explore the exponential state space of the translation model in a reasonable amount of time. We also draw attention to limitations of the proposed approach.

5.1 Overview

We begin by assuming a phrase-based translation model in which the input sentence $\mathbf{f}$ of length m is segmented into phrases which are sequences of adjacent words. These phrases are not necessarily linguistically motivated. Each foreign phrase is translated into the target language to produce an output sentence $\mathbf{e}$ and an alignment $\mathbf{d}$ representing the mapping from source to target phrases. Phrases are allowed to be reordered during translation.

The model is defined in a log-linear form, with a vector of feature functions h and parametrised by weight vector λ, as described in Koehn et al. (2003).

$$\begin{aligned} p(\mathbf{e},\mathbf{d}|\mathbf{f};\lambda) &= \frac{\exp\left[s(\mathbf{e},\mathbf{d},\mathbf{f})\right]}{\sum_{\langle \mathbf{e}',\mathbf{d}'\rangle}\exp\left[s(\mathbf{e}',\mathbf{d}',\mathbf{f})\right]} \\ \text{where } s(\mathbf{e},\mathbf{d},\mathbf{f}) &= \lambda\cdot h(\mathbf{e},\mathbf{d},\mathbf{f}) \end{aligned} \tag{5.1}$$

The features h of the model are described in Chapter 4. There is a further parameter Λ, the reordering limit, that limits how many source language words may intervene between two adjacent target language phrases. As explained in Chapter 4, for the experiments in this thesis, we use $\Lambda = 6$.

For inference in this model, we use Markov chain Monte Carlo (MCMC) sampling. MCMC probabilistically generates sample derivations from the complete search space. The probability of generating each sample is conditioned on the previous sample, forming a Markov chain. Given a suitable initialisation point, eventually this chain will converge to the desired distribution, $p(\mathbf{e},\mathbf{d}|\mathbf{f})$.

The MCMC algorithm we use is Gibbs sampling (Geman and Geman, 1984) which obtains samples from the joint distribution of a set of random variables $X =$

$\{X_1, \ldots, X_n\}$ by sampling each variable at a time from the conditional distribution of the variable given all other variables.

The Gibbs sampler produces a sequence of N samples of derivations drawn from the desired distribution $p(\mathbf{e}, \mathbf{d}|\mathbf{f})$. This sample set $\mathcal{S}_1^N = (\mathbf{e}_1, \mathbf{d}_1, \mathbf{f}) \ldots (\mathbf{e}_N, \mathbf{d}_N, \mathbf{f}) = \{(\mathbf{e}_i, \mathbf{d}_i, \mathbf{f})\}_{i=1}^N$, where a derivation is denoted as the triplet $(\mathbf{e}_i, \mathbf{d}_i, \mathbf{f})$, can then be used to estimate the expectation of any function $g(\mathbf{e}, \mathbf{d}, \mathbf{f})$ under the distribution as follows:

$$\mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f})}[g] \approx \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{e}_i, \mathbf{d}_i, \mathbf{f}) \qquad (\mathbf{e}_i, \mathbf{d}_i, \mathbf{f}) \sim p(\mathbf{e}, \mathbf{d}|\mathbf{f}) \tag{5.2}$$

Equation 5.2 is the *Monte Carlo estimate* which we encountered earlier in Section 3.1. It can be used to provide an estimate of $p(\hat{\mathbf{e}}, \hat{\mathbf{d}}|\mathbf{f})$, the probability of a derivation $(\hat{\mathbf{e}}, \hat{\mathbf{d}}, \mathbf{f})$ by taking $g(\mathbf{e}, \mathbf{d}, \mathbf{f})$ to be an indicator function of the form:

$$g(\mathbf{e}, \mathbf{d}, \mathbf{f}) = \begin{cases} 1 & \text{if } \mathbf{e} = \hat{\mathbf{e}} \text{ and } \mathbf{d} = \hat{\mathbf{d}} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

Likewise, we can obtain a Monte Carlo estimate of $p(\hat{\mathbf{e}}|\mathbf{f})$ by using an indicator function $g(\mathbf{e}, \mathbf{d}, \mathbf{f})$ which marginalises over all derivations:

$$g(\mathbf{e}, \mathbf{d}, \mathbf{f}) = \begin{cases} 1 & \text{if } \mathbf{e} = \hat{\mathbf{e}} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

Similarly, if g is an indicator function over a feature in the model, then Equation 5.2 calculates the expectation of the feature under the distribution, a quantity useful for some discriminative training algorithms.

Discussion

At this point, we would like to pause and draw the reader's attention to the magic of Monte Carlo sampling. Equation 5.2 tells us that, provided the samples are drawn from the distribution of interest, a finite number of samples is enough to give us an *unbiased* estimate of $p(\hat{\mathbf{e}}, \hat{\mathbf{d}}|\mathbf{f})$ or $p(\hat{\mathbf{e}}|\mathbf{f})$ or of any other function which can be written in the form of an expectation *without explicitly computing* $Z(\mathbf{f})$.

In contrast, previous methods (Smith and Eisner, 2006; Zens et al., 2007; Li and Eisner, 2009) run DP with extensive pruning, then compute expectations of functions of interest by first computing Z over the resulting hypergraph or over an n-best list of derivations extracted from the hypergraph. Since this hypergraph is pruned, often

severely so, the expectations computed are not on the true distribution but on one that is unknown and potentially biased (Blunsom and Osborne, 2008).

This is not to say that sampling is the panacea for all inference ills in SMT. The Monte Carlo estimate might be unbiased but its variance can be high. As discussed in Section 3.5, variance can be reduced by using a large sample set containing few correlated samples. This requires an efficient sampler from which a large number of samples can be drawn in a reasonable amount of time. Designing such a sampler for the large structured space of an SMT model presents non-trivial challenges.

5.2 Sampling in Structured Spaces

Before diving in the description of our proposed sampler, we would like to draw attention to some of the challenges in designing such a sampler. Refer to Lopez (2010) for a much more thorough discussion on the subject.

Many NLP tasks such as parsing or machine translation are examples of *structured problems*. Though there is not a clear cut agreed upon definition of the term, a structured problem can be seen as one where we are predicting the values of multiples variables and the output variables in the model are mutually dependent or constrained. These dependencies and constraints reflect the structure inherent in the problem domain. For example, in the case of parsing, a valid assignment of the variables in the model has to respect, among other constraints, the constraint that the resulting parse tree be rooted in a non-terminal labelled with an **S** symbol.

The presence of these dependencies and constraints poses problem to MCMC algorithms. For example, the usual way of doing Gibbs sampling, which consists of sampling each variable at a time from its conditional distribution, is inappropriate since it might change the value of a variable to a new value which breaks the constraints of the model.

Recent work on applying MCMC sampling to the structured problem of modeling SMT phrase alignment (DeNero et al., 2008; Blunsom et al., 2009) has done so by using the following algorithm: assuming a joint distribution $P(X,Y)$ over input variables $X \in \mathbf{X}$ and output variables $Y \in \mathbf{Y}$, draw a new sample Y' given current sample Y from a small set of neighbours $N \subset \mathbf{Y}$ with probability proportional to $P(Y'|X)$. The combination of N and $P(Y'|X)$ is called an *operator*. The sampler may use several operators.

This algorithm bears a stark similarity to the usual implementations of the Metropolis-Hasting algorithm. The neighbourhood N which consists of samples that are a *local change* away from the current sample can be viewed as an example of typical *proposal distributions* used during Metropolis-Hastings sampling. There is one major difference though: in Metropolis-Hastings, the proposal distribution is different from the target distribution, whereas in the algorithm of (DeNero et al., 2008; Blunsom et al., 2009), the proposal distribution is the target distribution itself. As such, as long as the conditions of irreducibility and aperiodicity are satisfied, the algorithm of (DeNero et al., 2008; Blunsom et al., 2009) is a Gibbs sampler.

The term Gibbs operator is introduced by DeNero et al. (2008) but it is very closely related to the concept of *block sampling*, which we previously encountered in Section 3.5. To see this, notice that the set of variables in an operator's neighbourhood can be equivalently mapped to a block of variables in a block sampler.

The sampling algorithm for structured spaces therefore comes down to constructing appropriate neighbourhoods or blocks of variable which allow rapid exploration of the entire space without breaking the deterministic dependencies and constraints in the model.

5.3 Sampler Description

Our sampler consists of simple operators which when concatenated together enable it to efficiently explore the entire distribution of a phrase-based translation model. Each operator proposes a small change to the existing translation; the likelihood of accepting the change is proportional to the conditional probability of the change with respect to the unchanged remainder of the translation. Given an initial sample, an *iteration* of the sampler will apply each operator at each possible point in the sentence. A new sample is then collected.

The sampler consists of three operators. RETRANS varies the translation of a single source phrase. Segmentation, alignment, and all other target phrases are held constant. MERGE-SPLIT varies the source segmentation at a single word boundary. If the boundary is a segmentation point in the current hypothesis, the adjoining phrases can be merged, provided that the corresponding target phrases are adjacent and the phrase table contains a translation of the merged phrase. If the boundary is not a segmentation point, the covering phrase may be split, provided that the phrase table contains a translation of both new phrases. Remaining segmentation points, phrase

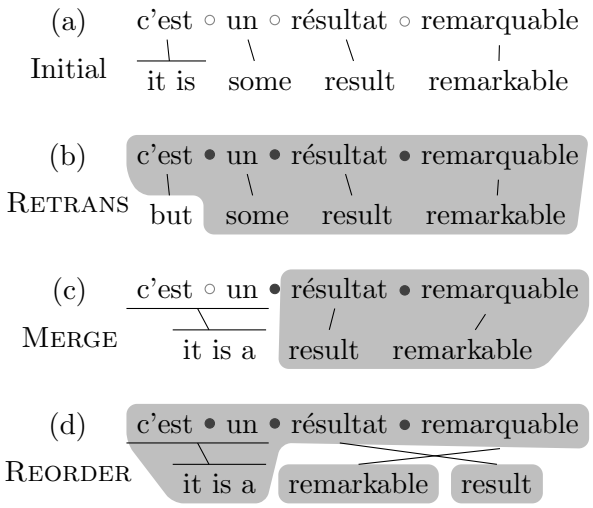


Figure 5.1: Example evolution of an initial hypothesis via application of several operators. Variables that stay constant during each sampling step are indicated by shading.

alignment and target phrases are held constant. REORDER varies the target phrase order for a pair of source phrases, provided that the new alignment does not violate reordering limit Λ. Segmentation, phrase translations, and all other alignments are held constant. Figure 5.1 illustrates sampling using the operators in our model.

The log-linear model of Equation 5.1 is effectively defined over the features of the phrase-pairs and alignments involved in the translation. While the RETRANS and REORDER operators keep the number of phrase-pairs/alignments used in a translation constant, the MERGE-SPLIT operator can vary this number. However, Gibbs sampling is suitable only for a distribution over a fixed number of variables. If the dimensionality is variable, then we must use alternate methods such as reversible-jump Monte Carlo (Green, 1995). To show that we are actually computing a distribution on a fixed number of variables, we will use an alternate representation. We must first formally define some variables and notations.

- Let i and j be inter-word source indices where $0 \leq i \leq j \leq m$ and m is the length of the source sentence.
- Let $[i, j]$ denote a source span. The left frontier of the span denotes position i and its right frontier refers to position j.
- A source span is *active* if $[i, j]$ is a current segmentation in source sentence $\mathbf{f}$ and is *inactive* otherwise.

- Let $\mathbf{f}_{ij}$ be the source phrase spanning $[i, j]$ in source sentence $\mathbf{f}$.
- Let E represent a target side phrase and $\mathcal{E}$ the set of all target side phrases.
- Then, $T_{[i,j,E]}$ is an indicator variable defined as follows.

$$T_{[i,j,E]} = \begin{cases} 1 & \text{if } \mathbf{f}_{ij} \text{ translates to E } \textit{as one phrase} \text{ in the translation } \mathbf{f} \rightarrow (\mathbf{e}, \mathbf{d}) \\ 0 & \text{otherwise} \end{cases}$$

  In other words, $T_{[i,j,E]}$ denotes a phrase-pair with $\mathbf{f}_{ij}$ as its source and E as its target.

- Let T consist of all $T_{[i,j,E]}$ variables.
- Let k and l be inter-word source indices where $0 \leq k, l \leq m$ and m is the length of the source sentence.
- Let $S_{[k,l]}$ be an indicator variable defined as follows.

$$S_{[k,l]} = \begin{cases} 1 & \text{if a span with right frontier } k \text{ is translated immediately before} \\ & \text{a span with left frontier } l \text{ in the translation } \mathbf{f} \rightarrow (\mathbf{e}, \mathbf{d}) \\ 0 & \text{otherwise} \end{cases}$$

- Let S consist of all $S_{[k,l]}$ variables.

The $T_{[i,j,E]}$ variables represent phrase pairs involved in a translation and the $S_{[k,l]}$ variables capture the alignment sequence of these phrase pairs. We denote an indicator variable with value equal to 1 as *active* and *inactive* otherwise.

We cannot freely assign any set of values to our variables. There are several constraints. Firstly, there can only be one active phrase-pair variable per *active source span*.

$$\sum_{E \in \mathcal{E}} T_{[i,j,E]} = 1, \forall i, j : [i, j] \text{ is an active source span} \tag{5.5}$$

Second, only one alignment variable may be active for the right frontier of a span; likewise for the left frontier.

$$\sum_{l'} S_{[k,l']} = 1, \forall k : \text{right frontier of an active source span} \tag{5.6}$$

$$\sum_{k'} S_{[k',l]} = 1, \forall l : \text{left frontier of an active source span} \tag{5.7}$$

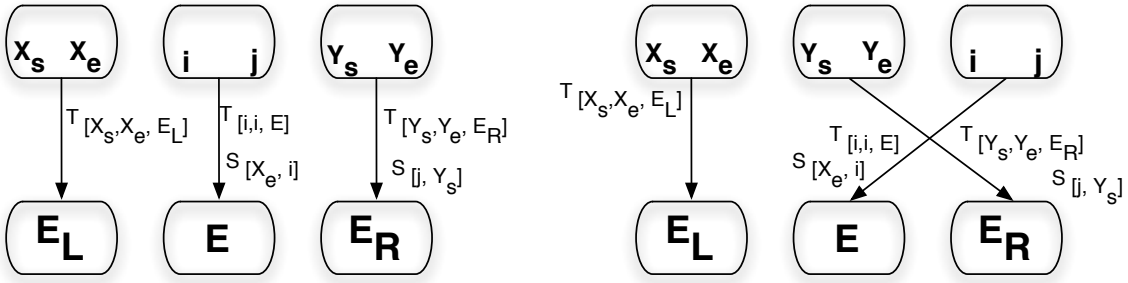


Figure 5.2: The left hand side diagram shows a monotone translation. Figure on the right shows a translation with source side reordering. Source phrases are annotated with their spans. Each translation step is annotated with its associated active phrase-pair and alignment variables. For example, translating source words spanning $[i, j]$ to target phrase E is captured by the phrase-pair variable $T_{[i,j,E]}$ and the alignment variable $S_{[X_e,i]}$, where X_e is the end position of the source span of the target phrase translated immediately before E.

Given valid configurations of T and S, we can reconstruct the derivation $(\mathbf{e}, \mathbf{d}, \mathbf{f})$. Figure 5.2 gives an example of two translation hypotheses annotated with active phrase-pair and alignment variables.

Features $h(\mathbf{e,d,f})$ in Equation 5.1 can be decomposed into simpler functions depending on mostly local information. Assume a phrase-based model with 4 such features, where the features correspond to the ones described in Section 4.1:

1. A translation model feature with weight λ_T and score $h_T(E, \mathbf{f}_{ij})$.

2. A word penalty feature with weight λ_W and score $h_W(E)$.

3. A linear distortion feature with weight λ_D and score $h_D(j, i)$.

4. A language model (LM) feature with weight λ_L. The LM contribution of phrase-pair $T_{[i,j,E]}$, given the alignments S in the translation, is represented as the triple $h_L(E, e_-^{[S,i,j]}, e_+^{[S,i,j]})$ where $e_-^{[S,i,j]}$ encodes the LM pre-context of $T_{[i,j,E]}$ and $e_+^{[S,i,j]}$ its LM post-context.

   Given a language model of order n, the LM pre-context and the LM post-context of a target phrase are the n-1 target words preceding it and the n-1 target words following it in the translation string respectively.

The model in Equation 5.1 can now be factorised as:

$$\begin{aligned} p(\mathbf{e},\mathbf{d}|\mathbf{f};\lambda) &= p(T,S|f;\lambda) \\ &\propto \exp\left[\lambda \cdot h(\mathbf{e},\mathbf{d},\mathbf{f})\right] \\ &= \exp\ \lambda_T \sum_{T_{[i,j,E]} \in T} \left[T_{[i,j,E]} h_T(E,\mathbf{f}_{ij})\right] \cdot \exp\ \lambda_W \sum_{T_{[i,j,E]} \in T} \left[T_{[i,j,E]} h_W(E)\right] \cdot \\ &\quad \exp\ \lambda_D \sum_{S_{[j,i]} \in S} \left[S_{[j,i]} h_D(j,i)\right] \cdot \\ &\quad \exp\ \lambda_L \sum_{T_{[i,j,E]} \in T} \left[T_{[i,j,E]} h_L(E, e_-^{[S,i,j]}, e_+^{[S,i,j]})\right] \end{aligned} \tag{5.8}$$

Since the model is defined over fixed-length T and S, we can apply Gibbs sampling to it. In basic Gibbs sampling we would deterministically scan the variables left-to-right, resampling each in turn. However, due to the deterministic constraints between variables, we use a block sampling strategy whereby mutually constrained variables are sampled together. To do this we define blocks of variables that allow us to vary their assignments while respecting the constraints in Equations 5.5, 5.6 and 5.7, respectively:

1. Let $T_{[i,j]}$ be the set of all phrase-pair variables spanning $[i,j]$.
2. Let $S_{[j,-]} = \{S_{[j',i']} | j' = j\}$ be the set of all alignment variables such that j is the right frontier of a source phrase translated immediately before another phrase.
3. Let $S_{[-,i]} = \{S_{[j',i']} | i' = i\}$ be the set of all alignment variables such that i is the left frontier of a source phrase translated immediately after another phrase.

We are now in a position to formally describe the operators.

5.3.1 RETRANS

The RETRANS operator changes the translation of a single source phrase. Segmentation, alignment, and all other target phrases are held constant.

Assume we want to sample a new target phrase for the active span $[I,J]$ and that the set of phrase table entries translating source phrase $\mathbf{f}_{IJ}$ is given by $\{\langle \mathbf{f}_{IJ}, E_1\rangle, \langle \mathbf{f}_{IJ}, E_2\rangle \cdots \langle \mathbf{f}_{IJ}, E_n\rangle\}$. The block $T_{[I,J]}$ therefore consists of the variables $\{T_{[I,J,E_1]}, T_{[I,J,E_2]} \cdots T_{[I,J,E_n]}\}$.

A new phrase-pair $T_{[I,J,E_e]}$ $(1 \leq e \leq n)$ is then sampled with probability:

$$p(T_{[I,J,E_e]} \mid T_{[I\setminus,J\setminus]}, S) = \frac{p(T_{[I,J,E_e]}, T_{[I\setminus,J\setminus]}, S)}{\sum_{i=1}^{n} p(T_{[I,J,E_i]}, T_{[I\setminus,J\setminus]}, S)} = \frac{s(T_{[I,J,E_e]}, T_{[I\setminus,J\setminus]}, S)}{\sum_{i=1}^{n} s(T_{[I,J,E_i]}, T_{[I\setminus,J\setminus]}, S)} \tag{5.9}$$

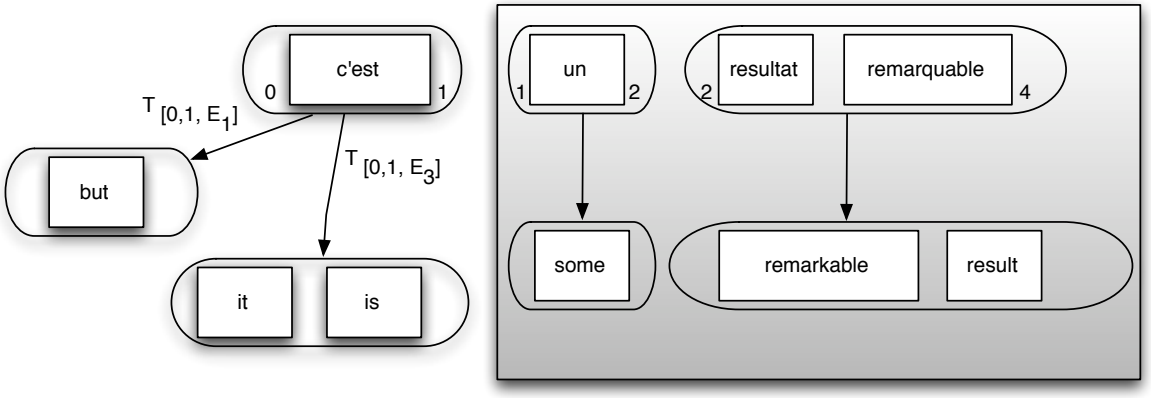


Figure 5.3: Example of a RETRANS step for block $T_{[0,1]} = \{T_{[0,1,E_1]}, T_{[0,1,E_3]}\}$ where E_1 = "but" and E_3 = "it is". The variables $T_{[0,1,E_1]}$ and $T_{[0,1,E_3]}$ correspond to the phrase-pair ⟨"c'est", "but"⟩ and ⟨"c'est", "it is"⟩ respectively. Source phrases are annotated with their spans. The shaded box covers all variables that stay constant during the sampling step. All alignment variables stay fixed.

where $T_{[i\backslash,j\backslash]} = \{T_{[i',j',E]} |\ i' \neq i \text{ and } j' \neq j\}$ is the set consisting of all phrase-pair variables that do not span $[i, j]$ and S is the set consisting of all alignment variables. The denominator normalises the probabilities so that they sum up to one.

Equation 5.9 is the basic Gibbs sampling operation: the generation of a random value for some variable from its conditional distribution given the current values of all other variables. Neal (1993) mentions that "the speed of the algorithm depends crucially on whether this operation can be done quickly." As defined in (5.9), this operation requires computing the scores s of *entire derivations* which is clearly expensive. Note that we can get away from computing Z since it cancels out in the numerator and denominator.

However, we can factorise the joint distribution (Equation 5.8) as a product of variables resampled by RETRANS ($T_{[I,J,E_e]}$) and constant variables (all other phrase pair and alignment variables). The constant terms cancel so that Equation 5.9 simplifies to:

$$p(T_{[I,J,E_e]} \mid T_{[I\backslash,J\backslash]}, S) = \frac{\exp\left[\lambda_T h_T(E_e, \mathbf{f}_{IJ}) + \lambda_L h_L(E_e, e_-^{[S,I,J]}, e_+^{[S,I,J]}) + \lambda_W h_W(E_e)\right]}{\sum_{i=1}^{n} \exp\left[\lambda_T h_T(E_i, \mathbf{f}_{IJ}) + \lambda_L h_L(E_i, e_-^{[S,I,J]}, e_+^{[S,I,J]}) + \lambda_W h_W(E_i)\right]} \tag{5.10}$$

We have therefore reduced the basic sampling operation for the RETRANS operator to that of only computing the scores of the phrases in the block. If all the features

in the model are local, then the score of a phrase can be computed in advanced, greatly speeding up sampling. In our model, we have two non-local features: the linear distortion feature and a language model feature. The former remains constant during RETRANS so can be ignored. On the other hand, computing the language model contribution of each phrase is an expensive operation since it cannot be pre-computed. This is because, given a language model of order n, the computation requires knowledge of the n-1 words in the target sentence preceding the target phrase and the n-1 words in the target sentence following the target phrase, words which are liable to change after each sampling step.

We now illustrate the RETRANS operator with an example.

Example Figure 5.3 shows an example of the RETRANS operator. We want to sample from $T_{[0,1]} = \{T_{[0,1,E_1]}, T_{[0,1,E_3]}\}$ where $\mathbf{f}_{0,1} =$ "c'est", $E_1 =$ 'but" and $E_3 =$ "it is". Assuming a bigram language model with the start of sentence marker denoted by $\langle s \rangle$ and setting all feature weights to 1.
Then, $T_{[0,1,E_1]}$ is chosen with probability:

$$P(T_{[0,1,E_1]} \mid T_{[I\setminus,J\setminus]}, S) = \frac{\exp\left[h_T(\text{“but”}, \text{“c'est”}) + h_L(\text{“but”}, \langle s \rangle, \text{“some”}) + h_W(\text{“but”})\right]}{Z'}$$

and $T_{[0,1,E_3]}$ is chosen with probability:

$$P(T_{[0,1,E_3]} \mid T_{[I\setminus,J\setminus]}, S) = \frac{\exp\left[h_T(\text{“it is”}, \text{“c'est”}) + h_L(\text{“it is”}, \langle s \rangle, \text{“some”}) + h_W(\text{“it is”})\right]}{Z'}$$

$$\begin{aligned} \text{where } Z' &= \exp\left[h_T(\text{“but”}, \text{“c'est”}) + h_L(\text{“but”}, \langle s \rangle, \text{“some”}) + h_W(\text{“but”})\right] \\ &+ \exp\left[h_T(\text{“it is”}, \text{“c'est”}) + h_L(\text{“it is”}, \langle s \rangle, \text{“some”}) + h_W(\text{“it is”})\right] \end{aligned}$$

5.3.2 REORDER

The REORDER operator varies the target phrase order for a pair of source phrases, provided that the new alignment does not violate the reordering limit Λ. Segmentation, phrase translations, and all other alignments are held constant. It takes a pair of source spans $[i, j]$ and $[k, l]$ and samples new values for the alignment variables from the blocks $S_{[-,i]}$, $S_{[-,k]}$, $S_{[j,-]}$ and $S_{[l,-]}$, such that reordering limit constraints are respected.

There are two possible outcomes to each REORDER operation: maintain the current alignments or swap the alignments.

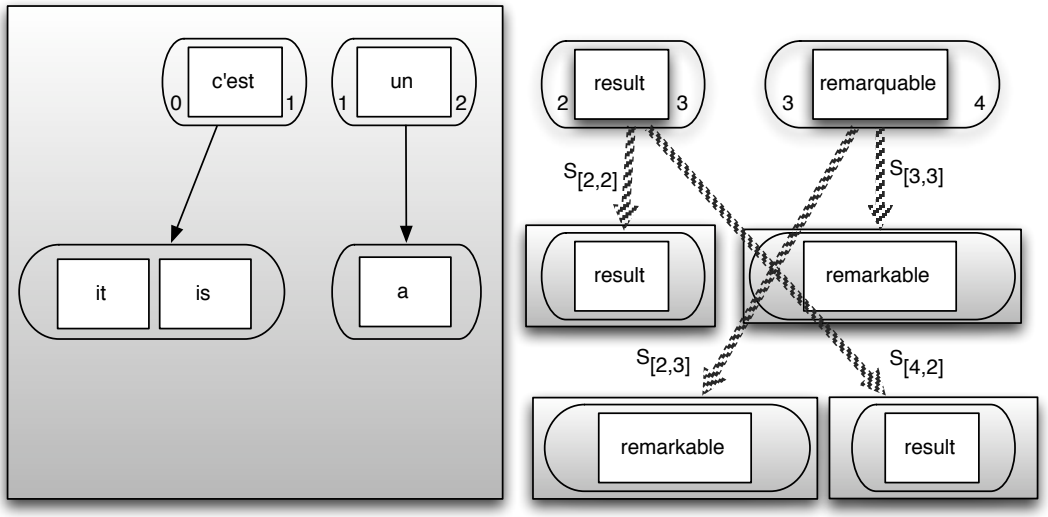


Figure 5.4: Example of a REORDER step for source spans [2,3] and [3,4]. The operator considers a monotone alignment (activating $S_{[2,2]}$ and $S_{[3,3]}$) and a reordered alignment (activating $S_{[2,3]}$ and $S_{[4,2]}$). Source phrases are annotated with their spans. Shaded boxes cover all variables that stay constant during the sampling step. All phrase-pair variables stay fixed.

Assume current active alignments $S_{[x_1,i]}$, $S_{[j,x_2]}$, $S_{[x_3,k]}$, and $S_{[l,x_4]}$ and proposed swapped alignments $S_{[x_3,i]}, S_{[j,x_4]}, S_{[x_1,k]}$ and $S_{[l,x_2]}$. The required conditional probabilities are:

$$\begin{aligned} p(S_{[x_1,i]}, S_{[j,x_2]}, S_{[x_3,k]}, S_{[l,x_4]} | S\backslash, T) &= p(S_{[x_1,i]}, S_{[j,x_2]}, S_{[x_3,k]}, S_{[l,x_4]}, S\backslash, T)/Z' \\ p(S_{[x_3,i]}, S_{[j,x_4]}, S_{[x_1,k]}, S_{[l,x_2]} | S\backslash, T) &= p(S_{[x_3,i]}, S_{[j,x_4]}, S_{[x_1,k]}, S_{[l,x_2]}, S\backslash, T)/Z' \end{aligned}$$

where

$$\begin{aligned} S\backslash &= \{S_{[j',i']} | (j',i') \notin \{(x_1,i),(x_3,i),(j,x_2),(j,x_4),(x_3,k),(x_1,k),(l,x_4),(l,x_2)\} \text{ and} \\ Z' &= P(S_{[x_1,i]}, S_{[j,x_2]}, S_{[x_3,k]}, S_{[l,x_4]}, S\backslash, T) + P(S_{[x_3,i]}, S_{[j,x_4]}, S_{[x_1,k]}, S_{[l,x_2]}, S\backslash, T) \end{aligned}$$

As with RETRANS, we can factor out constant terms. These are word penalty and translation model scores for all phrase pairs and distortion and language model scores for all alignment blocks that are held constant. For each of the two alignment possibilities, the conditional probabilities reduce to calculating 4 distortion scores and 2 language model scores. Note however that if the alignments are adjacent on both source and target side and translated monotonically with respect to each other, then only 3 distortion scores need to be computed.

Example We illustrate this operator using the example in Figure 5.4 in which the sampler considers reordering the alignments at source spans $[2,3]$ and $[3,4]$. There are 2 possible outcomes to each reorder operation : (a) maintain the current alignment or (b) swap the alignment (since doing so does not violate reordering constraints).

The blocks being sampled from are $S_{[-,2]}$, $S_{[-,3]}$, $S_{[3,-]}$ and $S_{[4,-]}$. The monotone alignment is represented by $S_{[2,2]}$, $S_{[3,3]}$ (duplicated as the phrases are adjacent on the target side) and $S_{[4,\langle\backslash s\rangle]}$ where $\langle\backslash s\rangle$ denotes the end of sentence marker . By definition $S_{4,\langle\backslash s\rangle}$ has a score of 0 so we eliminate the term from our calculations. Also, we remove the duplicate alignment variable leaving us with $S_{[2,2]}$ and $S_{[3,3]}$

The swapped alignment is represented by $S_{[4,2]}$, $S_{[2,3]}$, $S_{[3,\langle\backslash s\rangle]}$ and $S_{[4,2]}$. Removing the duplicate variable and the variable involving $\langle\backslash s\rangle$ leaves us with $S_{[4,2]}$ and $S_{[2,3]}$.

Assuming:

$$S\backslash = \{S_{[j',i']} | \left[j',i'\right] \not\in \{[2,2],[2,3],[3,3],[4,2]\}\}$$

Setting all feature weights to 1, the monotone alignment $(S_{[2,2]}, S_{[3,3]})$ is chosen with probability:

$$\begin{aligned} p(S_{[2,2]}, S_{[3,3]} | S\backslash, T) &= \frac{p(S_{[2,2]}, S_{[3,3]}, S\backslash, T)}{p(S_{[2,2]}, S_{[3,3]}, S\backslash, T) + p(S_{[4,2]}, S_{[2,3]}, S\backslash, T)} \\ &= \frac{\exp\left[h_D(2,2) + h_D(3,3) + h_L(\text{“result remarkable”}, \text{“a”}, \langle\backslash \text{s}\rangle)\right]}{Z'} \end{aligned}$$

and the reordered alignment $(S_{[4,2]}, S_{[2,3]})$ chosen with probability:

$$\begin{aligned} p(S_{[4,2]}, S_{[2,3]} | S\backslash, T) &= \frac{p(S_{[4,2]}, S_{[2,3]}, S\backslash, T)}{p(S_{[2,2]}, S_{[3,3]}, S\backslash, T) + p(S_{[4,2]}, S_{[2,3]}, S\backslash, T)} \\ &= \frac{\exp\left[h_D(4,2) + h_D(2,3) + h_L(\text{“remarkable result”}, \text{“a”}, \langle\backslash \text{s}\rangle)\right]}{Z'} \\ \text{where } Z' &= \left(\exp\left[h_D(2,2) + h_D(3,3) + h_L(\text{“result remarkable”}, \text{“a”}, \langle\backslash \text{s}\rangle)\right]\right) \\ &+ \left(\exp\left[h_D(4,2) + h_D(2,3) + h_L(\text{“remarkable result”}, \text{“a”}, \langle\backslash \text{s}\rangle)\right]\right) \end{aligned}$$

5.3.3 MERGE-SPLIT

The first 2 operators considered so far keep the number of source side segments and therefore the number of active phrase-pairs/alignments in the model constant. The MERGE-SPLIT operator, on the other hand, looks to increase this number (by performing a split operation) or decrease this number (by merging) or keep it constant.

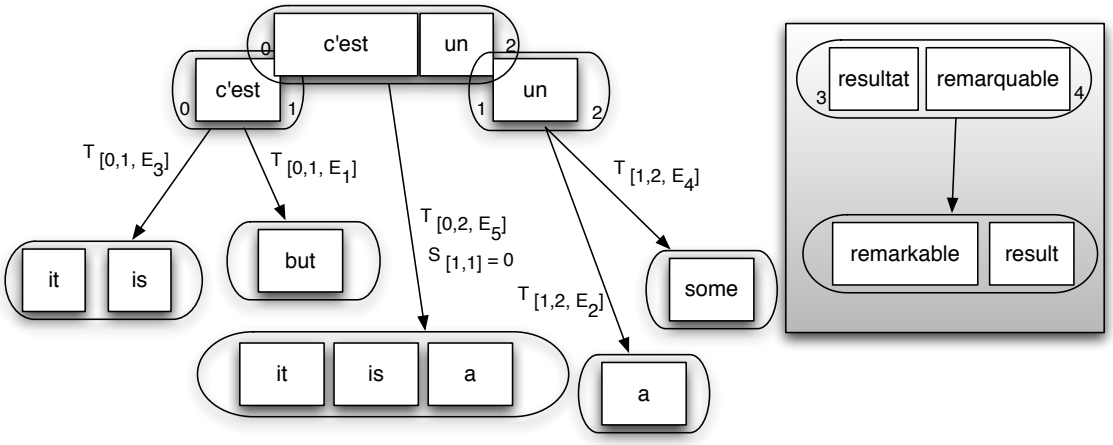


Figure 5.5: Example of a MERGE involving source spans [0,1], [0,2] and [1,2]. The operator considers translating the source span [0,2] using one phrase-pair or by maintaining the current segmentations. Here, E_1 = "but", E_2 = "a", E_3 = "it is", E_4 = "some" and E_5 = "it is a". Merging span [0,2] by activating $T_{[0,2,E_5]}$ requires setting off the alignment variable $S_{[1,1]}$. The shaded box covers variables that stay constant during the sampling step.

MERGE: Given a position j such that $[i,j]$ and $[j,k]$ are active spans, the MERGE operator samples from all the possible ways of translating the span $[i,k]$. The latter can be translated either by maintaining the current segmentations or by merging the segmentations in to one span. Reordering is not allowed during this sampling operation.

The operator first considers all the possibilities of translating $[i,k]$ using the variables in the blocks $T_{[i,j]}$ and $T_{[j,k]}$. Additionally, if existing spans $[i,j]$ and $[j,k]$ are currently being translated monotonically with respect to each other and if their translations are adjacent on the target side, i.e. $S_{[j,j]} = 1$, then the operator also considers variables from the block $T_{[i,k]}$. The operator then samples a new configuration for the variables.

If the operator chooses to merge the segmentations, it has to:

1. activate the new segmentation $[i,k]$ by activating one variable from the $T_{[i,k]}$ block.

2. inactivate the segmentations $[i,j]$ and $[j,k]$ by turning off all variables in $T_{[i,j]}$ and $T_{[j,k]}$ and by setting $S_{[j,j]}$ to 0.

The case where the operator chooses to maintain the current segmentations is equivalent to performing the RETRANS operator on each of the blocks $T_{[i,j]}$ and $T_{[j,k]}$.

Figure 5.5 illustrates the MERGE operator. The span [0,2] can be either translated by sampling from the block $T_{[0,2]} = \{T_{[0,2,E_5]}\}$ or by maintaining the current segmentations and sampling from blocks $T_{[0,1]} = \{T_{[0,1,E_1]}, T_{[0,1,E_3]}\}$ and $T_{[1,2]} = \{T_{[1,2,E_2]}, T_{[1,2,E_4]}\}$. In the latter case, the operator considers the set of variables formed by a cartesian product over the two blocks. In total, the operator considers 5 possible phrase-pair variable assignment configurations.

SPLIT: The split operator is the converse of the MERGE operator. Given a position j ($i < j < k$) such that the block $T_{[i,k]}$ has an active phrase-pair variable, the split operator samples from the phrase-pair blocks $T_{[i,j]}$, $T_{[j,k]}$ and $T_{[i,k]}$. Reordering is not allowed during this sampling operation.

If the operator decides to split the current segmentation, then it has to:

1. activate one variable from each of the $T_{[i,j]}$ and $T_{[j,k]}$ blocks and turn off all variables in the $T_{[i,k]}$ block.

2. set the value of the alignment variable $S_{[j,j]}$ to 1.

In case the operator decides against splitting, it samples a new phrase-pair assignment from the block $T_{[i,k]}$ (this is equivalent to a RETRANS operation).

The MERGE-SPLIT operator can therefore be seen as trying to translate a source span $[i,k]$ either with one phrase-pair or with two source adjacent phrase-pairs while leaving distortions constant. Conditional probabilities are derived in a manner similar to those for RETRANS.

5.3.4 Discussion

The use of Gibbs operators for SMT is first introduced in DeNero et al. (2008) for the task of directly inducing a phrase-based translation model from parallel corpora without detouring via word alignments. Their model is discussed in more detail in Section 3.8.

Our sampler is most similar to the decoders of (Germann et al., 2001; Langlais et al., 2007) which start with an approximate solution and then incrementally improve it via operators such as RETRANS and MERGE-SPLIT. It is also similar to the estimator of Marcu and Wong (2002), who employ the same operators to search the alignment

space from a heuristic initialisation. Although the operators are similar, the use is different. These previous efforts employed their operators in a greedy hill-climbing search. In contrast, our operators are applied probabilistically, making them theoretically well-founded for a variety of inference problems.

5.4 Sampler Correctness

We now show that the proposed Gibbs sampler for phrase-based translation models is a valid one, i.e. the sampler converges to the distribution of interest. As we saw in Section 3.4, Gibbs sampling, by construction, satisfies detailed balance therefore satisfying the requirement which states that the target distribution has to be invariant with respect to the Markov chain of the sampler. In this section, we examine the second requirement: that of the sampler's ergodicity.

Ergodicity means that the sampler will converge to the desired distribution irrespective of its initialisation point. To show that this is the case for our sampler, we will need to reuse some of the notation encountered in Section 3.2. There, we mentioned that a sampler is ergodic as long as its transition kernel K satisfies the following two conditions:

1. Irreducibility: There is a positive probability of visiting all other states starting from a given state of the Markov chain, i.e. the transition graph must be connected.

2. Aperiodicity: The chain should not get trapped in cycles, since otherwise it might never settle to an invariant distribution.

We also noted that the transition kernel K is usually constructed by the concatenation of simpler transition operators O and that these base operators do not individually have to be ergodic.

We therefore begin by defining the transition matrix K of our sampler. In our case, the base operators O are the Gibbs operators described earlier. It is clear that none of these base operators are individually ergodic since they each only sample from a subset of the whole distribution.

The base operators can be combined through successive application, such that

$$K(\mathbf{x}' \leftarrow \mathbf{x}) = \sum_{\mathbf{x}''} \sum_{\mathbf{x}'''} O_3(\mathbf{x}' \leftarrow \mathbf{x}'') O_2(\mathbf{x}'' \leftarrow \mathbf{x}''') O_1(\mathbf{x}''' \leftarrow \mathbf{x}) \tag{5.11}$$

where for clarity of exposition, we denote the sampling operators MERGE-SPLIT, RETRANS, and REORDER as O_1, O_2 and O_3 respectively and collapse the variables T and S referring to them collectively as $\mathbf{x}$.

Irreducibility

In our model irreducibility implies that, starting from a given derivation, all other derivations are reachable in a finite state of steps. To prove that our model is indeed irreducible, we begin by considering the simpler case where the source-side phrasal segmentation *is fixed*. In this case, K is a transition kernel created by applying the RETRANS operator repeatedly to each source position followed by applying the REORDER operator repeatedly to each pair of source positions. This means we need to show that for a given source-side segmentation held fixed, K can explore all the possible derivations allowed under the model, irrespective of the initial derivation.

It should be clear that this is the case. The RETRANS operator assigns a positive probability for sampling all phrase-pairs in the phrase table, so given a source phrase, all the target phrases it can translate to have a probability of being sampled. Similarly, the REORDER operator assigns a non-zero probability to each of the two reordering configurations possible (swap and monotone) with the operator being applied to every pair of source positions which do not violate the hard reordering limit constraint of the model. Thus, any derivation allowable under the model *can be reached* from any other derivation in a finite number of steps.

We now need to show that the MERGE-SPLIT operator is able to explore all source side segmentations. Let us recall what this operator does: when in split mode, it takes a phrase as input and segments it into two sub-phrases, provided the sub-phrases each have at least one entry in the phrase table. Conversely, when in merge mode, it takes two adjacent phrases and merges them together to form a larger phrase, provided the larger phrase has at least one entry in the phrase table.

Figure 5.6 and 5.7 depict the space of segmentations for the source sentence *"un resultat remarquable"* and *"un resultat tres remarquable"* under the MERGE-SPLIT operator respectively. Each word in the sentence is annotated with position indices as superscripts. The figures show that by applying a series of merge and split operations at appropriate word indices, the whole space of segmentations can be explored.

So, is the sampler irreducible? It turns out that there are some pathological cases where the sampler can get stuck. Figure 5.9 illustrates such an example. Here, there are

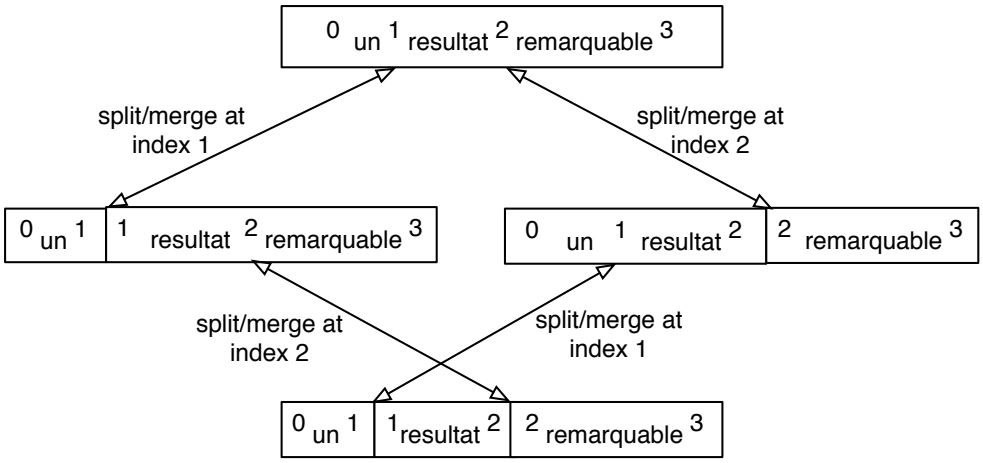


Figure 5.6: Space of source side segmentations for an example 3-word source phrase.

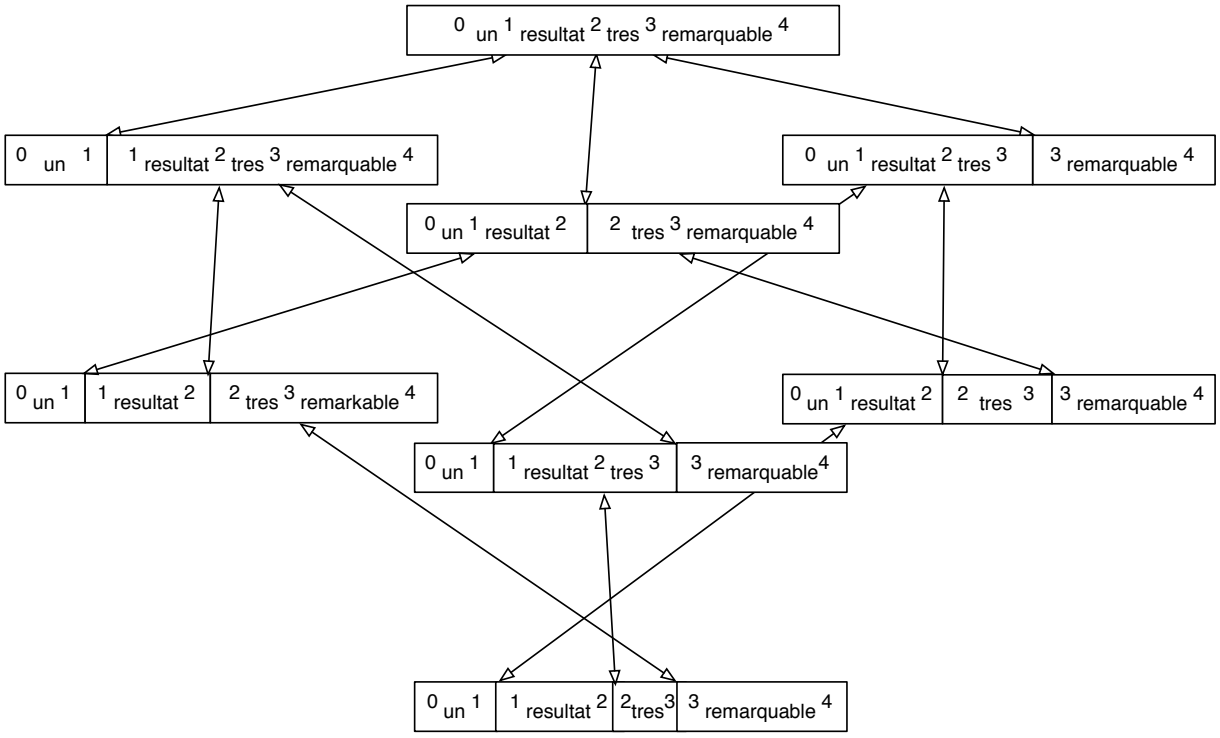


Figure 5.7: Space of source side segmentations for an example 4-word source phrase.

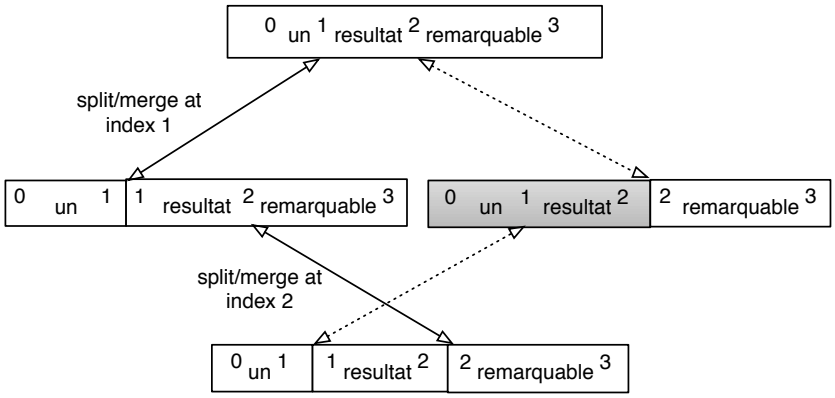


Figure 5.8: Space of source side segmentations for an example 3-word source phrase. The phrase indicated by shading does not have any entries in the phrase table.

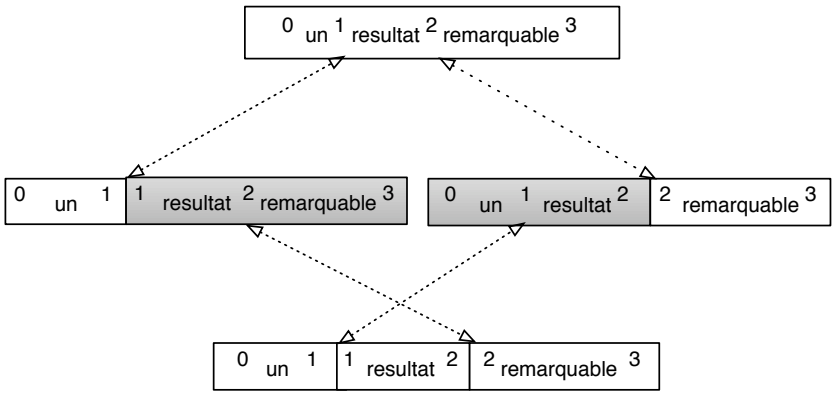


Figure 5.9: Space of source side segmentations for an example 3-word source phrase. The phrases indicated by shading do not have any entries in the phrase table. There is no sequence of split/merge moves to go from the configuration at the top to that in the bottom and vice-versa.

phrase table entries for translating the three-word segment and for translating each of the one-word segments but no entry for either of the two-word segments indicated by shading. This means that if the sampler is initially in the three-word segment configuration, there is *no* sequence of MERGE-SPLIT moves that will take it to the one-word segment configuration and vice versa.

The space of phrasal segmentations can be interpreted as a graph. The MERGE-SPLIT operator gets stuck when this graph gets *disconnected*. Taking the SPLIT operator as example (the argument for the merge operator is similar but in the reverse direction), we observe that for a source phrase of length n ($n > 1$), there exist n-1 positions where the operator can be applied. These positions are denoted as split points. Each split operation applied at such a point j produces a pair of substrings - a prefix substring spanning 0 to j and a suffix substring spanning j to n. A segmentation graph is connected if *at least one* out of the $n-1$ pair of substrings produced is a pair in which both prefix and suffix substrings have phrase table entries. The graph is disconnected otherwise.

This is illustrated by the example in Figure 5.8 where the segmentation graph is connected despite the phrase pair "*resultat remarquable*" not having a phrase-pair entry since phrase entries exist for the prefix and suffix phrases created by splitting "*un resultat remarquable*" at position 1. On the other hand, Figure 5.9 is a disconnected graph.

A *sufficient* condition for ensuring that every phrasal segmentation graph is connected is that the phrase table is well behaved. We define a *well-behaved* phrase table as one where for every source phrase of length n ($n > 1$), all $n(n+1)/2-1$ substrings of the source phrase have entries in the phrase table. An *ill-behaved* phrase table is one which is not well behaved.

Are there any *disconnected sentences*, i.e. sentences having disconnected segmentation graphs given the provided phrase table, in practice? A scan of our corpora and phrase tables shows that such cases exist but that they are very rare. We present the statistics in Tables 5.1 and 5.2.

Disconnected graphs arise as a result of the heuristics used during the phrase extraction step where, sometimes, due to noisy alignments, the necessary phrase-pairs fail to get extracted. They also occur when the significance score filtering technique used to reduce the size of the phrase table (described in Section 4.1) prunes away some necessary phrase-pairs.

Corpus	Sentences Total	Non-Ergodic Sentences
Tuning - MT02	1,043	75
Held-out - MT03	663	11
Test - MT05	1,056	8

Table 5.1: Number of sentences per corpus for Arabic-English for which the sampler is non-ergodic.

Corpus	Sentences Total	Non-Ergodic Sentences	
		French-English	German-English
Tuning - DEV2006	2,000	6	3
Held-out - TEST2007A	1,000	2	5
Test - TEST2008A	1,000	2	2
Test - NEWSDEV2009B	1,026	5	0

Table 5.2: Number of sentences per corpus for French-English and German-English for which the sampler is non-ergodic.

Since this limitation of the sampler was discovered recently, we did not manage to address it during the course of the thesis. As we shall see in Chapter 6 and 7, in spite of this limitation, the sampler's empirical performance remains competitive when compared against a dynamic programming based inference algorithm which can handle this problem. We believe that this is because of the limited number of sentences affected by the issue.

We now discuss some possible solutions for making the sampler irreducible irrespective of the vagaries of the phrase table. One unprincipled solution would be to simply manually fix the phrase tables by introducing the required phrase-pairs. Another solution would be to overload the MERGE-SPLIT operator so as to perform 3-way splits/merges instead of the current pairwise operations. The resulting operator would be able to handle the scenario in Figure 5.9, albeit at the cost of increased complexity. However, it is easy to can construct scenarios in which even this 3-way operator would get stuck. For example, imagine the case of a 4-word source phrase for which the phrase table additionally contains only its unigrams. The 3-way operator will be unable to explore the space of segmentations.

At this point, it is clear that we need a different solution to the problem. A principled solution is based around the observation that a dynamic programming algorithm such as stack decoding does not run into the problem just described. Therefore, we should aim to combine sampling and dynamic programming. One way to do so would be to use to occasionally apply a Metropolis-Hastings step after applying the MERGE-SPLIT operator. This Metropolis-Hastings step would propose a new derivation sampled from a stack decoder's translation lattice. This new derivation is accepted based on the standard Metropolis Hastings criterion. The appeal of this solution is that by maintaining an MCMC approach, we retain the theoretical guarantees provided by Monte Carlo sampling methods.

We conclude this discussion by stating that given a *well-behaved* phrase table, the sampler is indeed irreducible. Therefore, another solution would be to ensure well-behaved phrase tables. This can be done by moving away from heuristics-based induction of a phrase table to more principled methods.

Aperiodicity

Aperiodicity is fulfilled for any irreducible transition matrix K with $p(\mathbf{x},\mathbf{x}) > 0$ for some $\mathbf{x}$. This is clearly the case in our sampler, thus the sampler is aperiodic.

Conclusion

We have shown that under certain conditions, the transition matrix, K is ergodic and that the required distribution $P(T,S)$ is invariant with respect to the Markov chain. Therefore the samples drawn from the proposed Gibbs sampler are guaranteed to be generated from the distribution of interest.

5.5 Sampling Algorithm

Starting with an initial derivation, a complete *iteration* or *scan* of the sampler consists of applying each operator at each possible point in the sentence. By collecting a sample only at the end of a scan, we ensure that the correlation between successive samples is minimal.

Algorithm 5.2 presents the pseudocode of the *Scan* function, the function which performs one scan of the sampler. The function takes in as argument an initial derivation and a temperature t. When $t = 1$, the algorithm generates samples from the true

distribution. By changing the value of t, the shape of the distribution is altered. When t is set to a high value, the distribution is flattened whereas when t is close to 0, the distribution is peaked with most of the probability mass concentrated around the mode of the distribution. The technique of varying the temperature is called *annealing* and was presented in Section 3.6.

Algorithm 5.1 Pseudocode of the *Scan* function which performs a scan of the sampler

```
Input: current sample s, temperature t
Global variables: length of source sentence m, reordering limit r.
for i = 0 to m − 1 do
    s = MERGE-SPLIT(s, i, t)
end for
for i = 0 to m − 1 do
    for j = i + 1 to m − 1 do
        if j < i + r then // Reordering limit check
            s = REORDER(s, i, j, t)
        end if
    end for
    for j = 0 to i − 1 do
        if j + r > i then // Reordering limit check
            s = REORDER(s, i, j, t)
        end if
    end for
end for
for i = 0 to m − 1 do
    s = RETRANS(s, i, t)
end for
return s // Return sample
```

In Section 3.5, we mentioned that getting a sampler to effectively explore the required distribution, also known as getting the sampler to *mix*, requires careful choices in the way the sampler is run. For example, a burn-in period might be required so as to overcome the effects of the way the sampler is initialised. A further consideration is the number of chains of the sampler one wishes to run. One can choose to run one long chain or a few chains of medium length or multiple short chains.

Algorithm 5.2 presents the sampling algorithm which we employ. The algorithm is parameterised by :

- C: the number of chains used,
- B: the number of samples to discard during burn-in,
- $Anneal_A$: the annealing function to be used (if any),
- *Init*: the method to initialise the sampler,
- N: number of samples per chain. We also refer to N as the number of *iterations* the sampler needs to be run for.

The sampling procedure is initialised with a derivation returned by *Init()*. We then run the sampler for C chains. Each chain begins with a period of burn-in where generated samples are discarded. By applying annealing during the burn-in phase, we can speed up the rate at which the sampler "forgets" its initialisation. The annealing function decreases linearly from a starting temperature $t = A$ to $t = 1$ at a rate equal to $(A-1)/B$. After the burn-in phase, the sampler stochastically generates new derivations by sampling from the true distribution. These sampled derivations are added to the sample set S.

In preliminary experiments, we found that running two chains ($C = 2$) of medium sized lengths was the most judicious use of our computing resources. Also, by setting B to 100 and starting the annealing schedule at $A = 3$, the sampler was able to rapidly move away from its initialisation point. We use these settings for all further experiments in this thesis.

The remaining parameters of the algorithm are a) N, the number of samples per chain and b) the way the sampler is initialised. We considered two possible ways of initialising the sampler:

1. *full*. This consists of using the max derivation solution of a phrase-based stack decoder such as Moses with a model using all features and standard reordering limit. Initialising the sampler with the *full* solution means that we start the sampler from close to the mode, if not the mode, of the distribution.

2. *random*. In this method, we run a phrase-based stack decoder with all feature weights set to 0 and choose a random final solution. We chose to use *random* initialisation so as to assess whether the sampler can converge to the stationary distribution *irrespective* of where in the state space it is started from.

Algorithm 5.2 Complete sampling algorithm

Inputs:
C: number of chains of the sampler,
B: number of burn-in iterations,
$Anneal_A$: an annealing function with initial temperature = A,
N: number of samples per chain,
Init: an initialisation function returning a sample.
Output: S, a set containing samples s drawn from the distribution
s = *Init()*
for $c = 1$ **to** C **do**
  for $b = 1$ **to** B **do**
    $t = Anneal_A(b, B)$
    s = *Scan*(s, t) // Do not collect sample during burn-in
  end for
  for $n = 1$ **to** N **do**
    s = *Scan*($s, 1$)
    Add s to S // Collect sample
  end for
end for
return S

We examine the effects of these parameters in Section 7.1.1.

5.6 Sampling Complexity and Speed

Complexity

The REORDER operator iterates over the positions in the input and considers reordering the target phrase of the current source phrase with the target phrase of all other source phrases, *provided* that the reordering limit is not violated. This means that it can only consider swaps within a fixed-length window where the size of the window is Λ, the reordering limit. Each application of the operator necessitates scoring of *two* different configurations. The total complexity of the operator is $O(m\Lambda)$ where m is the length of the input sentence.

In addition to m, the complexity of the RETRANS and MERGE-SPLIT operators also depends on p, the average number of phrase-table entries per source phrase. As indicated in Section 4.1, p is capped to a maximum of 20 in the experiments in this thesis.

The RETRANS operator is applied at each position i in the source sentence. Each operation involves scoring the p phrase-pairs for the source phrase at position i. The complexity of this operator is therefore $O(mp)$.

The MERGE-SPLIT operator is similarly applied at each position i in the source sentence. As shown in the example in Figure 5.5, the operator considers all possible ways of translating a source segment using one single source phrase and using two source phrases. In the former case, a maximum of p phrase-pairs need to be scored while the latter requires scoring p^2 different translation configurations. The complexity of MERGE-SPLIT is thus $O(mp^2)$ and therefore dominates the total complexity of a scan of the sampler.

Speed

We also ran experiments to determine the speed of the sampler. The experiments were performed for the French to English translation task using a phrase table where p, the average number of phrase-table entries per source phrase, was equal to 8.68.

We used the sampling algorithm and default parameters detailed in Algorithm 5.2, initialising the sampler in *full* mode. We ran the sampler on a randomly selected 36 word long sentence from the TEST2008 dataset several times, each time varying N, the

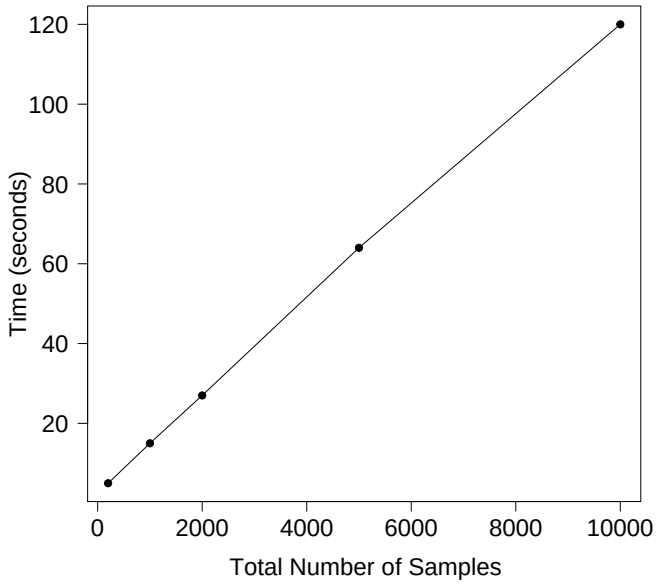


Figure 5.10: Sampling time in seconds as a function of total number of samples collected. The statistics are averaged over two sampler runs.

number of samples per chain. For each value of N, we ran the sampler twice in order to take into account any variations in the compute load on the machine used for running the experiments and report the average result. Note that since we run the sampler for 2 chains, the total number of samples is equal to $2N$. Experiments are run on an Intel Xeon CPU with a processor speed of 2.67GHz.

We present the results of our timing experiments in Figure 5.10. We find that the sampling time is proportional to the number of samples collected. Sampling 10,000 derivations takes approximately 120 seconds.

5.7 Sampler Convergence

In Section 5.4, we proved formally that under certain conditions our sampler will converge to the desired distribution. We now investigate how the sampler behaves experimentally. We are interested in verifying that, as the theory indicates, the chain does indeed converge to the desired distribution in the limit of the sample set size. Moreover, we are also interested in knowing the rate of convergence of the sampler.

A slow convergence rate would indicate that the sampler is *poorly mixing* - it stays in small regions of the parameter space for long periods of time. Our computational resources are finite so we would like the sampler to be *well mixing*, that is, to explore the entire space. Since Gibbs sampling, apart from the annealing temperature, is a hyper-parameter free algorithm, the only control we have for optimising the mixing rate of the chain is via the design of efficient sampling operators. This experiment therefore also serves to ascertain whether our operators have been sensibly designed and whether their concatenation produces an appropriate transition matrix.

To monitor the convergence of the sampler when translating a sentence, we would like to compare the distribution estimated by the sampler with the true distribution. We are interested in two distributions: the derivation level distribution $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ and the distribution over translations, $p(\mathbf{e}|\mathbf{f})$.

Both distributions are estimated using the Monte Carlo estimate given in Equation 5.2. The estimated derivation level distribution is obtained using the indicator function in (5.3). We denote this distribution by $\tilde{p}(\mathbf{e},\mathbf{d}|\mathbf{f})$. The estimated translation level distribution, $\tilde{p}(\mathbf{e}|\mathbf{f})$, is obtained using the indicator function in (5.4).

The most commonly used similarity measure for comparing probability distributions is the *Kullback-Leibler (KL) divergence*. The KL divergence between two probability distributions $P(x)$ and $Q(x)$ is:

$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{5.12}$$

Typically P represents the true distribution and Q is an approximation of P. It is a non-negative number where a high value indicates that P and Q are far apart and a low value indicates that P and Q are close. The KL divergence is equal to 0 only if $P = Q$.

The KL divergence is a non-symmetric measure ($D_{KL}(P||Q) \neq D_{KL}(Q||P)$) only defined when $P(x) > 0$ and $Q(x) > 0$ for all x and when P and Q both sum to 1. It also requires that both distributions have the *same support* that is they are both defined over the same elements.

In our case, the support of the estimated distribution consists of all the derivations or translations in the sample set whereas the support of the true distribution is the space of all derivations or all translations that can be produced by the model. For all but the shortest of input sentences, the latter space is orders of magnitude larger than any set of samples we can reasonably be expected to generate. This means that we *cannot* use KL divergence as a similarity measure to assess the sampler's performance.

Instead, we propose using a KL-like similarity measure, which we term *sampler divergence*. This divergence is given by:

$$D_{Sampler}(Q||P) = \sum_{x \in \mathcal{S}} Q(x) \log \frac{Q(x)}{P(x)} \tag{5.13}$$

This measure is almost similar to $D_{KL}(Q||P)$ except that the summation is performed over items in the sample set $\mathcal{S}$ rather than the entire support. Over the sample set, $Q(x)$ is a well-formed probability distribution summing up to 1 thus satisfying one requirement of a KL divergence measure. However, the measure *is not* a KL divergence because $P(x)$ does not sum up to 1 over the support. Nevertheless, we found the sampler divergence to be a good and useful measure for monitoring the sampler's performance.

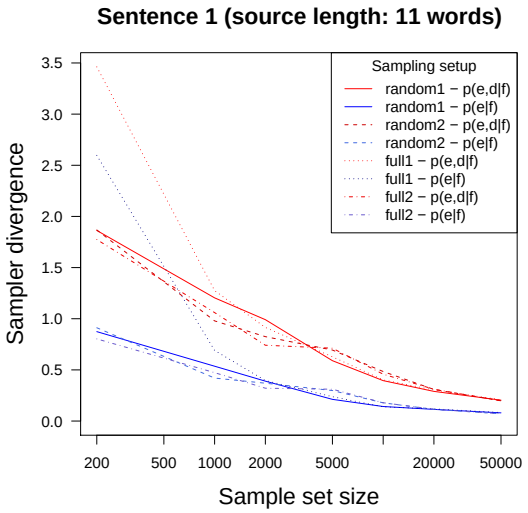
By setting $P(x)$ to $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ and $Q(x)$ to $\tilde{p}(\mathbf{e},\mathbf{d}|\mathbf{f})$ and summing over the samples we can compute the sampler divergence between the two derivation level distributions. We can do likewise for the translation distributions. Note that we require the true probability only for those derivations or translations in the sample set S.

How can we compute the true value of $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ for a given derivation? Recall that the whole purpose of using sampling is because this distribution and the corresponding translation distribution cannot be computed tractably. However, for short enough sentences, we can use a beam decoder to compute an exhaustive translation lattice, i.e. one without pruning. We then apply the *forward algorithm* on the lattice to compute Z. The latter is used to convert the score $s(\mathbf{e},\mathbf{d},\mathbf{f})$ of a derivation to a probability:

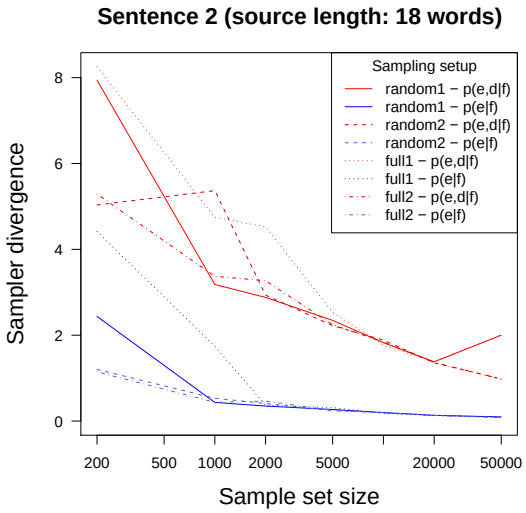
$$p(\mathbf{e},\mathbf{d}|\mathbf{f}) = \frac{\exp\left[s(\mathbf{e},\mathbf{d},\mathbf{f})\right]}{Z} \tag{5.14}$$

To compute the probability of a translation, we *intersect* each translation in the sample set S with the lattice and normalise this translation score by Z.

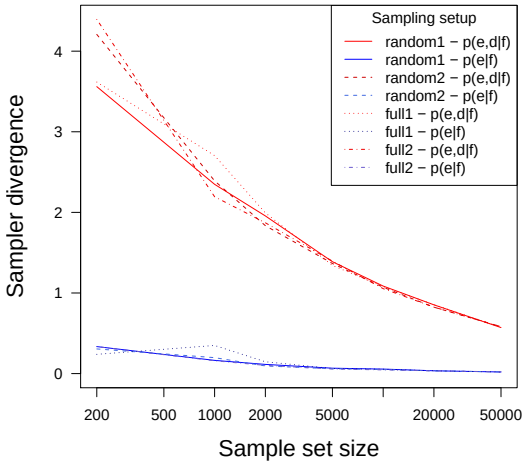
Since we keep the exhaustive translation lattice is memory and the lattice can be very large, we only apply this algorithm to sentences shorter than 20 words. The complexity of this algorithm is additionally increased by the very time-consuming operation of intersecting the lattice with every translation in the sample set.


Sentence 1 (source length: 11 words)
Sampler divergence
Sample set size
Sampling setup
random1 – p(e,d|f)
random1 – p(e|f)
random2 – p(e,d|f)
random2 – p(e|f)
full1 – p(e,d|f)
full1 – p(e|f)
full2 – p(e,d|f)
full2 – p(e|f)

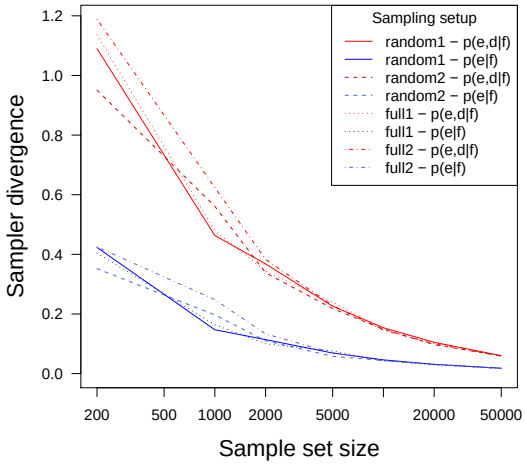
(a)


Sentence 2 (source length: 18 words)
Sampler divergence
Sample set size
Sampling setup
random1 – p(e,d|f)
random1 – p(e|f)
random2 – p(e,d|f)
random2 – p(e|f)
full1 – p(e,d|f)
full1 – p(e|f)
full2 – p(e,d|f)
full2 – p(e|f)

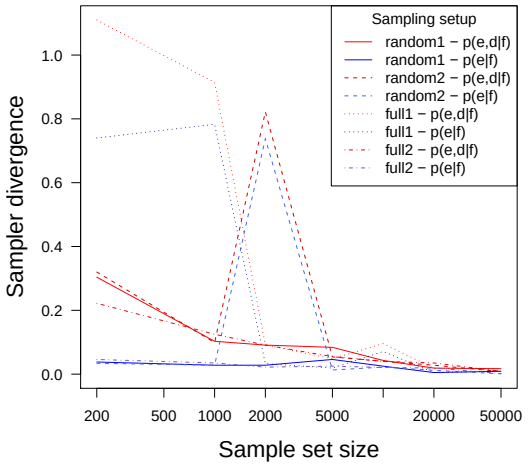
(b)


Sentence 3 (source length: 19 words)
Sampler divergence
Sample set size
Sampling setup
random1 – p(e,d|f)
random1 – p(e|f)
random2 – p(e,d|f)
random2 – p(e|f)
full1 – p(e,d|f)
full1 – p(e|f)
full2 – p(e,d|f)
full2 – p(e|f)

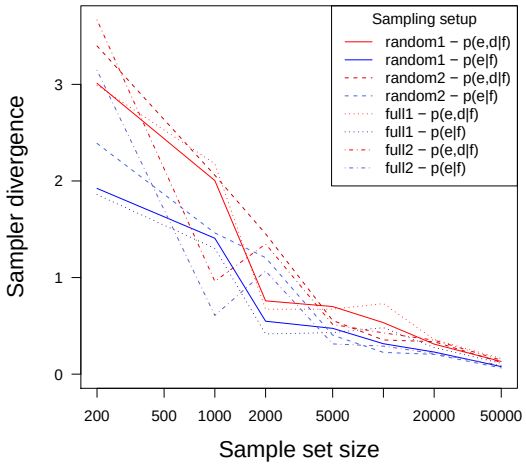
(c)


Sentence 4 (source length: 11 words)
Sampler divergence
Sample set size
Sampling setup
random1 – p(e,d|f)
random1 – p(e|f)
random2 – p(e,d|f)
random2 – p(e|f)
full1 – p(e,d|f)
full1 – p(e|f)
full2 – p(e,d|f)
full2 – p(e|f)

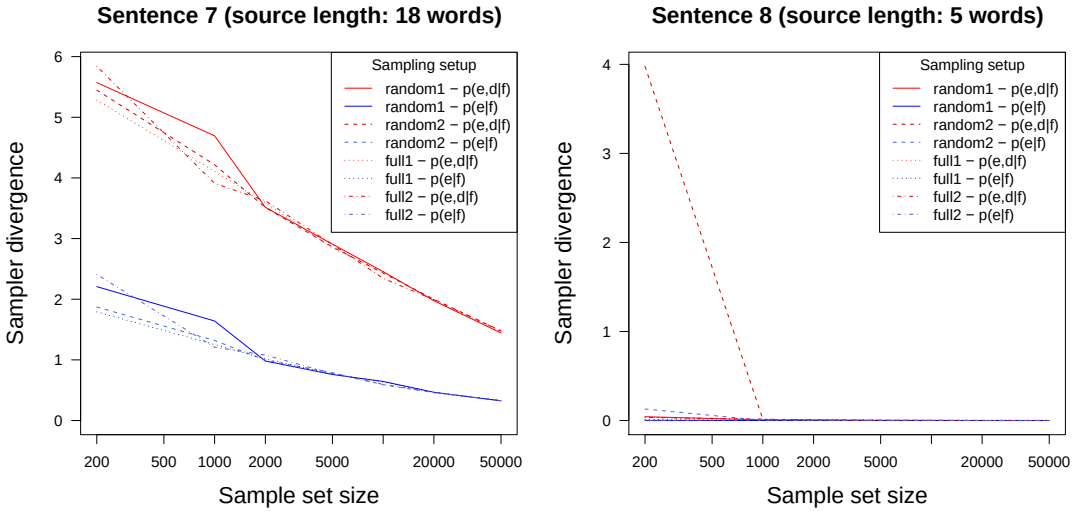
(d)


Sentence 5 (source length: 11 words)
Sampler divergence
Sample set size
Sampling setup
random1 – p(e,d|f)
random1 – p(e|f)
random2 – p(e,d|f)
random2 – p(e|f)
full1 – p(e,d|f)
full1 – p(e|f)
full2 – p(e,d|f)
full2 – p(e|f)

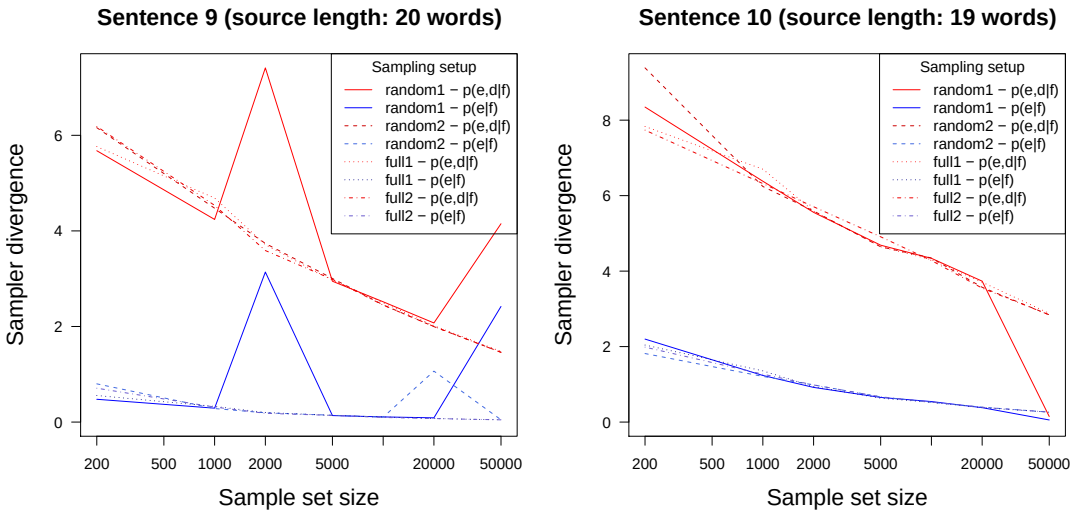
(e)


Sentence 6 (source length: 10 words)
Sampler divergence
Sample set size
Sampling setup
random1 – p(e,d|f)
random1 – p(e|f)
random2 – p(e,d|f)
random2 – p(e|f)
full1 – p(e,d|f)
full1 – p(e|f)
full2 – p(e,d|f)
full2 – p(e|f)

(f)



(g) (h)



(i) (j)

Figure 5.10: Sampler divergence between sampler estimated distribution and true distribution as a function of the number of samples collected for French to English data set. Lines in red correspond to divergences between derivation level distributions. Lines in blue correspond to divergences between translation level distributions. We compare random initialisation and full initialisation running the sampler twice for each condition.

Experiments

We ran experiments to monitor the sampler divergence between the estimated and the true distributions as a function of the sample set size. We compared results obtained using the full initialisation and the random initialisation described in Section 5.5. We ran these experiments on the French to English translation task using 10 randomly selected sentences of lengths shorter than 20 words from the TEST2008A data sets. Since the sampler is stochastic, results vary across runs. To observe and to account for this variance, we run the sampler twice in each configuration reporting both sets of results. We use the sampling algorithm presented in Section 5.5 with the burn-in, number of chains and annealing parameters set to the values described there.

Figure 5.10 presents the results. The lines in red indicate sampler divergences for distributions over *derivations* whereas the lines in blue indicate divergences between estimated and true *translation* distributions. First of all, we observe that at all sample set sizes the translation divergences are lower than the derivation ones. This is because the support of the translation distribution is smaller than that of the derivation distribution, i.e. there are far fewer distinct translations than there are derivations. Therefore, it is easier to get a good estimate of the former rather than the latter. Secondly, we note that in general, as the sampler set size increases, the sampler divergence decreases. This behaviour is consistent with the theory behind Monte Carlo sampling.

In most cases, a very accurate estimation of the translation distribution can be obtained from just 2,000 samples. However estimating the derivation distribution to the same level of accuracy requires more samples, typically, at least 10,000 samples are required. With a sample set size of 50,000, the divergence for most sentences is almost zero. This confirms that the Gibbs sampler described in this chapter *does* in fact sample from the translation model probability distribution and is thus an appropriate tool for performing inference tasks in such distributions.

There are a number of other interesting things to note in Figures 5.10. For instance, we find that the method of initialising the sampler *does not* have an appreciable effect on the sampler's performance except with low number of samples. This lack of dependency on the starting point indicates that the sampler *mixes well*: starting at the mode (*full* initialisation), the sampler does occasionally make stochastic jumps to lower probability regions. Whereas when started at a random point in the state space (*random* initialisation), the sampler is able to quickly move to the high probability regions of the distribution.

We have pointed out before that because the Gibbs sampler *stochastically samples* from the conditional distributions of variables in the model, it is very likely that results across different runs of the sampler give differing results. From Figure 5.10, we observe that this variance is in most cases quite low, especially when the sample set is large. For example, for sentences 1, 3, 4, 5, 6, 7 and 8, the results for derivation divergence across 4 runs (2 types of initialisation x 2 different runs) are almost indistinguishable from each other; likewise when comparing translation divergences.

Occasionally, however, there are considerable variations in sampler divergences. We take as example Sentence 2 when run in full initialisation mode for 50,000 samples. Here, the sampler divergence between the derivation distributions is 0.98 in the first run and is 2.01 in the second. The increase in divergence in the second run indicates that in this run the sampler is overestimating the probability of one or more derivations. This is confirmed when looking at the samples generated. The most probable derivation as per the sampler has a derivation probability of 0.00162 whereas its true probability is 3.15327e-05. In fact, 9 out of the top 10 most probable derivations as per the sampler are derivations whose true probability are 2 orders of magnitude lower.

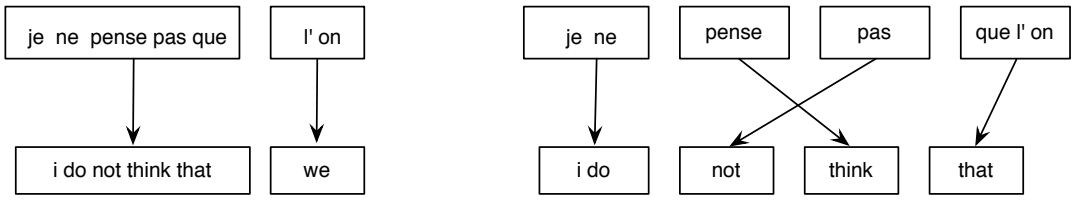


Figure 5.11: Examples of two derivations of the same source sentence. The derivation on the left is a high probability derivation under the model whereas the one on the right has low probability.

Figure 5.11 illustrates the situation discussed. The derivation on the left of the figure corresponds to the top ranked derivation in the first run of the sampler, one where the divergence is low. This derivation translates the source segment by splitting it in two and subsequently translating the resulting two segments in the order that they appear in the source. This derivation is therefore *monotone*. The derivation on the right depicts the top ranked derivation in the second sampler run, a run where the divergence is high. Here, the source segment is split in four segments and there is a reordering performed to translate the target word *"not"* before the word *"think"*. To go from the low probability derivation on the right to the high probability derivation to the left

using the sampler operators necessitates first reordering the second and third source phrases after which a series of split/merge operations can be applied. However, the reordering operation produces the target phrase *"think not"* which given the contexts on the left and on the right is an unlikely string with low language model score so that the monotone translation has low probability.

The derivation on the right is an example of a point in the state space trapped in a local optima where the only way to move to a region of higher probability is to go through a region of low probability. Note that since the sampler makes moves stochastically, it will eventually decide to make this move were we to run it for long enough. This is the reason why we see in Figure 5.10 that at low sample set sizes, there tends be high variance between divergences but that these variations reduce when the sampler is run for longer.

Note also the contrast between the sampler and a greedy search algorithm which is based on similar sequences of local changes. Since greedy search always makes moves to configurations which increase the probability for the derivation, once it reaches a local optima, it can never escape it.

5.8 Summary

In this chapter, we have described a novel alternative to dynamic programming based approximate inference for phrase-based SMT. Our proposed approach uses Gibbs sampling to explore the probability distribution of phrase-based models. We formally showed that under certain conditions on the phrase table the sampler is correct and therefore guaranteed to converge to the true distribution. We then ascertained this fact empirically showing that the sampler will converge to the desired distribution irrespective of where it is initialised from, thus also demonstrating the mobility of the sampler in the search space.

Chapter 6

Sampling for Decoding

Having ascertained in Chapter 5 that the sampler can be used for exploring the probability distribution of a standard phrase-based model, we now turn to the task of decoding with the sampler. Decoding amounts to finding the translation $\mathbf{e}^*$ that maximises or minimises some criterion given a source sentence $\mathbf{f}$ as input. This criterion is referred to as the *decision rule*. In this chapter, we investigate using the sampler for decoding with two common decision rules: Maximum A Posteriori (MAP) decoding and minimum Bayes risk (MBR) decoding.

6.1 Maximum A Posteriori Decoding

The Maximum A Posteriori (MAP) decision rule consists of finding the *mode* of the distribution $p(\mathbf{e}|\mathbf{f})$. Since in most SMT systems, the probability distribution is defined over derivations in the model, the MAP decision rule is of the form:

$$\mathbf{e}^*_{\text{MAP}} = \arg\max_{\mathbf{e} \in T(\mathbf{f})} \sum_{\mathbf{d} \in D(\mathbf{e},\mathbf{f})} p(\mathbf{e}, \mathbf{d}|\mathbf{f}) \tag{6.1}$$

For every translation string $\mathbf{e}$ the model can produce, Equation 6.1 requires summing the scores of all the derivations that yield $\mathbf{e}$. Since the sizes of both $D(\mathbf{e},\mathbf{f})$, the set of derivations yielding $\mathbf{e}$, and $T(\mathbf{f})$, the set of translation strings which the model can produce, may be exponential in the length of the input string, the maximisation in (6.1) turns out to be NP-hard.

The decision rule in (6.1) is often referred to as *max translation* decoding. A common approximation to the latter is *max derivation* decoding. Here, the `sum` operation

in (6.1) is replaced by a less expensive `max` operation, to give the following decision rule:

$$\mathbf{e}^*_{\text{MAP}} = Y\left(\arg\max_{\mathbf{d}\in D(\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f})\right) \tag{6.2}$$

The max derivation approximation to MAP decoding consists of finding the most probable derivation rather than the most probable translation: the probability of a string is approximated by the probability of its most likely derivation. While simpler than the decision rule in (6.1), computing the max derivation solution remains intractable, thus requiring approximate inference methods.

As described in Section 5.1, the Gibbs sampler can be used to efficiently generate sample derivations from $p(\mathbf{e},\mathbf{d}|\mathbf{f})$,the probability distribution over derivations. These samples can then be used to obtain an estimate, $\tilde{p}(\mathbf{e},\mathbf{d}|\mathbf{f})$, of this distribution. The maximum of the estimated distribution is the most likely derivation, that is, the max derivation solution.

Similarly, we can marginalise (sum) over the samples yielding the same translation string to obtain $\tilde{p}(\mathbf{e}|\mathbf{f})$, an estimate of the probability distribution over translations. The maximum of this estimated distribution is the max translation solution.

The Gibbs sampler for phrase-based translation can therefore also be used as a decoder for both MAP decision rules.

6.1.1 Related Work

6.1.1.1 Max Derivation Decoding

Various methods for computing the max derivation solution have been proposed in the SMT literature. Some of these methods are discussed in Section 2.4.1. The most popular method is beam decoding (**?**Koehn et al., 2003; Chiang, 2005), a heuristic-based search algorithm which uses dynamic programming methods for efficient decoding.

There are two main differences between using beam search and using sampling for max derivation decoding. Firstly, the former's use of dynamic programming to perform polynomial time decoding restricts the model to the use of local or near-local features. In our sampling-based method, this restriction is lifted: since we sample whole derivations, *any* function of $h(\mathbf{e},\mathbf{d},\mathbf{f})$ may participate in the translation model subject only to its own computability.

To explain the second difference, we begin by noting that the decision rule in (6.2) is equivalent to the following decision rule:

$$\mathbf{e}^*_{\text{MAP}} = Y\left(\arg\max_{\mathbf{d}\in D(\mathbf{f})} s(\mathbf{e},\mathbf{d},\mathbf{f})\right) \tag{6.3}$$

where $s(\mathbf{e},\mathbf{d},\mathbf{f}) = \boldsymbol{\lambda}\cdot h(\mathbf{e},\mathbf{d},\mathbf{f})$ is the *unnormalised* score of a derivation given features $h(\mathbf{e},\mathbf{d},\mathbf{f})$ and weights $\boldsymbol{\lambda}$. The decision rules in (6.2) and (6.3) are equivalent because the normalisation term $Z(\mathbf{f})$) is constant for all derivations in the model and thus can be dropped from the decision rule formulation.

The beam search algorithm uses the decision rule in (6.3), that is, it finds the derivation with the highest score s. The sampler, on the other hand, first estimates a probability distribution $\tilde{p}(\mathbf{e},\mathbf{d}|\mathbf{f})$ from the generated samples and then picks its mode as the solution.

Assuming the sampler is drawing samples from the correct distribution and that the translation model only uses local or near-local features, in the sampling limit, the solutions of both decoders will be identical (we assume that there are no search errors in the beam decoder). However, the beam decoder will be much more efficient than the sampler. This is because unless most of the probability mass is centered in the vicinity of the mode, samples will only rarely be drawn from around the mode, and the sampler will waste time exploring areas of the state space of no interest.

A more efficient way of using the sampler for max derivation decoding would be to use it as a *stochastic search* procedure. This can be done using *annealing*. Instead of sampling from the true distribution $p(\mathbf{e},\mathbf{d}|\mathbf{f})$, annealing at iteration i samples from a distribution $p_i(\mathbf{e},\mathbf{d}|\mathbf{f}) \propto p^{1/T_i}(\mathbf{e},\mathbf{d}|\mathbf{f})$ where T_i is a cooling schedule. At early iterations, the temperature is set to a high value which effectively smooths the distribution and allows the Markov chain high mobility in the state space. At $T = 1$, annealing samples from the true distribution. As the temperature is progressively cooled to approach 0, the probability mass concentrates around the mode. While annealing gives no guarantee of finding the global optimum of the distribution, it has proved useful in many applications (Finkel et al., 2005; Goldwater et al., 2006; Goldwater and Griffiths, 2007).

While we are curious to know how well the sampler does at max derivation decoding, its main purpose in this thesis is to allow us to test the hypothesis that for phrase-based SMT, decoding algorithms, such as max translation decoding, which model the translation task as a direct mapping from source string $\mathbf{f}$ to target string $\mathbf{e}$ by

marginalising over the space of derivations $D(\mathbf{e}, \mathbf{f})$ can lead to improved translation performance. Since performing max translation decoding with the sampler requires generating samples drawn from the base distribution $p(\mathbf{e}, \mathbf{d}|\mathbf{f})$, we eschew the use of annealing.

6.1.1.2 Max Translation Decoding

Beam decoding is an efficient approximate search algorithm for computing the max derivation solution; however, there is no comparable tractable dynamic programming based algorithm for decoding in the marginalised distribution $p(\mathbf{e}|\mathbf{f})$.

A beam search algorithm for the task is presented by Blunsom et al. (2008). In their method, instead of storing just the last n-1 generated target words which are required for language model score computation, each hypothesis stores the entire target string generated so far. If two partial hypotheses have translated the same source words and have produced the same target string, their scores are combined and only one of the hypotheses is retained for further expansion.

Blunsom et al.'s algorithm is simple and requires only a slight modification of the standard algorithm used for max derivation decoding. However, it does not scale well. This is because the dynamic program used for recombination is more involved resulting in fewer hypothesis recombinations. Consequently, there is an explosion in the number of hypotheses to be expanded, slowing down decoding dramatically. Therefore, Blunsom et al. resort to aggressive pruning and only use their algorithm to decode sentences fewer than 10 words long.

A more efficient algorithm is the variational decoding method proposed by Li et al. (2009b) which is able to scale to long sentences. Variational decoding is an instantiation of a general class of approximate inference algorithms known as variational inference in which the original intractable distribution of interest p is approximated by a simpler distribution q which supports exact inference. The variational distribution used by Li et al. is a distribution over n-grams given their history.

Variational decoding consists of first decoding the input sentence with a max derivation beam decoder, then computing the variational distribution over the resulting search hypergraph. The hyperedges of the hypergraph are finally rescored using the variational distribution. The yield of the rescored best scoring derivation is the max translation solution. Variational decoding is fast and is able to exploit information from the entire pruned search space. Additionally, Li et al. find that variational

decoding gives significant improvements over max derivation decoding on a large scale translation task.

The limitations of variational decoding are that the variational distribution used can be a poor approximation of the true distribution. In fact, Li et al. have to compensate for such cases by interpolating the variational approximation of distributions over n-grams with the derivation level distribution $p(\mathbf{e}, \mathbf{d}|\mathbf{f})$. Further details on existing max translation decoding algorithms are given in Section 2.4.2.

Like variational decoding, the sampling-based approach proposed here offers a tractable solution to max translation decoding. The sampling method offers the theoretical guarantee that in the limit, the estimated distribution will match the true distribution. In Section 7.1.1, we provided empirical evidence which suggests that this is the case even with a finite sample set. Using the sampler for max translation will therefore help us determine whether this decision rule offers any benefit over the usual max derivation decision rule.

6.1.2 Decoding Parameters

We have shown that the Gibbs sampler can be used to perform max derivation and max translation decoding. Gibbs sampling is an MCMC algorithm with the convenient property that since it samples from a series of exact conditional distributions, it is largely a parameter-free algorithm[1]. Yet, from a practical point of view, there are a number of hyper-parameters that need to be optimised before we can use the sampler to perform the task we are interested in which, in this chapter, is decoding.

In Section 7.1.1, we investigated the impact of the sampler's initialisation and of the number of samples collected on how well sampler estimated distributions match the true distributions. We found that the initialisation point did not matter and that as the sample set size grows, the sampler's estimates become more accurate.

In this section, we investigate whether the same results hold when performing decoding. Our experiments in this section are conducted on the French to English and the German to English translation tasks using the experimental setup described in Chapter 4.

[1] There is one parameter in Gibbs sampling: the annealing temperature.

6.1.2.1 Scaling Factor

In order to be run as a decoder, the sampler first needs to be specified weights λ for the features in the model. For the experiments reported in this section, we used feature weights trained with minimum error rate training (MERT; Och, 2003).

As discussed in Section 2.5.3, the objective function during MERT training is:

$$\begin{aligned} \lambda_{\text{MERT}} &= \arg\min_{\lambda} \sum_{c=1\ldots C} \text{Loss}(\arg\max_{\mathbf{d}\in D(\mathbf{f}_c)} s(\mathbf{e},\mathbf{d},\mathbf{f}_c;\lambda),\mathbf{e}_c) \\ &= \arg\min_{\lambda} \sum_{c=1\ldots C} \text{Loss}(\arg\max_{\mathbf{d}\in D(\mathbf{f}_c)} \lambda\cdot h(\mathbf{e},\mathbf{d},\mathbf{f}_c),\mathbf{e}_c) \end{aligned} \tag{6.4}$$

where $\text{Loss}(\mathbf{e},\mathbf{e}_c)$ quantifies the error in hypothesising translation $\mathbf{e}$ when the reference translation is $\mathbf{e}_c$.

The objective function above minimises the error rate under an unnormalised *linear* model, i.e. one which ignores the normalising term $Z(\mathbf{f}_c)$. Notice that the arg max inside (6.4) is invariant with respect to the scale of the weight vector λ; scaling the individual components of λ k times will still return the same arg max solution as when using λ itself. As a result, the MERT objective function is invariant to the weight vector scaling; the Moses implementation simply normalises the weight vector it finds by its ℓ_1-norm.

However, when we use these weights in a true probabilistic model, the scaling factor α affects the behaviour of the model since it determines how peaked or flat the distribution is, as can be seen from the following equation:

$$p(\mathbf{e},\mathbf{d}|\mathbf{f};\lambda,\alpha) = \frac{\exp\left[\alpha\cdot\lambda\cdot h(\mathbf{e},\mathbf{d},\mathbf{f})\right]}{\sum_{d'\in D(\mathbf{f})}\exp\left[\alpha\cdot\lambda\cdot h(Y(\mathbf{d}'),\mathbf{d}',\mathbf{f})\right]} \tag{6.5}$$

We optimised the scaling factor for each language pair using the first 200 sentences of the DEV2006 tuning set. We ran the sampler collecting 10,000 samples and used the *full* initialisation. Table 6.1 shows the effects of the scaling factor on the BLEU score of the sampler running in max derivation (MaxD) and max translation (MaxT) modes for French-English and German-English.

We find that the scaling factor can bring about substantial differences in performance. When $\alpha = 1$, the BLEU score for French-English and German-English max derivation is 2 and 3.5 respectively. By increasing the scaling factor, performance goes up dramatically reaching a peak of 33.3 BLEU on max translation for French-English when $\alpha = 10$ and a peak of 26.0 BLEU for German-English max translation when

Scale/Dataset	1	2	3	4	5	6	8	10	15	20
FR-EN MaxD	2.0	18.5	28.8	31.0	32.6	32.5	**33.0**	**33.0**	32.9	32.7
FR-EN MaxT	2.8	21.2	30.3	31.7	32.3	32.5	32.9	**33.3**	32.9	33.0
DE-EN MaxD	3.5	11.6	20.1	24.3	25.0	**25.5**	25.0	25.1	24.7	24.8
DE-EN MaxT	3.1	11.9	21.8	24.6	25.4	**26.0**	25.3	24.9	24.8	24.7

Table 6.1: Effects of scaling factor on the BLEU score of the sampler running in max-derivation (MaxD) and max-translation (MaxT) modes on 200 sentences of French-English (FR-EN) and German-English (DE-EN) tuning sets. Translation performance measured in BLEU. Best performances for each translation condition are highlighted in bold.

$\alpha = 6$. We use these values of α for the experiments in this chapter. Increasing α further leads to a drop in performance.

Why is the performance so poor at low values of α and why is there a drop in performance at high values of α? This is because when α is too small, the distribution is too flat and the sampler spends too much time exploring unimportant probability regions. When it is too large, e.g when α = 20, the distribution is too peaked and the sampler may concentrate on a very narrow probability region.

We can quantify this argument by computing the average *derivational entropy* and the average *translation entropy* of the sample sets for the French-English dataset as α is varied. The derivational entropy, $H_{\mathbf{d}}(p)$ of a sample set $\mathcal{S}$ is given by:

$$H_{\mathbf{d}}(p) = -\sum_{\mathbf{d}\in\mathcal{S}} p(\mathbf{e},\mathbf{d}|\mathbf{f}) \log p(\mathbf{e},\mathbf{d}|\mathbf{f}) \tag{6.6}$$

where $\mathbf{e}$ is the yield of $\mathbf{d}$ and $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ is the sampler estimated probability of derivation $\mathbf{d}$. The average derivational entropy is computed by averaging $H_{\mathbf{d}}(p)$ over the data set.

The translation entropy is computed similarly but using $p(\mathbf{e}|\mathbf{f})$, the sampler estimated distribution over translations, and summing over the translations in the sample set $\mathcal{S}$:

$$H_{\mathbf{e}}(p) = -\sum_{\mathbf{e}\in\mathcal{S}} p(\mathbf{e}|\mathbf{f}) \log p(\mathbf{e}|\mathbf{f}) \tag{6.7}$$

Figure 6.1 shows how both entropies vary with the scaling factor. At $\alpha = 1$, both estimated distributions are at maximum entropy (indicated by the dotted horizontal line

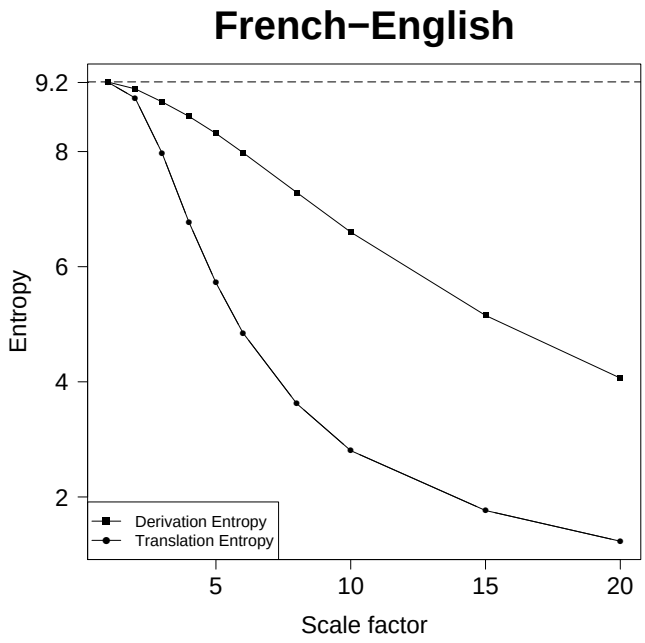


Figure 6.1: Entropy of derivation and translation distributions estimated by sampler as a function of scaling factor. 10,000 derivations are sampled for each of 200 French-English sentences drawn from the tuning set.

in the graph). Any derivation or translation seen more than once is returned as the mode of their respective distributions. In the maximum entropy case, no one derivation or translation is sampled more than once, in which case the sampler arbitrarily breaks the tie. As the scaling factor is increased, the distributions sharpen. As a result, the likely derivations in the model get sampled often, so the entropy decreases and translation performance goes up. At $\alpha = 20$, the translation entropy is especially low. However, the translation performance at that scaling is worse than when $\alpha = 10$, indicating that the sampler is concentration on a too narrow region of the distribution and missing out on regions where good translations are to be found.

As we saw in Section 7.1.1, the translation entropy tends be lower than the derivation entropy. This is because the distribution over translations is the marginal of the distribution over derivations so therefore the partition inequalities apply.

6.1.2.2 Sampling Initialisations and Iterations

In the next set of experiments, we examine the effects of different initialisation strategies and the sample set size on the max derivation decoding performance of the sampler using the tuned scaled factors.

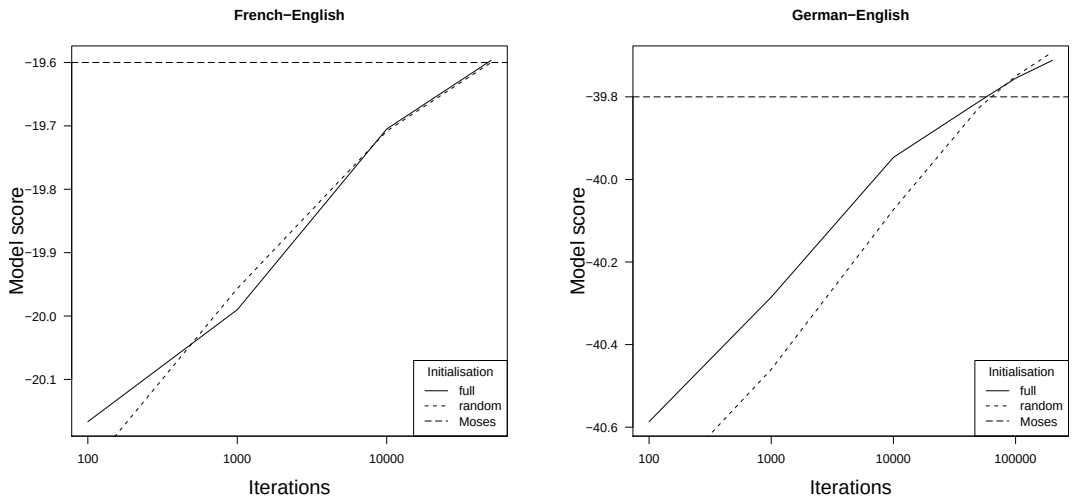


Figure 6.2: Mean maximum model score, as a function of iteration number and starting point. The starting point can either be the full max derivation solution (**full**), or a random derivation (**random**).

Since the features and weights stay constant during the experiments, we can compare decoding performance by looking at the model scores of the solutions found by the sampler in each run. We use as baseline the model scores of the max derivation solutions found by Moses.

We compared the two initialisation strategies from Section 5.5: *full* and *random* on the first 200-sentences portion of the DEV2006 tuning set. We varied the number of samples from 100 to 50,000 in the case of French-English and 100 to 200,000 in the case of German-English. We were unable to run the experiments for more sampling iterations due to computational constraints.

From Figure 6.2 we can see that for French-English, the starting point did not have an appreciable effect on the model score of the best derivation, except with low numbers of iterations. For German-English, *full* initialisation does better than *random* until we reach a high number of samples at which point it looks as if their respective performances are about to converge. This confirms the findings in Section 7.1.1 that the sampler mixes well.

Comparing the best model scores found by the sampler with those found by the Moses decoder with its default settings, we found that around 50,000 samples were required for French-English and 100,000 for German-English for the sampler to give equivalent model scores to Moses. Running the sampler for 100,000 iterations took on average 1552 seconds per sentence on German-English, meaning that the sampler

needs to be run for almost 3 orders of magnitude longer than Moses for obtaining the same model score.

As discussed in Section 6.1.1.1, this way of computing the max derivation solution is far too inefficient to be practical. However, it serves as proof of concept to show that given enough sample derivations, the sampler will find the mode of the distribution. One interesting fact to note is that the sampler finds solutions with better model scores than Moses when run for 200,000 iterations in the German-English translation task. The solutions found by the sampler are derivations which were pruned away during beam search in Moses.

The performance of the sampler running as a max derivation decoder is properly measured by comparing the model scores of its solutions with those of the solutions found by Moses. However, even if the sampler is making search errors, it is possible that it is finding solutions that are good enough translations of the source sentence. It will therefore be instructive to see how good these solutions are. In Figure 6.3, we plot how the BLEU scores of the sampler max derivation and max translation solutions vary as a function of sampling iterations for models run with *full* initialisation. We use the same 200 sentence portion of the German-English and French-English DEV2006 tuning set as in the previous experiment.

For both language pairs, once a large enough number of samples is collected, there is a steady improvement in max derivation BLEU score as the number of samples increases. The trend for max translation decoding performance is less clear: increasing the number of samples does in general lead to increased translation performance but there are exceptions.

In the case of French-English translation, sampler performance for both decision rules is still far from Moses performance even with 50,000 samples. Conversely, at the same point in German-English, both sampler decoding methods have overtaken Moses max derivation. Generating and storing 50,000 samples, however, is too computationally demanding. For the remaining decoding experiments in this thesis, we use 10,000 samples which gives us a good trade-off between sampling speed and decoding performance.

6.1.2.3 Max Translation vs Max Derivation

In addition to showing us how scaling the MERT learnt weights affects translation performance, Table 6.1 also gives an indication as to how max translation decoding compares to max derivation. For both language pairs, best performance as measured

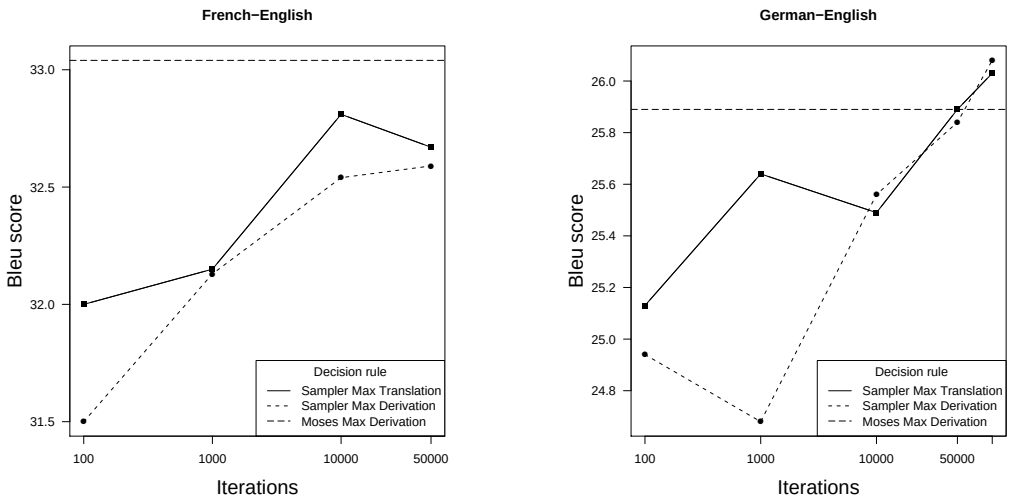


Figure 6.3: BLEU score as a function of sample set size.

by BLEU is obtained using max translation decoding thus showing the benefits of this decision rule. However, max translation decoding does not always outperform max derivation decoding. For example when the scaling factor is 5, max derivation does better than max translation decoding on French-English. This erratic behaviour is also seen in Figure 6.3.

However, we would be too hasty were we to draw conclusions about translation performances from any one run of the sampler. This is because the use of random numbers to generate samples introduces non-determinism in any task for which the sampler is used. When using Gibbs sampling for decoding, different runs of the sampler will give different results. To investigate the variation in translation performance, we run the sampler for 10,000 iterations on the same set of 200 sentences from the French-English DEV2006 dataset 10 different times using the tuned scale factor. This will also allow us to see whether max translation does in fact consistently outperform max translation decoding.

In Table 6.2, we report the best and the worst max derivation (MaxD) and max translation (MaxT) BLEU performance across the 10 runs along with the mean and the standard deviation (Std Dev) BLEU. We compared running the sampler with *full* and with *random* initialisation.

We see that the difference in BLEU between the best and the worst results across runs for the same decoding condition can be anything between 0.5 and 0.9 BLEU. We can also observe that the variance between results is slightly less for runs initialised in

Condition	Worst	Best	Mean	Std Dev
full - MaxD	32.4	**33.3**	32.7	0.25
full - MaxT	32.8	**33.3**	**33.1**	0.14
random - MaxD	*32.3*	33.0	32.7	0.27
random - MaxT	32.5	33.2	32.8	0.17

Table 6.2: Variation in max derivation (MaxD) and max translation (MaxT) BLEU scores when decoding a 200 sentence subset of the DEV2006 French-English dataset 10 times using *full* and *random* initialisation. Worst result is italic and best results are in bold.

the *full* mode compared to *random* initialisation and for max translation decoding runs compared to max derivation decoding.

When comparing the performance between the two decision rules, we see that with *full* initialisation, the mean max translation performance is 0.4 BLEU is better than mean max decoding result; the improvement when using *random* initialisation is a much smaller difference of 0.1 BLEU. Since these results are averaged across 10 different runs, we can say that max translation does in fact perform better that max derivation, although the improvements are at times marginal.

We also wanted to see how often the max translation solution differed from the max derivation solution. We find that when the sampler is run with *full* initialisation, the 2 solutions differed on average 24% of the time. In Table 6.3, we show 4 randomly chosen sentences for which the decision rules produced differing translations.

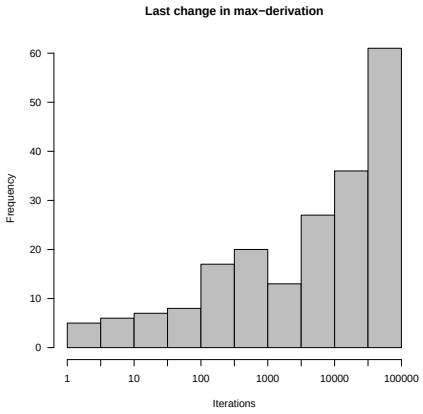
6.1.2.4 Further Analysis

We ran additional experiments on both French-English and German-English to monitor at what sampling iterations the max derivation and the max translation solutions settle on their final solutions. We also kept track on how often these solutions change through the sampling process. We used the same 200 sentences drawn from DEV2006 initialising them with the *full* Moses solution and running the sampler till 100,000 samples are collected.

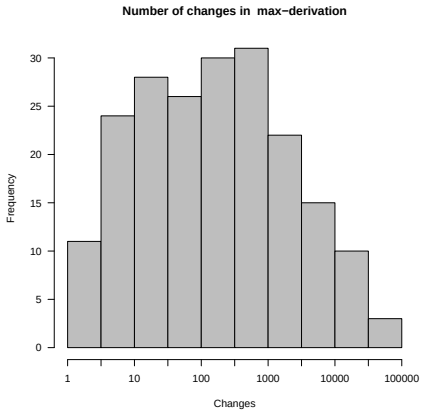
	Output
R	and i can tell you why ; because these people wanted to be on the right side once and for all , i.e. securely anchored in the family of democratic nations .
D	i can also give you the right : these *once and for all people wanted to* be on the right side , which means be firmly anchored in the family of democratic nations .
T	i can also give you the right : these *people wanted once and for all* be on the right side , which means be firmly anchored in the family of democratic nations .
R	that is also why it opposed those of the amendments tabled by our fellow members in order to demand planet-wide general disarmament , with the european union to set the example .
D	that is why , too , it will oppose those of the amendments tabled by our colleagues in calling for a general disarmament *across* the world *in* which the european union should set an example .
T	that is why , too , it will oppose those of the amendments tabled by our colleagues in calling for a general disarmament *throughout* the world , which the european union should set an example .
R	the provision according to which the notion of professional , essential and crucial requirement may justify exemptions on the grounds of religion is not , in my view , acceptable .
D	the provision under which the concept of professional requirement , *which is* essential and decisive , can justify derogations on the grounds of religion is , in my view , unacceptable .
T	the provision under which the concept of professional requirement , essential and decisive , can justify derogations on the grounds of religion is , in my view , unacceptable .
R	the civilisation we share asserts its greatness through respect for the rules of an open , tolerant and liberal society , with its inclusive and multicultural dynamics .
D	our common civilisation states its greatness *in* respect of the rules of the open society , tolerant and liberal , for its dynamic inclusive and multicultural .
T	our common civilisation states its greatness *by* respect for the rules of the open society , tolerant and liberal , for its dynamic inclusive and multicultural .

Table 6.3: Comparison of reference translation (R) and max derivation (D) and max translation (T) outputs on for 5 randomly chosen sentences on French-English translation task. Differences between max derivation and max derivation solutions are marked in italics.

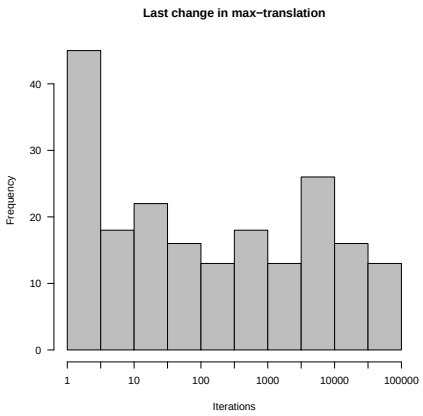
Figures 6.4 and 6.5 shows these statistics for French-English and German-English respectively. We see that the max derivation solution keeps on changing during sampling whereas the max translation solution settles early. This is consistent with our previous findings that since there are many more derivations than translations, it requires many more samples to get a good estimate of the distribution over derivations (and therefore a good estimate of its mode) than it does to do the same for the distribution over translations.



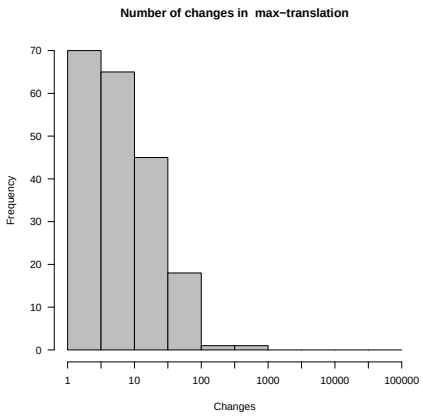
(a) Sampling iteration at which the max derivation last changes



(b) Number of times the max derivation solution changes during sampling

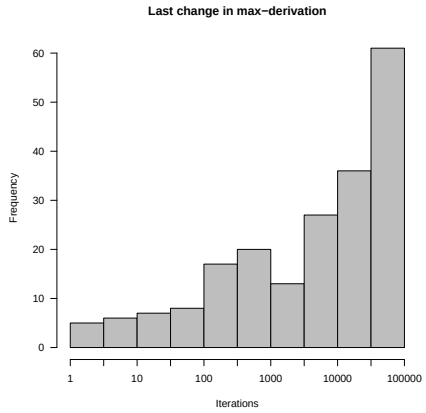


(c) Sampling iteration at which the max translation last changes

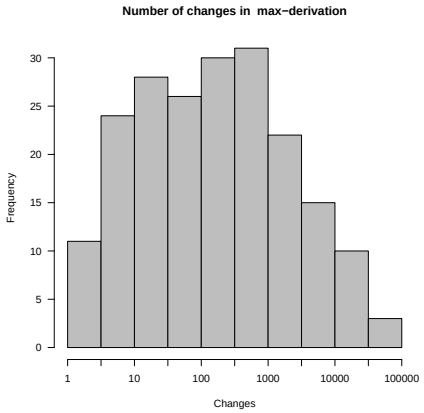


(d) Number of times the max translation solution changes during sampling

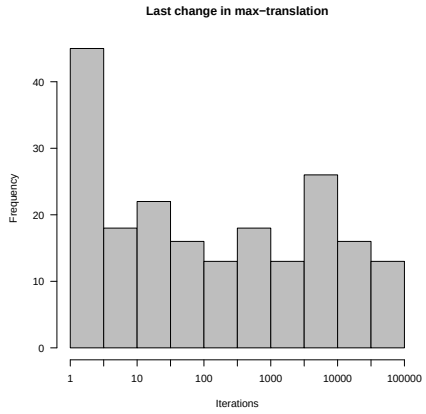
Figure 6.4: Changes in decoding solutions during French-English sampling experiments.



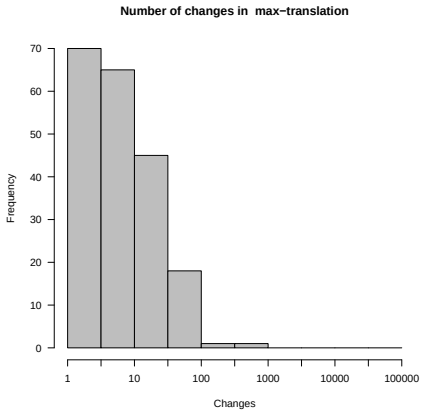
(a) Sampling iteration at which the max derivation last changes



(b) Number of times the max derivation solution changes during sampling



(c) Sampling iteration at which the max translation last changes



(d) Number of times the max translation solution changes during sampling

Figure 6.5: Changes in decoding solutions during German-English sampling experiments.

6.2 MBR Decoding

An alternative decision rule to MAP decoding is minimum Bayes risk (MBR) decoding. The MBR decision rule comes from statistical decision theory which states that the optimal decision rule for any statistical model is the solution which minimises its risk or expected loss. As applied to SMT systems, the decision rule is given by:

$$\begin{aligned}
\mathbf{e}^*_{\text{MBR}} &= \arg\min_{\mathbf{e}\in T(\mathbf{f})} R_{\mathbf{f}}(\mathbf{e}) \\
&= \arg\min_{\mathbf{e}\in T(\mathbf{f})} \sum_{\mathbf{e}'\in T(\mathbf{f})} \ell(\mathbf{e},\mathbf{e}')p(\mathbf{e}'|\mathbf{f}) \\
&= \arg\min_{\mathbf{e}\in \varepsilon_H} \sum_{\mathbf{e}'\in \varepsilon_E} \ell(\mathbf{e},\mathbf{e}')p(\mathbf{e}'|\mathbf{f}) \\
&= \arg\max_{\mathbf{e}\in \varepsilon_H} \sum_{\mathbf{e}'\in \varepsilon_E} \text{BLEU}(\mathbf{e},\mathbf{e}')p(\mathbf{e}'|\mathbf{f})
\end{aligned}$$

where $R_{\mathbf{f}}(\mathbf{e})$ represents the risk when translating $\mathbf{f}$ of choosing $\mathbf{e}$ and $\ell(\mathbf{e},\mathbf{e}')$ is the loss incurred when choosing solution $\mathbf{e}$ if the true solution is $\mathbf{e}'$. The arg min operation can be replaced by an arg max if we use a gain function such as BLEU instead of a loss function.

The space of translation candidates over which the risk is computed is usually referred to as the *evidence space* of the algorithm and denoted by ε_E. Similarly, the translation minimising the risk is chosen from a space of candidates, ε_H denoted as the *hypothesis space*.

Both $|\varepsilon_E|$ and $|\varepsilon_H|$ are exponential in the length of the input sentence, so computing the MBR solution exactly is intractable. However, since the MBR decision rule involves computing an *expectation* term for each translation in the hypothesis space, we can use the described sampler to compute these expectations while benefitting from the theoretical guarantees provided by Monte Carlo sampling. Once the expectations are computed, then the MBR solution is simply the translation which has the highest expected value.

In our implementation of sampler-based MBR decoding, the evidence space is set to be the *set of sampled translations*. For each translation $\mathbf{e}$ in the sampled set, we compute the Monte Carlo estimate of its posterior probability $p(\mathbf{e}|\mathbf{f})$ using the technique described in Section 5.1.

Tromble et al. (2008) show that while the MBR risk computation benefits from a large evidence space, MBR decoding performance is just as good, and the decoding much faster, if the hypothesis space is limited to the top n most probable translation candidates. We likewise limit the hypothesis space to the top n most probable translations in the sample set.

Since MBR decoding is performed at a sentence level whereas BLEU is a corpus level metric, we use SBLEU as the sentence-level approximation of BLEU.

6.2.1 Related Work

We discussed n-best MBR and lattice MBR, the two existing methods of performing MBR decoding in detail in Sections 2.3.2 and 2.4.3. Our sampler-based MBR decoder shares similarities with these methods but also differs in some fundamental aspects.

When run with MERT trained weights, all three algorithms require that the weights be scaled so as to obtain a probability distribution conducive to MBR decoding. This weight scaling is done similarly to the procedure described in Section 6.1.2.1 with the introduction of a scaling factor which is tuned on translation performance as measured by BLEU on a held-out set.

Existing methods mine the search graph of a first pass beam decoder. In n-best MBR, both the evidence and the hypothesis spaces are formed using an n-best list of distinct derivations extracted from this search graph. n-best MBR makes a max derivation approximation: the probability of a translation is approximated by the probability of its most likely derivation. Additionally, the normalisation term $Z(f)$ required to convert the score of a derivation into a probability is computed by summing up the scores of the derivations in the n-best list. Therefore, the estimate of $p(\mathbf{e}|\mathbf{f})$ of each translation string e in the n-best list is liable to be a crude approximation of the true probability.

Lattice MBR is able to leverage a much larger evidence space for the computation of the Bayes risk of each translation in the hypothesis space by using a dynamic programming algorithm over the entire search lattice. However, for efficiency purposes, this lattice, which is already a pruned representation of the entire search space, has to be further pruned prior to risk computation. The dynamic program used during lattice MBR requires a gain function which decomposes over the edges of the lattice. This precludes the use of SBLEU, so lattice MBR uses a linear approximation of log BLEU.

Both Tromble et al. (2008) and Kumar et al. (2009) find that lattice MBR (and its variant for use with grammar-based translation models, hypergraph MBR) significantly outperforms n-best MBR. Their experiments show that the improvement in performance comes from a more accurate estimation of risk brought about by using a much larger evidence space while as far as the hypothesis space is concerned, results using the entire hypergraph are as good as just using a 1000-best list.

Lattice MBR however introduces several parameters. Final decoding performance is very sensitive to the settings of these parameters so they need to be carefully tuned on a held-out set. One such parameter controls the amount of pruning which needs

to be performed on the lattice prior to risk computation. Another set of parameters determine how closely the linear approximation to log BLEU matches true BLEU. A final parameter balances the contribution of the lattice MBR solution and the max derivation solution. The latter interpolation is used to increase the robustness of the algorithm.

On the other hand, the sampler-based MBR decoder combines the simplicity of the *n*-best MBR decoder and its use of SBLEU, rather than an approximation thereof, while theoretically guaranteeing an accurate and unbiased risk assessment, without the introduction of any additional parameters in the algorithm.

6.3 Experiments

Having tuned the scale factor for each language pair and ascertained that *full* initialisation produced better translation results, we ran the sampler with these settings on our test sets. These are the in-domain TEST2008 test set and the out-of-domain newsdev2009b data set. Details about these datasets are given in Chapter 4.

We compare sampler-based max derivation, max translation and MBR decoding with Moses max derivation and MBR decoding. For the sampler experiments, we report mean results across 5 runs of the sampler, sampling for 10,000 iterations in each run.

For both sampler and Moses MBR decoding experiments, we use the same scaling factor (10 for French-English and 6 for German-English) as found in Section 6.1.2.1. Sampler MBR is performed using all 10,000 samples as evidence space and the top 1,000 most probable translations as hypothesis space; in Moses MBR, the hypothesis space is limited to the 1,000-best distinct derivations. In *n*-best MBR, the evidence space is equal to the hypothesis space whereas in lattice MBR, the evidence space consists of the search lattice pruned using Forward-Backward pruning (Sixtus and Ortmanns, 1999) such that the average number of edges per word in the resulting reduced lattice is equal to 50.

The translation results as evaluated by BLEU are shown in Table 6.4. Note that for all these experiments, the test set contains only 1 reference sentence for each input sentence. First we observe that when decoding with Moses the *n*-best MBR decision rule does at least as well as the max derivation baseline on 3 out of 4 datasets. The small gains obtained are consistent with results reported in the literature. When using

	fr-en		de-en	
	in	out	in	out
Moses MaxD	**33.5**	19.1	**27.8**	15.9
Moses N-MBR	33.4	**19.3**	**27.8**	16.1
Moses L-MBR	33.4	**19.4**	**27.8**	16.1
Sampler MaxD	33.2	18.9	27.0	15.3
Sampler MaxT	33.2	19.0	27.4	15.8
Sampler MBR	33.2	19.0	27.5	**16.3**

Table 6.4: Comparison of the BLEU score of the Moses decoder running in max derivation (MaxD), n-best MBR (N-MBR) and lattice MBR (L-MBR) modes with the sampler running in max derivation (MaxD), max translation (MaxT) and MBR modes. The test sets are TEST2008 (in) and NEWS-DEV2009B (out). Numbers in bold indicate the best results for each test set.

lattice MBR, we fail to get improvements over n-best MBR except a minor one in the case of the French-English out-of-domain data set.

In comparison to beam search, sampler results are, except in one test condition, systematically lower. As we explained in Section 6.1.1.1, we do not expect the sampler to perform as well as beam search as far as max derivation decoding is concerned. This is borne out by the results in Table 6.4 where sampler max derivation trails beam search max derivation by a margin of 0.2-0.8 BLEU. However, we do expect the sampler to do well on max translation and MBR decoding since both require estimating $p(\mathbf{e}|\mathbf{f})$, a probability distribution which, as we have seen in Section 7.1.1, the sampler is good at estimating. We find that this is the case, especially in German-English translation: max translation decoding does at least as good as max derivation decoding and likewise MBR decoding does at least as good as max translation decoding. In fact, in the case of out-of-domain German-English translation, the best performance is obtained using sampler MBR.

Nevertheless, the sampler generally fares worse that Moses. We hypothesise that this is because the weights used for these experiments were optimised by MERT for max derivation decoding whereas the decision rules we use at decoding time are sensitive to the translation model's entire predictive distribution. Therefore, the experiments do not rule out the possibility that max translation and MBR decoding

will offer an advantage on an appropriately trained model. We consider methods of training such a model in the next chapter.

6.4 Summary

In Chapter 5, we saw that our proposed Gibbs sampler for phrase-based translation is able to reliably estimate the posterior distribution over derivations and the posterior distribution over translations with only a finite number of samples. In this chapter, we have investigated the use of the sampler for the task of *decoding*. More precisely, we have described how the sampler can be used to perform approximate inference for two intractable optimisation tasks, namely Maximum A Posteriori (MAP) decoding and MBR decoding.

MAP decoding consists of finding the most probable translation in the model. Few efficient algorithms exist for implementing this decision rule or even for computing a close approximation of it; however, since such algorithms do exist for approximately computing the most probable derivation in the model, the MAP decision rule is usually approximated by the latter.

The sampler provides a tractable solution for computing both the most probable derivation translation and the most probable translation. When used to implement the two decision rules on test data, we found that the quality of the most probable translation is always at least as good as that of the most probable derivation, confirming recent findings in the SMT literature. However, we found that the sampler worked best when used as an MBR decoder.

Nevertheless, the sampler's empirical performance on these tasks trails behind heuristic based search algorithms devised for these decision rules. We conjecture that the sampler's poor performance is due to the use of weights which have been optimised to maximise 1-best performance, whereas the decision rules we would like to apply them on take into account the entire distribution. The challenge therefore is to tune the parameters of the translation model so that they exploit the predictive power of the complete distribution. We present methods to do so in the next chapter.

Chapter 7

Minimum Risk Training

In the previous chapter, we described the use of the sampler for providing approximate inference solutions when exact decoding is intractable. We found that the best translation performance was obtained when using the minimum Bayes risk (MBR) decision rule. The MBR decision rule comes from statistical decision theory which says that the optimal decision rule for any statistical model is the solution that *minimises its risk* or *expected loss*. Since machine translation models are typically evaluated by BLEU, a loss function which rewards partial matches, it is preferable to use the MBR solution rather than the typical maximum a posteriori (MAP) solution, which, in fact, is the MBR solution under the much harsher 0/1 loss function.

The MBR decision rule consists of calculating the expected loss (also known as the *risk*) of each translation candidate drawn from a hypothesis space and then returning the solution which minimises this risk. Since the decision rule involves calculating the *expectation* of a function defined over translations in the model, the MBR decision rule is ideally suited to be computed using sampling methods such as the Gibbs sampler introduced in this thesis. However, the empirical performance of sampling MBR lagged behind beam search implementations of the MAP and MBR decision rules.

We hypothesised that this unsatisfactory performance is due to the use of feature weights trained using the Minimum Error Rate Training (MERT) parameter estimation technique. The objective function used in this algorithm optimises the single best derivation in the model to the detriment of other derivations which might be of good quality too. An alternative objective which maximises the probability of a large number of good quality translations in the model might produce distributions more suited for use with the sampler. An additional drawback in using MERT optimised weights with

the sampler is that a time consuming grid-search is required to scale the weights such that they produce a distribution shaped appropriately for sampling.

Instead of MERT, we propose optimising the feature weights using *minimum risk training* (Smith and Eisner, 2006). In this chapter we show that minimum risk training, a parameter estimation technique which considers the entire distribution, is well suited for the sampler. It produces already scaled weights and experimental evidence suggests that these weights lead to better and more stable translation performance than when using MERT optimised weights.

7.1 Motivation

In this thesis, we propose optimising the weights of the features of our log-linear model using *minimum risk training*. This training regime aims to find weights that minimise the expected loss or maximise the expected gain of the model on a given training set. When used with the BLEU evaluation metric, this training criteria is also referred to as expected BLEU training (Zens et al., 2007). The objective function for minimum risk training using BLEU is:

$$\begin{aligned} \lambda_{mr} &= \arg\min_{\lambda} \sum_{c=1}^{C} \sum_{\mathbf{e},\mathbf{d}} p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)\ell(\mathbf{e},\mathbf{e}_c) \\ &= \arg\max_{\lambda} \sum_{c=1}^{C} \sum_{\mathbf{e},\mathbf{d}} p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)\text{BLEU}(\mathbf{e},\mathbf{e}_c) \end{aligned} \tag{7.1}$$

where $\{\mathbf{f}_c,\mathbf{e}_c\}_{c=1}^{C}$ is a training corpus consisting of source sentences $\mathbf{f}_c$ and $\mathbf{e}_c$ denotes one or many reference target sentences. Also, $\mathbf{e}$ is the yield of derivation $\mathbf{d}$ and $\text{BLEU}(\mathbf{e},\mathbf{e}_c)$ is the BLEU score of hypothesised translation $\mathbf{e}$ when the reference translation is $\mathbf{e}_c$.

Minimum risk training is an appealing training regime for several reasons:

- The training objective function takes into account the whole distribution of derivations rather than focusing on the single best derivation like in the MERT objective. Therefore, decoding algorithms such as max translation and MBR, which marginalise over derivations, should fare better with a minimum risk trained model.

- The training objective function has the exact same form as the MBR objective used at decoding time. By maintaining a *unified* objective across the translation pipeline, we expect better translation performance.

- Since the objective function is defined in terms of an expectation, model parameters trained with respect to this criterion will already be appropriately scaled for use with probabilistically defined decoding decision rules.

- The training objective function is continuous and differentiable, so, standard gradient-based optimisation techniques, which scale to a large number of features, can be employed.

Computing the minimum risk training objective involves an *intractable summation* over an exponential number of derivations. Previous approaches to minimum risk training for SMT models have approximated this summation using the derivations in an n-best list (Smith and Eisner, 2006; Zens et al., 2007). A more recent approach is that of Li and Eisner (2009) which describes a dynamic programming based algorithm for performing minimum risk training on a translation forest, thus leveraging information from a translation space orders of magnitude greater than that of an n-best list. Further details on these techniques are provided in Section 2.5.4.

7.1.1 Global View of Distribution

The use of translation forests for minimum risk training follows a recent trend in SMT inference tasks such as training and decoding to move away from n-best lists and towards packed representations of a decoder's search graph, referred to as *lattices* in the case of phrase-based models and as *forests* for syntax-based models. Lattices and forests are potentially more informative since they encode many more translation hypotheses in them compared to an n-best list.

Decoding in SMT models typically uses beam search in conjunction with dynamic programming, with low probability derivations heuristically pruned away from the beam for faster decoding. Therefore, even though translation forests and lattices encode many more hypotheses than n-best lists, they still only capture those high probability derivations that have remained in the beam. Blunsom and Osborne (2008) argue that looking at only the most probable derivations is liable to *bias* the learned model to its detriment:

> "The space of derivations contained within the beam will be tightly clustered about a maximum, and thus a model trained with such an approximation will only see a very small part of the overall distribution, possibly leading it astray. Consider the example of a language model feature: as this is a very strong indicator of translation quality, we would expect all

> derivations within the beam to have a similar (high) language model score, thereby robbing this feature of its discriminating power. However if our model could also see the low probability derivations it would be clear that this feature is indeed very strongly correlated with good translations. Thus a good approximation of the space of derivations is one that includes both good and bad examples, not just a cluster around the maximum."

Our implementation of minimum risk training uses sampling to approximate the feature expectations required by the optimisation algorithm. While the majority of the sampled derivations will be drawn from regions of high probability, there should also be some derivations sampled from the rest of the distribution.

To verify whether low probability derivations are indeed generated during sampling, we ran both the sampler and an exhaustive decoder on an example 10 word long input sentence from French-English TEST2008 data set. Note that exhaustive decoding in our model is tractable for a sentence of that length. We ran the sampler for 10,000 iterations which generated 1,100 unique derivations. We then extracted the 1,100 most probable derivations from the search space of the exhaustive decoder. We ran the sampler twice to account for any possible variation in the results. In Figure 7.1, for each sampler run, we plot the true value of the log-probability of each derivation, computed using the procedure described in Section 7.1.1 against (a) its sampler estimated rank, shown as a scatterplot and (b) its true rank, shown as a curve.

We observe that the majority of the sampler data points lie close to the n-best curve in both sampler runs thus indicating that the sampler-estimated distribution closely matches the true distribution. Note too that the mode of the sampled distribution is the true mode of the distribution.

We also find that there is a small but significant number of low probability derivations in the sample set. These are derivations which are likely to be pruned away during beam search and therefore would not appear in n-best lists or lattices but do get observed during sampling. How many such low probability derivations exist in the sample set? In order to find out, we ran the sampler for 10,000 iterations on 10 sentences of lengths shorter than 20 words drawn from the French to English translation task. At the end of each sampler run, we intersected each sample derivation with the search lattice produced by a beam decoder translating the same sentence and calculated the percentage of derivations in the sample set which are not in the lattice.

We found that on average, between 5% and 9% of sampled derivations are outside of the pruned lattice and that around 50% are outside n-best lists containing an equivalent number of derivations. As argued by Blunsom and Osborne (2008), in contrast

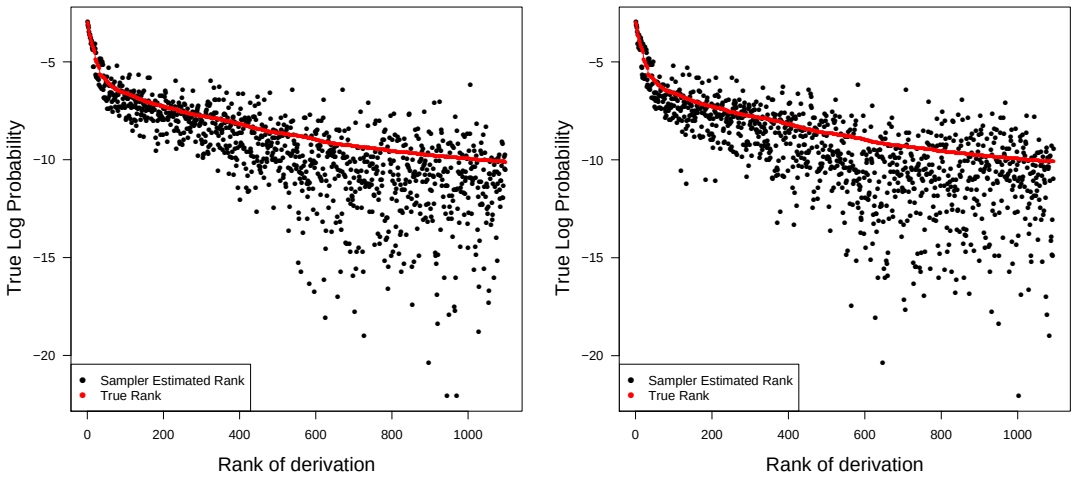


Figure 7.1: Comparison of the true log probability of the top 1,100 most probable derivations as estimated by the sampler (shown in black) and of the 1,100 most probable derivations as per the model (shown in red) for 2 different runs of the sampler on the same input sentence.

to the more blinkered view obtained when using beam search, the presence of these derivations should provide the model with a more global view of the distribution and potentially enhance its power to *discriminate* between good and bad translations.

7.1.2 Comparison to MERT

The minimum risk training objective is in fact a smoothed version of the objective function used during minimum error rate training. To observe this, let us first define the MERT objective when used with BLEU as error function:

$$\lambda_{\text{MERT}} = \arg\max_{\lambda} \sum_{c=1}^{C} \text{BLEU}(\arg\max_{\mathbf{e}} p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda),\mathbf{e}_c) \tag{7.2}$$

$$= \arg\max_{\lambda} \sum_{c=1}^{C} \lim_{\alpha\to\infty} \sum_{\mathbf{e},\mathbf{d}} \text{BLEU}(\mathbf{e},\mathbf{e}_c) p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda,\alpha) \tag{7.3}$$

Equation 7.2 is the standard MERT objective function. In Equation 7.3, the introduction of α, a term that scales the probability distribution in the same way as in Section 6.1.2.1, smooths the objective in (7.2). When $\alpha \to \infty$, the objective in (7.2) is recovered, whereas when $\alpha = 1$, the objective in (7.3) is equal to that of minimum risk training (7.1).

Figure 7.2, reproduced from Och (2003), compares the error surface of the smoothed and unsmoothed objective as the values of two different feature weights are varied.

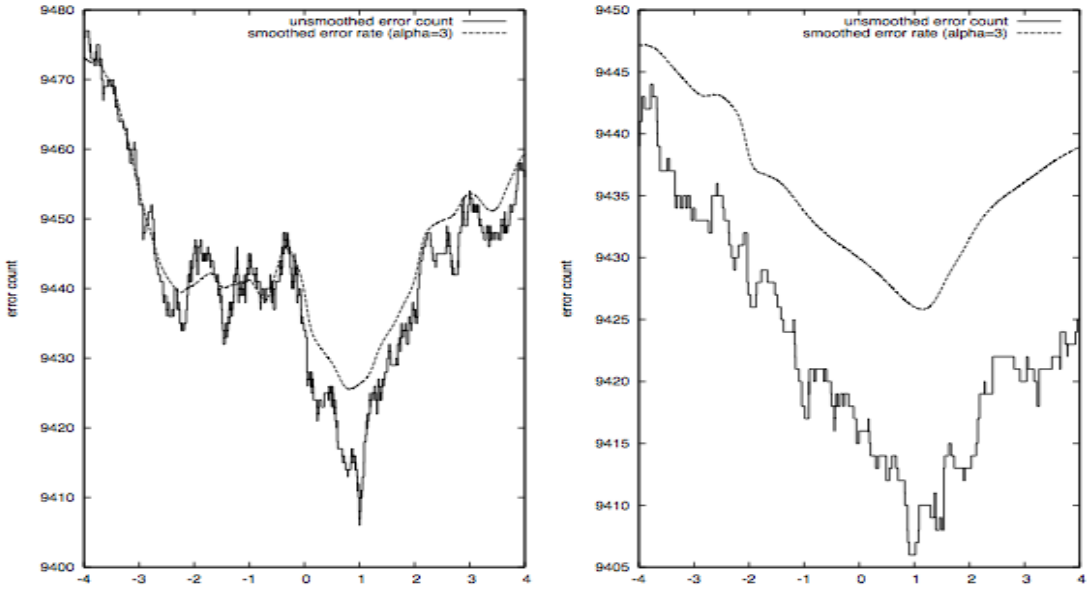


Figure 7.2: Shape of error count and smoothed error count for two feature weights. Diagram reproduced from (Och, 2003).

We can see that both objective functions are non-convex: they are riddled with local optima. However, a much smoother curve is obtained by scaling the distribution. While Och (2003) found that the smoothed and unsmoothed objectives give almost identical results, (Smith and Eisner, 2006; Zens et al., 2007; Li and Eisner, 2009) all report performance improvements using minimum risk training compared to MERT.

Since the smoothed objective is continuous and differentiable, it is amenable to gradient descent methods of optimisation which are able to scale to a large number of features. Li and Eisner (2009) exploit this characteristic to train a model with 20,000 sparse features in addition to the typical features used in SMT models, resulting in improved translation performance.

On the other hand, as discussed in Section 2.5.3, MERT cannot be used to train models with more than a few (around 15) features. This is because the single-parameter line minimisation algorithm at the core of the MERT optimisation algorithm does not scale well.

7.2 Computing Feature Expectations

A crucial term needed for gradient-based optimisation of log-linear models is the expectation of the features of the model under the current parameter settings. Since the number of derivations in the model is exponential in the length of the input sentence, it is too computationally expensive to calculate this expectation exactly for most sentences and therefore approximations are required.

In this thesis, we approximate the feature expectation using Monte Carlo sampling. Given a set of N samples drawn from the distribution, this expectation is given by:

$$\mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f})}[h] \approx \frac{1}{N}\sum_{i=1}^{N} h(\mathbf{e}_i,\mathbf{d}_i,\mathbf{f}) \qquad (\mathbf{e}_i,\mathbf{d}_i,\mathbf{f}) \sim p(\mathbf{e},\mathbf{d}|\mathbf{f}) \tag{7.4}$$

Previous approaches have approximated the space of all derivations with either an n-best of derivations (Smith and Eisner, 2006) or with a packed representation of all the derivations in the pruned search space of a first-pass beam decoder (Li and Eisner, 2009). Both these approaches operate over a search space pruned in the first place using heuristics which potentially introduce arbitrary biases in the resulting expectations.

As mentioned in Section 7.1.1, the argument against using heuristically pruned search spaces for computing expectations in SMT log-linear models was first laid out by Blunsom and Osborne (2008) who claim that the resulting approximation is too concentrated around the mode of the distribution. A similar concern motivates the work of Bouchard-Côté et al. (2009). Both Blunsom and Osborne (2008) and Bouchard-Côté et al. (2009) present MCMC-based solutions to this problem. The former augment the pruned forest of their syntax-based translation model with derivations sampled from the distribution and show that this brings about an improvement in translation performance. The latter propose an auxiliary variable sampling technique for computing expectations in a bilingual parsing task and present empirical evidence showing that their technique yields a reduction in bias.

To verify whether the claims of Blunsom and Osborne (2008) and of Bouchard-Côté et al. (2009) are corroborated in the case of phrase-based translation models, we ran experiments to compare expectations computed using three different *evidence spaces*: a) n-best lists b) pruned lattices and c) sample set. As gold standard, we exhaustively decoded 10 sentences from the French-English TEST2008 data set of lengths shorter than 20 words using Moses and computed the *exact feature expectations* on the resulting unpruned lattice using a variant of the forward-backward algorithm used for training Hidden Markov Models (HMMs). Note that while this algorithm

Condition	Size	DP	WP	LM	TM1	TM2	TM3	TM4	PP
Exact	10^{70}	-1.72	-13.57	-66.13	-14.99	-23.79	-7.72	-14.46	8.19
Lattice	10^{11}	-1.68	-13.57	-66.12	-14.98	-23.79	-7.70	-14.46	8.18
N-best	2×10^2	-1.36	-13.63	-65.56	-13.58	-23.46	-6.89	-14.60	7.84
Samples	2×10^2	-2.13	-13.68	-67.04	-14.45	-23.17	-7.74	-14.62	8.29
N-best	10^3	-1.44	-13.61	-65.79	-13.87	-23.45	-7.08	-14.51	7.92
Samples	10^3	-2.00	-13.62	-66.57	-14.82	-23.62	-7.77	-14.58	8.24
N-best	2×10^3	-1.46	-13.60	-65.85	-14.03	-23.48	-7.16	-14.49	7.95
Samples	2×10^3	-1.68	-13.60	-66.44	-14.81	-23.67	-7.76	-14.53	8.18
N-best	5×10^3	-1.49	-13.58	-65.90	-14.24	-23.53	-7.26	-14.45	8.00
Samples	5×10^3	-1.80	-13.56	-66.21	-14.94	-23.76	-7.70	-14.41	8.19
N-best	10^4	-1.52	-13.58	-65.89	-14.38	-23.58	-7.32	-14.43	8.03
Samples	10^4	-1.87	-13.56	-66.10	-14.98	-23.74	-7.72	-14.40	8.21
N-best	2×10^4	-1.55	-13.56	-65.75	-14.50	-23.59	-7.38	-14.39	8.05
Samples	2×10^4	-1.86	-13.59	-66.24	-14.95	-23.67	-7.75	-14.47	8.22

Table 7.1: Comparison of the expectations of features in a standard phrase-based model computed exactly and estimated using a pruned lattice, n-best lists and sampling. The features of the model are a distortion feature (DP), a word penalty feature (WP), a language model (LM) feature, four translation model features(TM1, TM2, TM3, TM4) and a phrase penalty feature (PP). Size indicates the number of derivations considered while computing the expectation.

runs in time linear to the size of the lattice, the packed representation of the unpruned search space is far too large for this algorithm to be practical for longer sentences.[1] We also ran Moses with default pruning parameter values and computed the expectations on the resulting pruned lattices. For the expectations computed using sampling and n-best lists, we varied the number of derivations being considered from 100 up to a maximum of 20,000.

[1]The unpruned lattice for a 19 word source sentence takes up 40G of RAM when stored in memory.

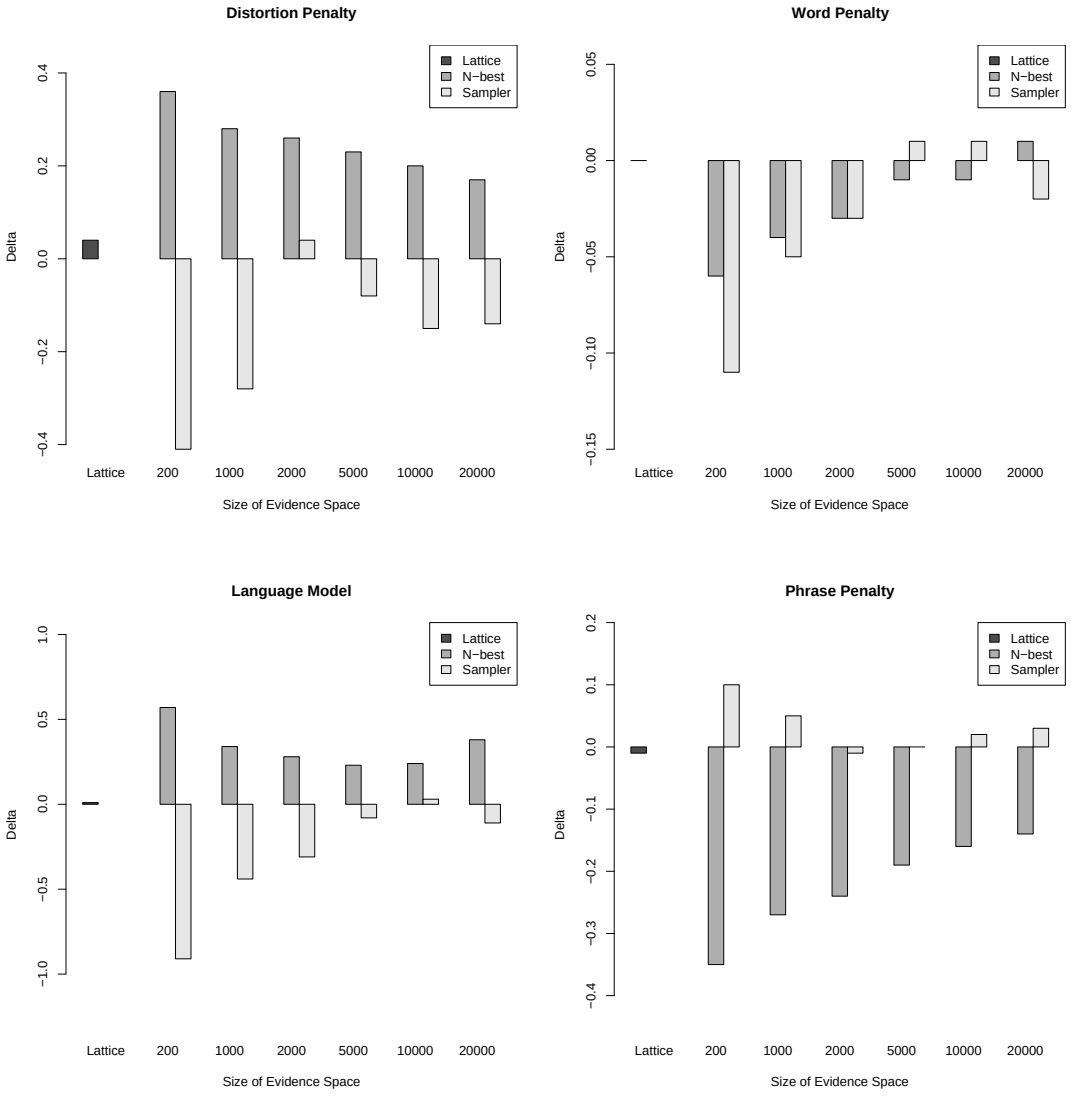

Distortion Penalty
Word Penalty
Language Model
Phrase Penalty
Lattice
N−best
Sampler
Delta
Size of Evidence Space
Lattice
200
1000
2000
5000
10000
20000

Figure 7.3

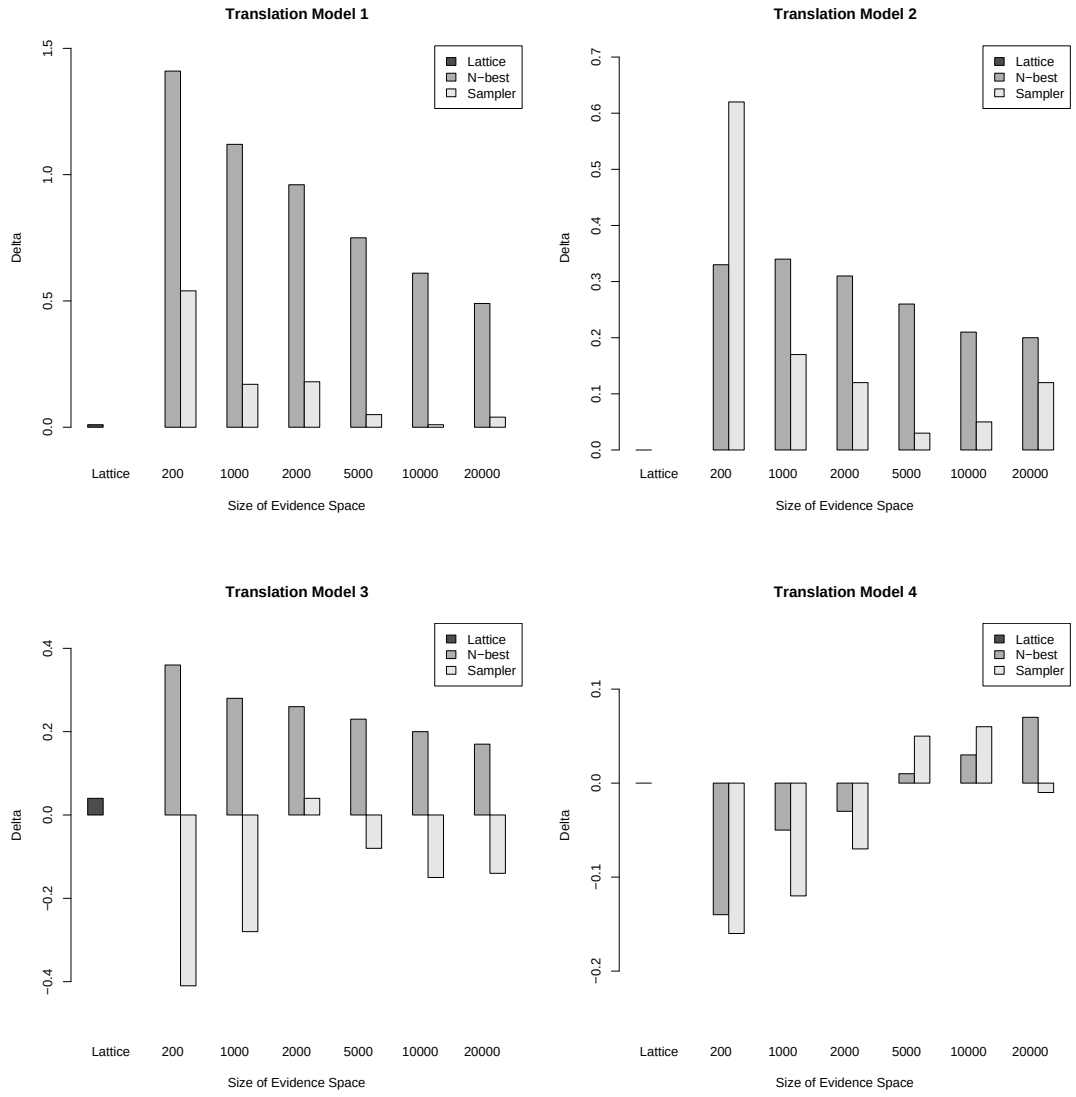


Figure 7.3: Difference between estimated feature expectation and true feature expectation as a function of evidence space type and size for each of the eight features in the model.

Table 7.1 summarises the results of our experiments. We report the expectation of each of the eight features in our model averaged over 10 sentences. The features are a distortion feature, a word penalty feature, a phrase penalty feature, a language model feature and 4 translation model features. Further details about these features are provided in Chapter 4. We also show the number of derivations considered in each technique. In an unpruned lattice, an average of 10^{70} paths are considered. Pruning reduces this number considerably although the resulting lattice still contains a very large number of paths (10^{11}).

To help show the different trends for each feature in the model, we also plot the difference between the estimated feature expectation and the true expectation as a function of the evidence space type and, in the case of n-best lists and sampling, size. These plots are shown in Figure 7.3.

From Figure 7.3, we see that the expectations computed on the pruned lattice are very close approximations to the true expectations. N-best approximations, on the other hand, are not adequate. Taking the language model feature as example, for n up to 20,000, the expectation of this feature is systematically overestimated. This is because the n-best list mostly contains good translations which have high language model scores. The lack of adequate negative examples results introduces a bias in the expectation. A similar situation can be seen for the case of the backward phrasal translation feature (Translation Model 1) where the lack of adequate negative examples leads to the expected value of this feature to be overestimated.

Compared to n-best lists of the same size, sampling, on average, provides more accurate estimates of the expectations of interest. For example, a very good approximation of most feature expectations can be obtained with only 2000 samples while for the same number of derivations, the n-best list estimates are in most cases much worse. On the other hand, whereas the n-best list estimates get steadily better as the evidence grows, there can be variance in the sampling estimates. This is due to the inherent stochasticity of MCMC sampling algorithms.

Table 7.1 and Figure 7.3 evidence that sampling-based feature expectations are more accurate than n-best list based ones. However, it suggests that we would be better off computing the expectations by running a polynomial time dynamic programming algorithm on the pruned lattice rather than by sampling. Not only are the estimates more accurate but the algorithm is faster too. This result also contradicts the claims of (Blunsom and Osborne, 2008; Bouchard-Côté et al., 2009) who state that heuristic

pruning provides biased estimates of the true feature expectations. So should we be sampling at all?

We present three arguments in favour of sampling. Firstly, while the lattice might be a good approximation of the search space for the relatively short sentences we considered, it might not be for longer sentences where many more derivations need to be pruned so as to ensure efficient decoding. In these cases, the more theoretically principled sampling approach could be more advantageous. Since in this thesis we translate sentences of length up to 120 words, sampling is potentially a good solution.

Secondly, note that the features in our model are few and are dense. Dense features are features which are active on every solution. For example, every derivation in the model has an associated language model score. An active research topic in SMT is to move towards models with large numbers of sparse features. Such models have been shown to give state of the art results in many structured prediction tasks in NLP such as dependency parsing (Mcdonald et al., 2005) and sentence compression (Mcdonald, 2006) and also in SMT (Chiang et al., 2008b, 2009). In these models, features may fire on many or only on a few solutions.

The translation model of Blunsom and Osborne (2008) which contains 2.9 million, mostly sparse, features is an instance of such a class of models. Note that in such models, features with low expectations may be completely skipped if their supporting derivations are pruned during beam search. In contrast, these low probability derivations may be observed when using sampling. We hypothesise that this is the reason why Blunsom and Osborne (2008)'s model benefits from sampling as compared to pruning assisted inference.

In the experiments in this thesis, we use a model with a small number of dense features. For this model, lattice-based techniques are likely to outperform sampling-based approaches. However, our model can easily be extended to include additional features. The sampling techniques presented here provide sound approximate inference solutions for such a model.

A final reason for using sampling is that it provides a general purpose solution for computing expectations of *any function* defined over structures in the model. For example, in Section 7.3.3 we will see that sampling can be used to compute the Monte Carlo estimate of the gradient of the entropy of a log-linear model in a straightforward manner; computing the same term over a packed representation requires designing special purpose dynamic programming algorithms, such as those presented in Li and Eisner (2009), which may be hard to implement correctly.

7.3 Sampling for Minimum Risk Training

In this section, we describe how sampling can be used to implement two variants of minimum risk training considered in this thesis; *sentence sampling* optimises an objective defined at the sentence level while *corpus sampling* optimises a corpus-based objective.

7.3.1 Sentence Sampling

We begin by defining our training objective at the sentence level using BLEU as gain function. Note that minimum risk training can be used in conjunction with any gain or loss function of interest. (Och, 2003) shows empirically that we achieve best results for any particular loss function when we use that function in our parameter estimation objective function. Since our model's translation performance is evaluated using BLEU, we choose to use it as gain function during training.

The expected gain $\mathcal{G}$ of the probabilistic translation model when defined at the sentence level is given by:

$$\mathcal{G} = \sum_{c=1}^{C} \sum_{\mathbf{d} \in D(\mathbf{f}_c)} p(\mathbf{e}, \mathbf{d} | \mathbf{f}_c) \text{SBLEU}(\mathbf{e}, \mathbf{e}_c) \tag{7.5}$$

where $\mathcal{C}$ is a training corpus $\{\mathbf{f}_c, \mathbf{e}_c\}_{c=1}^{C}$ of source sentences and one or many reference sentences for each source sentence, $D(\mathbf{f}_c)$ is the set of all derivations that the model can generate given source sentence $\mathbf{f}_c$, and $\mathbf{e}$ is the yield of derivation $\mathbf{d}$.

Note that since we compute the gain at the sentence level whereas BLEU is defined at the corpus level and does not decompose over individual sentences, we approximate BLEU using the sentence-level variant presented in Section 2.2.1. In Equation 7.5, $\text{SBLEU}(\mathbf{e}, \mathbf{e}_c)$ denotes the sentence level BLEU score of hypothesised translation $\mathbf{e}$ when the reference translation is $\mathbf{e}_c$.

The objective function in (7.5) is identical to the one used by (Zens et al., 2007), except that the latter approximate the space of all derivations with an n-best list. In their implementation of minimum risk training, the objective function is optimised using the Downhill Simplex algorithm, a general purpose optimisation procedure. This method which requires only function evaluations, not derivatives, is not very efficient in terms of the number of function evaluations that it requires. Another drawback with the algorithm is that it does not scale well to a large number of parameters.

Recall that the probabilistic formulation of our translation model is given by:

$$p(\mathbf{e},\mathbf{d}|\mathbf{f};\lambda) = \frac{\exp\left[\lambda \cdot h(\mathbf{e},\mathbf{d},\mathbf{f})\right]}{\sum_{\langle \mathbf{e}',\mathbf{d}'\rangle} \exp\left[\lambda \cdot h(\mathbf{e}',\mathbf{d}',\mathbf{f})\right]} \tag{7.6}$$

where h is a feature vector and λ is a weight vector of m components each.

We can exploit Equation 7.6 to facilitate the optimisation of the objective in (7.5). This is because, given the probabilistic formulation of the translation model, the objective function is continuous and differentiable with respect to the model parameters λ. Therefore, we can use powerful gradient descent based optimisation techniques. During optimisation, algorithms such as Stochastic Gradient Descent (SGD) only require the value of the gradient of the objective function with respect to each parameter λ_m; the value of the objective function itself is not needed. This gradient is given by:

$$\begin{aligned} \frac{\partial \mathcal{G}}{\partial \lambda_m} &= \sum_{c=1}^{C} \sum_{\mathbf{d} \in D(\mathbf{f})} \text{SBLEU}(\mathbf{e},\mathbf{e}_c) \frac{\partial p}{\partial \lambda_m} \\ \text{where } \frac{\partial p}{\partial \lambda_m} &= p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)(h_m - \mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)}[h_m]) \end{aligned} \tag{7.7}$$

Notice that the gradient can be rewritten in terms of an expectation thus making it amenable to Monte Carlo estimation:

$$\frac{\partial \mathcal{G}}{\partial \lambda_m} = \sum_{c=1}^{C} \mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)} \left[\text{SBLEU}(\mathbf{e},\mathbf{e}_c) \cdot (h_m - \mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)}[h_m])\right] \tag{7.8}$$

Calculating (7.8) therefore requires a first pass through the sample set to calculate $\mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)}[h_m]$ (the expectation of the feature values under the distribution), followed by a second pass to compute the gradient itself.

7.3.2 Optimisation Algorithm

As we have shown how to compute the gradient of the objective function with respect to the model parameters we want to optimise, we can use any standard first-order optimisation technique. Since the sampler introduces stochasticity into the gradient and objective, we use stochastic gradient descent (SGD) methods which are more robust to noise than more sophisticated quasi-Newtonian methods like L-BFGS (Liu and Nocedal, 1989). For the experiments in this thesis, we used the approximated exponentiated gradient descent algorithm of Schraudolph (1999).

At each iteration t of the optimisation, this algorithm updates the weight vector λ based on the gradient $\partial \mathcal{G}_t$ of the objective function at iteration t and a dynamic learning rate vector η which is composed of a rate for each of the m features in the model:

$$\lambda_{t+1} = \lambda_t + \eta_t \cdot \partial \mathcal{G}_t \tag{7.9}$$

The learning rates η are adapted based on a linear approximation to exponentiated gradient descent described in Schraudolph (1999) :

$$\eta_t = \eta_{t-1} \cdot \max(0.1, 1 + \mu \cdot \partial \mathcal{G}_t \cdot v_t) \tag{7.10}$$

where μ is a user-defined global meta-learning rate and the multiplier is set to be at least 0.1 to prevent unreasonably small or negative values.

The *gradient trace* v measures the effect that a change in the local learning rate has on the corresponding weight. Assuming that θ_{t+1} depends only on η_t, the gradient trace can be simplified to:

$$v_{t+1} = \eta_t \cdot \partial \mathcal{G}_t \tag{7.11}$$

Note that this is an *online algorithm*: the gradient of the objective function is approximated by the gradient of a single training instance. As the algorithm goes through the training data, it performs a parameter update after seeing each training example. Online learning algorithms contrast with batch learning algorithms which make a parameter update only after inspecting the entire training data. As a result, batch algorithms can be expensive when dealing with large amounts of training data. In contrast, online learning algorithms, while only using an approximation of the true gradient, typically converge faster since parameter updates are done after each training instance. A compromise between online learning and batch learning uses mini-batches where the true gradient is approximated by a sum over a small number of training examples. For the experiments in this thesis, we use mini-batches.

Online learning can be susceptible to the order in which training instances are presented to the algorithm. In order to account for this, we draw batches of randomly chosen training instances from the training set.

The approximated exponentiated gradient descent algorithm used in this thesis is a simple algorithm to implement and contains only two hyper-parameters, μ and η_0. While in principle we could have a different learning rate for each feature, we typically use the same initial value for all features.

7.3.3 Deterministic Annealing

Global optimisation of non-convex functions with local optima is a very hard problem (Torn and Zhilinskas, 1989). The algorithms available for optimisation of non-convex functions, including the class of stochastic gradient descent methods, can only be expected to find a local optimum.

In our initial sentence sampling experiments, we observed a tendency for translation performance on held-out data to quickly increase to a maximum and then plateau (these experiments are described in the next section). Hypothesising that we were being trapped in local maxima as $\mathcal{G}$ is non-convex, we decided to employ *deterministic annealing* (DA; Rose (1998)).

Note that *any* hill-climbing method, including gradient descent, is liable to get stuck in a local optimum depending on its initialisation point (this is why, for instance, EM or the optimisation algorithm inside MERT require multiple random restarts). DA alleviates this dependency by first smoothing the objective function into a convex one for which it is easier to find a global optimum. In the next step, the function is transformed into one which is a little harder to optimise; the solution to the preceding step is used as the new initialisation point. As the current function is similar to the previous one, a local optimum of the current function should be close by and therefore easy to find, and possibly be the global optimum. DA is an iterative algorithm which terminates when the current function being optimised is equal to the original one. The algorithm provides no guarantee as to the goodness of the optimum eventually found but in practice has been found to give good results for many NLP tasks (Smith and Eisner, 2004, 2006; Smith, 2006).

Our instantiation of deterministic annealing is based on the work of Smith and Eisner (2006). It involves the addition of an entropic prior to the objective in Equation (7.5) to give

$$\mathcal{G} = \sum_{c=1}^{C} \left[\left(\sum_{\mathbf{d} \in D(\mathbf{f}_c)} p(\mathbf{e}, \mathbf{d} | \mathbf{f}_c) \text{SBLEU}(\mathbf{e}, \mathbf{e}_c) \right) + T \cdot H(p) \right] \tag{7.12}$$

where $H(p)$ is the entropy of the probability distribution $p(\mathbf{e}, \mathbf{d}|\mathbf{f}_c)$, and $T > 0$ is a temperature paramater which is gradually lowered as the optimisation progresses according to a configurable *cooling schedule*.

At high temperature settings, the objective function is dominated by the entropy term - the optimiser is lead to find weights which describe a high entropy or a fairly flat distribution. As the temperature is gradually diminished, the impact of expected

gain overshadows the entropy term and pushes the optimiser towards a more peaked distribution.

Using the definition of the entropy of a distribution ($H(p) \stackrel{\text{def}}{=} -\sum p \log p$) and the product rule, we can differentiate (7.12) with respect to λ_m to obtain the following expression for the annealed gradient:

$$\frac{\partial \mathcal{G}}{\partial \lambda_m} = \sum_{c=1}^{C} \sum_{\mathbf{d} \in D(\mathbf{f})} \left(\text{SBLEU}(\mathbf{e}, \mathbf{e}_c) - T\left(1 + \log p(\mathbf{e}, \mathbf{d}|\mathbf{f}_c)\right)\right) \frac{\partial p}{\partial \lambda_m}$$
$$\text{where } \frac{\partial p}{\partial \lambda_m} = p(\mathbf{e}, \mathbf{d}|\mathbf{f}_c; \lambda)(h_m - \mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f}_c;\lambda)}[h_m]) \tag{7.13}$$

In comparison to the gradient in (7.7), the annealed gradient also requires computing the log probability of each derivation. When using sampling, this term can be computed trivially. This is also the case when performing deterministic annealing using n-best lists as in Smith and Eisner (2006), although the estimate of the probability is likely to be biased. A dynamic programming algorithm for computing the gradient of the entropy over a hypergraph is presented in Li and Eisner (2009).

When the temperature is high, the contribution of the entropy term also serves to *regularise* the model so that it does not overfit to the training examples seen early during optimisation. We also experimented with using a Gaussian prior (Chen and Rosenfeld, 1999) which penalises feature weights that grow too large but found no additional benefits.

In our implementation of DA, at each temperature setting, we performed a configurable number of iterations (typically between 10 and 20) of SGD. In their deterministic annealing formulation, (Smith and Eisner, 2006; Li and Eisner, 2009) have an additional free parameter γ that scales the exponential distribution given in (7.6) and whose value is optimised along with λ. We did not find any benefits from optimising this term and therefore left its value to 1.

7.3.4 Sentence Sampling Experiments

We ran sentence sampling experiments on three different language pairs: Arabic-English, French-English and German-English. As described in Chapter 4, the tuning set for each of the language pairs consisted of 1,043, 2,000 and 2,000 sentence pairs respectively and contained 10 reference sentences for each source sentence in the case of Arabic-English and a single reference per source sentence for the other two language pairs.

The sampler is run with random initialisation using our default setup of two sampling chains each starting with a burn-in step of 100 sampling iterations. In Table 7.1, we saw that a good approximation of the feature expectations can be obtained with a sample size of around 5,000. In the sentence sampling experiments, we ran each of the 2 chains used until 2,000 samples were collected ending up with 4,000 samples in total.

We initialised the optimisation algorithm with all feature weights set to zero and use mini-batches of randomly drawn 100 training instances. We found in preliminary experiments that a batch of that size allowed us to perform enough iterations of gradient descent while at the same time providing good estimates of the true gradient.

The optimisation algorithm has two hyper-parameters: μ and η_0. In preliminary experiments, we found that setting them both to a value of 2 provided a good balance between rapid optimisation and good translation performance on a held-out set.

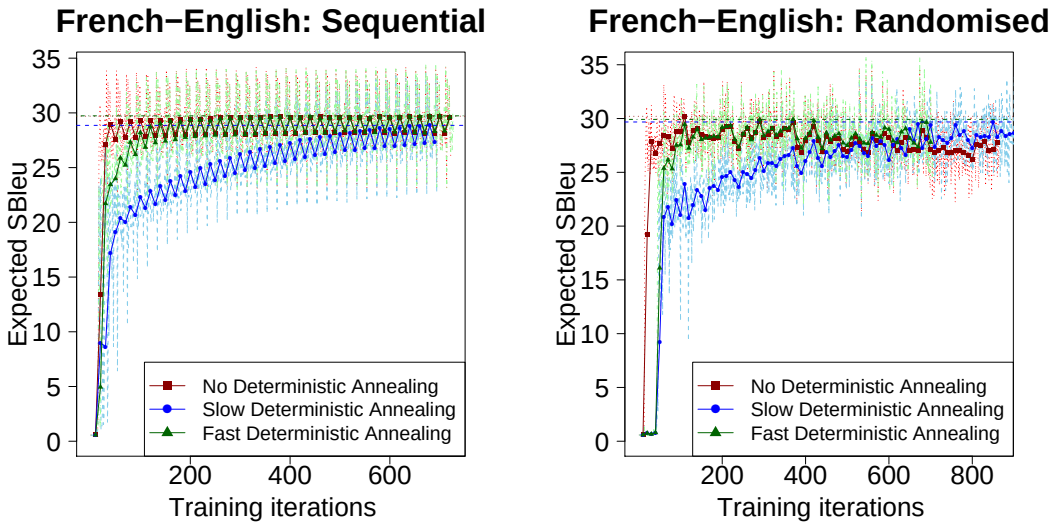
The deterministic algorithm has 2 hyper-parameters: a) the initial temperature T and b) a cooling schedule. We found that when initialising to T to 100, optimisation was fast and converged to a good optimum. We experimented with two exponentially decaying cooling schedules, one with a slow decay rate of 0.9 and one with a fast decay rate of 0.5, performing 20 iterations of SGD optimisation at each temperature setting. Training was stopped when T reached 0.0001 or at the end of 48 hours of processing, whichever came first.

To account for the variance in results due to the stochasticity of the sampler, we ran training for each condition 5 different times and report the averaged results.

7.3.4.1 Training Performance

Figure 7.4 shows the training learning curve of expected SBLEU for French-English as a function of the number of iterations of the optimiser. Each iteration is equal to a pass over 5% of the tuning data. We compare sequential batch training with random batch training. Expected SBLEU for the no DA, slow DA and fast DA conditions are shown with points in red, blue and green respectively in the background. In the foreground, we plot the average expected SBLEU after every 10 iterations. We also indicate with horizontal lines the maximum averaged expected SBLEU value for each training condition.

In Figure 7.4a, without annealing, a peak is reached quickly after which the curve plateaus. With deterministic annealing, performance at the start of the optimisation is low for a long time. This is because when T is high, the objective function is



(a) (b)

Figure 7.4: Expected tuning SBLEU averaged across 5 training runs. (a) shows training with sequential batches and (b) shows training with randomised batches. Best scores are indicated by dotted lines.

dominated by the entropy term: the optimiser finds weight settings which produce high entropy distributions rather than good translations. As T lowers eventually, DA ends up finding weights which produce good translations. With fast annealing, these weights are obtained quicker than with slow annealing. In fact, in Figure 7.4a, unlike fast annealing performance which has already plateaued, the training objective is still increasing when using slow annealing.

While the learning curve during sequential training for each training regime is smooth, this is not the case in Figure 7.4b reflecting the random nature of the mini-batches. Still the general trend during training is similar to that in Figure 7.4a. There are two main differences. First, the peaks reached during random batch training are slightly higher than those obtained during sequential batch training suggesting that the learner finds better weights in the former case. This is in line with conventional wisdom about online learning algorithms which states that it is preferable to randomise the order in which training instances are presented to the learner. For the remaining experiments in this thesis, we use randomised batches. We also observe that the learning curves for both slow DA and no DA drop in the later iterations of randomised training whereas in Figure 7.4a they plateau. We attribute this behaviour to the vagaries

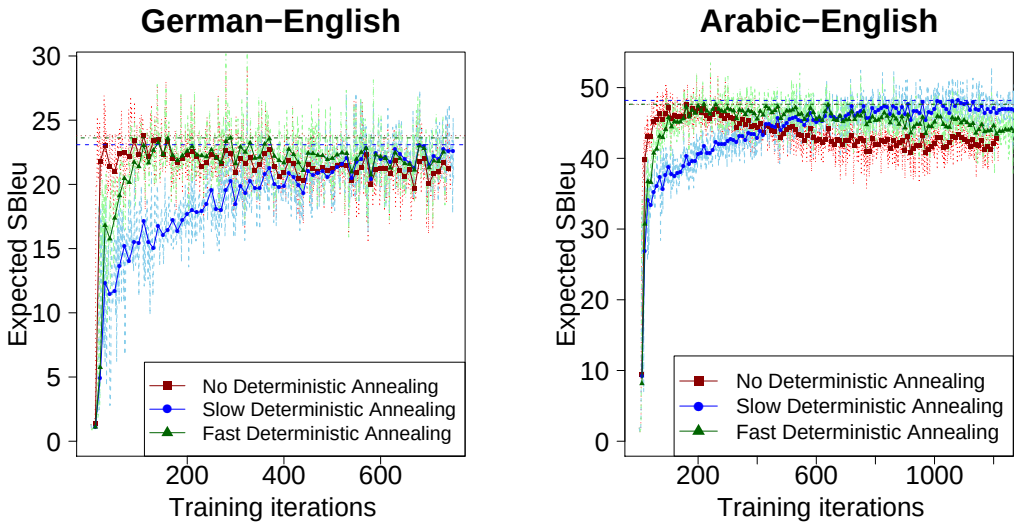


Figure 7.5: Expected tuning SBLEU averaged across 5 training runs using random batches. Best scores are indicated by dotted lines.

of randomised training: the objective function might appear to drop if the learner encounters a long consecutive run of hard to translate sentences.

Figure 7.5 shows similar random batch training learning curves for German-English and Arabic-English. Each iteration corresponds to seeing roughly 10% of the tuning data in the case of Arabic-English and 5% for German-English.

When comparing the 3 training setups, we see that for 2 language pairs the best training expected SBLEU is obtained when no determistic annealing is used. For the third language pair, Arabic-English, if the optimisation is run for long enough, slow annealing eventually marginally outperforms no annealing. These results seem to suggest that the benefits of deterministic annealing are marginal at best and that if training time is at a premium, then it is better to forego annealing.

7.3.4.2 Decoding Performance on Held-out Data

Recall that the objective function used during training differs from test time decision rules. Thus, there is no guarantee that weights which give the best expected SBLEU results at training time will also produce the best translations for the different decoding decision rules under consideration. To account for the discrepancy between training and testing objectives, we output feature weights after every 50 iterations of training which we then use to measure max derivation, max translation and MBR decoding performance on a held-out set by running the sampler as a decoder. The held-out set for

the Arabic-English was the 663 sentence long MT03 data set and for German-English and French-English was the 1000 sentence long TEST2007A set. Further details on the held-out sets are given in Chapter 4.

Note that we can use the sampler as decoder with the learnt weights without having to resort to feature weight scaling. This is because the probabilistic training objective function produces weights which are already appropriately scaled. These weights can be plugged in the sampler directly. This is also the case even when performing MBR decoding (recall that when using a beam decoder to perform MBR decoding, MERT optimised weights need to be scaled first.)

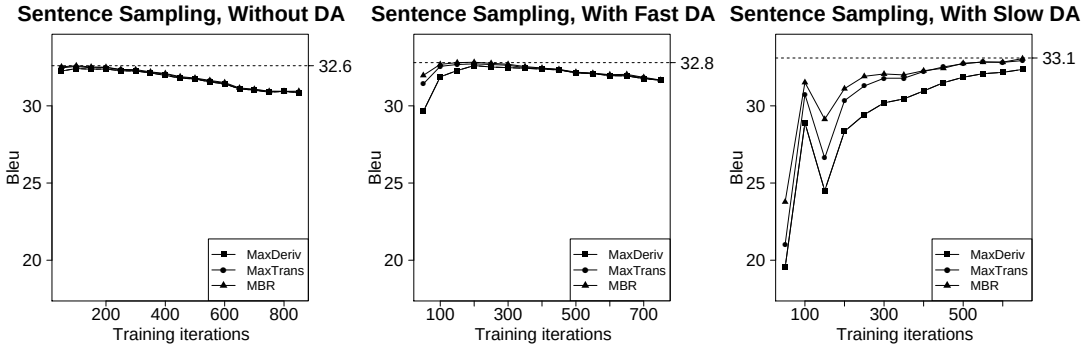


Figure 7.6: Held-out performance for French-English training averaged across 5 training runs. Best scores achieved are indicated by dotted line.

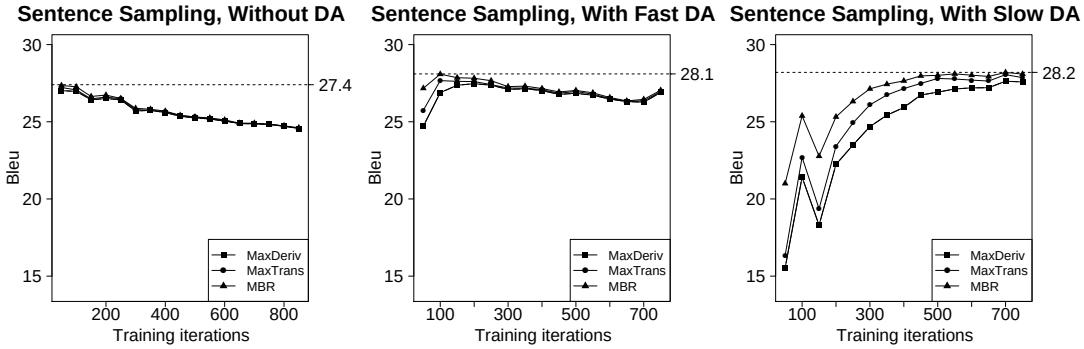


Figure 7.7: Held-out performance for German-English training averaged across 5 training runs. Best scores achieved are indicated by dotted line.

Figures 7.6, 7.7 and 7.8 show the translation BLEU scores on the French-English, German-English and Arabic-English held-out sets respectively. The figures compare sentence sampling without DA, with slow DA and with fast DA. We observe that

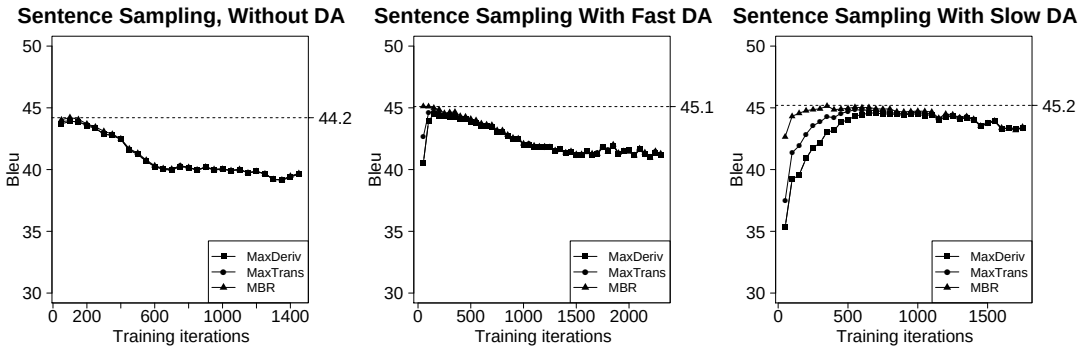


Figure 7.8: Held-out performance for Arabic-English training averaged across 5 training runs. Best scores achieved are indicated by dotted line.

deterministic annealing is beneficial for all three language pairs with slow deterministic annealing giving the best translation performance. While the difference between slow and fast DA is small (between 0.1 and 0.3% BLEU), the difference with no annealing is much more marked (between 0.5 and 1.0% BLEU). These results are in contrast to what we observed during training where, in 2 out of 3 cases, minimum risk training without annealing gave better expected SBLEU performance.

The results suggest that, without annealing, the learning algorithm *overfits* the training data whereas the addition of the entropic prior during annealing plays an important role in *regularising* the model and in improving its ability to *generalise* to unseen data. Even with annealing, the learning algorithm does eventually overfit the training data. This is demonstrated by the fact that in all three training regimes, test time performance eventually starts to drop.

We also observe when not using any annealing that all three decoding decision rules give very similar results. This is because the optimisation algorithm finds weights which produce low entropy distributions. Since there is not much diversity amongst the sampled derivations, the benefits of marginalising over derivations are mitigated. In contrast, the addition of the entropic prior during annealing encourages high entropy distributions. In the presence of increased diversity, the benefits of summing over derivations are clear: max translation decoding does better than max derivation decoding and MBR does best. As training proceeds and the temperature decreases to zero, the model sharpens, thus voiding the benefits of max translation and MBR decoding.

The benefits of high entropy distributions on translation performance when using max translation and MBR decoding rules can be seen in Table 7.2 where for all three language pairs, we compare the best averaged max derivation, max translation and

Language Pair	Training regime	Iteration	MaxD	MaxT	MBR	Entropy
Fr-En	No DA	100	32.5	32.6	32.6	6.51
	Fast DA	200	32.7	32.7	32.8	7.12
	Slow DA	750	32.5	33.0	**33.1**	7.91
De-En	No DA	50	27.1	27.4	27.4	6.54
	Fast DA	100	26.9	27.6	28.1	8.35
	Slow DA	550	26.9	27.7	**28.2**	8.37
Ar-En	No DA	100	43.8	44.0	44.2	6.11
	Fast DA	50	40.0	42.1	45.1	8.65
	Slow DA	350	43.3	44.4	**45.2**	8.31

Table 7.2: Entropy of estimated distribution at iteration at which best translation performance is obtained on French-English, German-English and Arabic-English held-out sets. Figures in bold indicate best BLEU score for each language pair. MBR decision rule produces better translations than max derivation (MaxD) and max translation (MaxT).

MBR decoding held-out performance with the entropy of the derivation distribution estimated by the sampler. We also report the training iteration at which the best translation performance was obtained. Table 7.2 confirms that the entropies of the distribution obtained using DA are higher than when not using any annealing and that these high entropy distributions benefit greatly from decision rules which marginalise over derivations.

7.4 Corpus Sampling

While the objective functions in Equations (7.5) and (7.12) use a sentence-level variant of BLEU as gain function, the model's test-time performance is evaluated with corpus level BLEU. As we discussed in Section 2.2.1, BLEU is not decomposable at the sentence level: there is no guarantee that improving the translation of one sentence leads to an increase in the overall score, or that degrading the translation of a sentence will lead to a drop in overall score. Thus, maximising the expected SBLEU of each sentence in the corpus individually does not necessarily lead to maximising the expected BLEU of the corpus of sentences.

There have been previous attempts to address the discrepancy between the tuning and the testing gain functions. In particular, the training objective function used in (Smith and Eisner, 2006; Li and Eisner, 2009) maximises expected log BLEU gain, defined as the change in corpus log BLEU brought about by the inclusion of a given translation relative to not including it in the corpus. Smith and Eisner (2006) report that models trained using this new objective function outperform ones trained using the objective in (7.5). However, when using it for hypergraph-based MBR decoding, Kumar et al. (2009) remark that this approximation is not guaranteed to be a close match to the actual corpus BLEU.

In this section, we present *corpus sampling*, an algorithm for maximising expected corpus BLEU directly. Given a training corpus of the form $\langle \mathcal{C}_F, \mathcal{C}_{\hat{E}} \rangle$ where $\mathcal{C}_F$ is a set of N source sentences $\mathbf{f}^1 ... \mathbf{f}^N$ and $\mathcal{C}_{\hat{E}}$ is a set containing the reference translations for each source sentence, let $\mathcal{D}(\mathcal{C}_F)$ denote the set of all hypothesised translations of $\mathcal{C}_F$, i.e. $\mathcal{D}(\mathcal{C}_F) = D(\mathbf{f}^1) \times D(\mathbf{f}^2) \times \cdots D(\mathbf{f}^N)$. Then, given a corpus translation $\mathcal{C}_E$ such that $\mathcal{C}_E \in \mathcal{D}(\mathcal{C}_F)$, $P(\mathcal{C}_E|\mathcal{C}_F)$ is the probability of translating $\mathcal{C}_F$ as $\mathcal{C}_E$ and BLEU $(\mathcal{C}_E, \mathcal{C}_{\hat{E}})$ gives the score of the corpus translation $\mathcal{C}_E$ given the corpus reference $\mathcal{C}_{\hat{E}}$.

The expected gain when defined at the corpus level is given by:

$$\mathcal{G} = \sum_{\mathcal{C}_E \in \mathcal{D}(\mathcal{C}_F)} P(\mathcal{C}_E|\mathcal{C}_F) \text{BLEU}(\mathcal{C}_E, \mathcal{C}_{\hat{E}}) \tag{7.14}$$

where we refer to a pair $\langle \mathcal{C}_E, \mathcal{C}_F \rangle$ drawn from the distribution as a *corpus sample*.

We can optimise the above corpus sampling objective using gradient descent. The gradient for the gain function in (7.14) with respect to a model parameter λ_m is given by:

$$\begin{aligned} \frac{\partial \mathcal{G}}{\partial \lambda_m} &= \sum_{\mathcal{C}_E \in \mathcal{D}(\mathcal{C}_F)} \text{BLEU}(\mathcal{C}_E, \mathcal{C}_{\hat{E}}) \frac{\partial P}{\partial \lambda_m} \\ \text{where } \frac{\partial P}{\partial \lambda_m} &= \left(h_m^{\mathcal{C}} - \mathbb{E}_{P(\mathcal{C}_E|\mathcal{C}_F)}[h_m^{\mathcal{C}}] \right) P(\mathcal{C}_E|\mathcal{C}_F) \end{aligned} \tag{7.15}$$

where $h_m^{\mathcal{C}}$ is the m-th component of a corpus sample feature vector, $h^{\mathcal{C}}$.

As in the case of sentence sampling, the gradient above can be rewritten as an expectation:

$$\frac{\partial \mathcal{G}}{\partial \lambda_m} = \mathbb{E}_{P(\mathcal{C}_E|\mathcal{C}_F)} \text{BLEU}(\mathcal{C}_E, \mathcal{C}_{\hat{E}}) \left(h_m^{\mathcal{C}} - \mathbb{E}_{P(\mathcal{C}_E|\mathcal{C}_F)}[h_m^{\mathcal{C}}] \right) \tag{7.16}$$

We use Monte Carlo estimation to approximate this gradient, which is otherwise intractable to compute, by drawing corpus samples from the distribution $P(\mathcal{C}_E|\mathcal{C}_F)$.

The feature values of a corpus sample are the average of the feature values of its constituting derivations and its BLEU score is computed based on the yield of its derivations.

Our phrase-based Gibbs sampler produces samples at the sentence level, whereas we require corpus samples. In order to generate n corpus samples, we use the following procedure. First, we draw a sequence of n samples $(\mathbf{e}_1, \mathbf{d}_1, \mathbf{f}), \ldots, (\mathbf{e}_n, \mathbf{d}_n, \mathbf{f})$ for each source sentence $\mathbf{f}$ in the corpus. We then generate corpus samples: the first corpus sample is obtained by iterating through the source sentences and taking the first sampled derivation for each sentence, then the second corpus sample is generated by taking the second sampled derivation for each sentence and so on until n corpus samples have been generated.

This procedure is simple and provides an efficient solution to the task of drawing corpus samples from $P(\mathcal{C}_E|\mathcal{C}_F)$ given that we only we have a mechanism to sample from $p(\mathbf{e}, \mathbf{d}|\mathbf{f})$. However, it introduces biases since any corpus sample is dependent on the order in which the sentence samples were generated in the first place. To eliminate this bias, we refine our corpus sampling procedure. For each source sentence, we first draw $m(m \geq n)$ sentence samples from which we obtain an empirical estimate of $p(\mathbf{e}, \mathbf{d}|\mathbf{f})$. We then resample n derivations from this empirical distribution. We can subsequently generate n corpus samples using the same procedure as above.

The corpus sampling procedure is illustrated in Figure 7.9. In practice, we do not have to store all the sentence samples from all sentences in order to perform the resampling step; we can just store the sufficient statistics of the samples (in our case, feature values and n-gram precision counts for computing BLEU).

The gradient in (7.15) is computed over the entire tuning set. In this case, the estimate of corpus BLEU is exact but training will be slow since we will effectively be doing full batch training. We can speed up optimisation by splitting the tuning corpus into sub-batches and updating the model weights using stochastic gradient descent each time a sub-batch is processed. Note that by using sub-batches, we are speeding up training but are potentially jeopardising the accuracy of the BLEU estimates. If the sub-batches are made too small, we are liable to obtain unreliable estimates of BLEU and consequently unreliable estimates of the gradient of the gain. In the extreme case where the sub-batch contains only sentence, if a hypothesised translation does not contain any of the high order n-grams in the reference translation(s), then its BLEU score (and its contribution to the gradient) will be zero, which is clearly undesirable. In

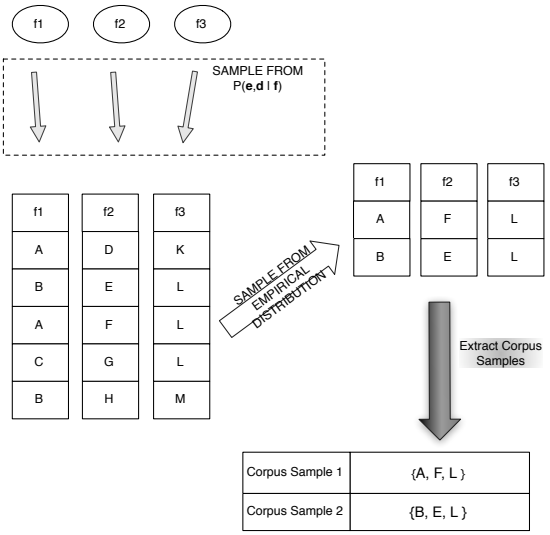


Figure 7.9: Example illustrating the extraction of 2 corpus samples for a corpus of source sentences f1, f2, f3. In the first step, we sample 5 derivations for each source sentence. We then resample 2 derivations from the empirical distributions of each source sentence. The n-th corpus sample is composed of the n-th resampled derivation for each of the source sentences.

section 7.4.2 we show that, as long as the sub-batches are large enough, we can obtain reliable estimates of BLEU.

7.4.1 Deterministic Annealing for Corpus Sampling

In the sentence sampling experiments, we found that deterministic annealing (DA) helps find weights which bring about improved translation performance. We now describe how to add DA to the corpus sampling objective.

When using deterministic annealing with sentence sampling, the entropy term is computed over the sampler estimate of $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ for each individual sentence. In corpus sampling, the distribution under consideration is $P(\mathcal{C}_E|\mathcal{C}_F)$; however, since the corpus sampling procedure invariably generates a set of samples which are all distinct, the Monte Carlo estimate of this latter distribution is almost always uniform. Therefore, any entropic prior defined over $P(\mathcal{C}_E|\mathcal{C}_F)$ will be of minimal use.

Instead, we define the entropic prior over the distribution $p(\mathbf{e},\mathbf{d}|\mathbf{f})$ of each of the input sentences in $\mathcal{C}_F$. The annealed sampling gain function is therefore:

$$\mathcal{G} = \sum_{\mathcal{C}_E \in \mathcal{D}(\mathcal{C}_F)} P(\mathcal{C}_E|\mathcal{C}_F)\text{BLEU}(\mathcal{C}_E, \mathcal{C}_{\hat{E}}) + \frac{T}{|\mathcal{C}_F|}\sum_{\mathbf{f}\in\mathcal{C}_F} H(p(\mathbf{e},\mathbf{d}|\mathbf{f})) \tag{7.17}$$

with gradient

$$\begin{aligned}
\frac{\partial \mathcal{G}}{\partial \lambda_m} &= \sum_{\mathcal{C}_E \in \mathcal{D}(\mathcal{C}_F)} \text{BLEU}(\mathcal{C}_E, \mathcal{C}_{\hat{E}}) \left(h_m^{\mathcal{C}} - \mathbb{E}_{P(\mathcal{C}_E|\mathcal{C}_F)}[h_m^{\mathcal{C}}] \right) P(\mathcal{C}_E|\mathcal{C}_F) \\
&- \frac{T}{|\mathcal{C}_F|} \sum_{\mathbf{f} \in \mathcal{C}_F} \sum_{\mathbf{d} \in D(\mathbf{f})} p(\mathbf{e},\mathbf{d}|\mathbf{f})(\log p(\mathbf{e},\mathbf{d}|\mathbf{f}) + 1) \frac{\partial p}{\partial \lambda_m} \\
\text{where} \quad & \frac{\partial p}{\partial \lambda_m} = h_m - \mathbb{E}_{p(\mathbf{e},\mathbf{d}|\mathbf{f})}[h_m] \text{ and } \mathbf{e} = Y(\mathbf{d})
\end{aligned}$$

Also, h is a *sentence*-level feature vector whereas $h^{\mathcal{C}}$ is a corpus level feature vector. Given this gradient, we can use SGD to optimise the annealed corpus sampling objective.

7.4.2 Corpus Sampling Experiments

We ran our corpus sampling experiments on the same language pairs and datasets used for the sentence sampling experiments in Section 7.3.4. We sampled using the same procedure as in sentence sampling and collected a total of 4,000 samples. We then resampled 2,000 corpus samples from the empirical distribution estimated from the first 4,000 samples.

Having ascertained during sentence sampling that deterministic annealing is beneficial, we focused our preliminary corpus sampling experiments on examining how the size of the batches used during corpus sampling affects translation performance, as measured by BLEU, on unseen data. To do so, we used batches of 200, 400 and 600 sentences. Note that the size of a batch corresponds to the number of sentences which form a corpus sample. The gradient of the objective function is computed over the 2,000 corpus samples drawn from a batch and subsequently a parameter update is made. With small batch sizes, we are able to compute gradients faster and therefore perform parameter updates more often. On the other hand, we should obtain a closer approximation to the true corpus BLEU score as the size of the batch increases. When the batch size is equal to the size of the tuning set, we are performing batch learning. However, this is too slow to be practical in our setup.

An important consideration when using annealing is the schedule at which the annealing temperature is cooled. In Section 7.3.4 we found that decaying the temperature at a slow rate enabled the optimiser to find weights which gave improved test time performance. In our experiments, we tried two different slow decay rates, 0.8 and

0.9. At each temperature setting, 10 iterations of gradient descent were performed. Training was stopped when T reached a floor temperature of 0.0001 or at the end of 48 hours of processing, whichever came first.

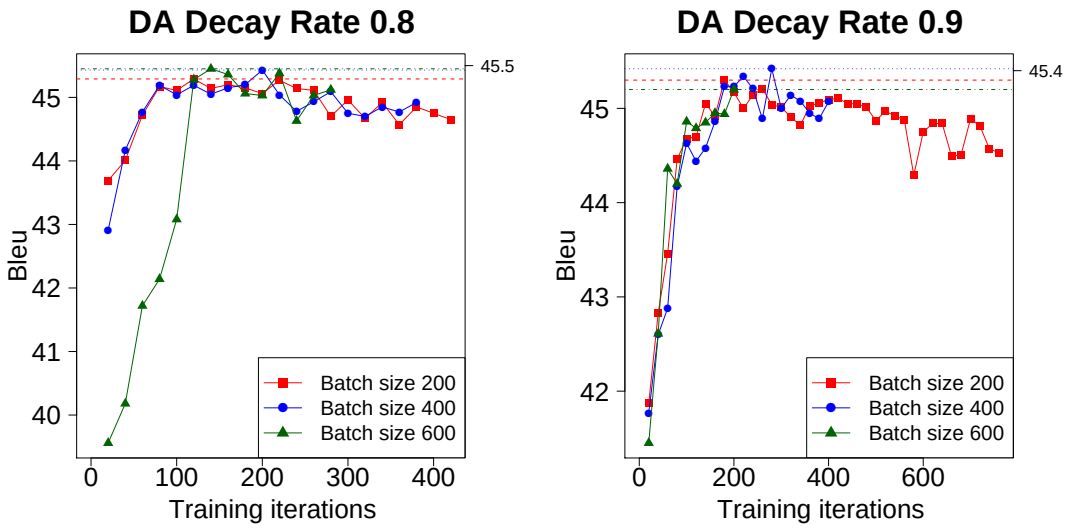


Figure 7.10: Held-out MBR decoding performance for Arabic-English training as a function of the number of training iterations. Best scores achieved are indicated by dotted line.

In Figure 7.10, we compare held-out MBR performance on Arabic-English, as measured by BLEU, against the number of training iterations for annealing decay rates of 0.8 and 0.9 respectively. For all batch sizes, when using a decay rate of 0.8, we are able to run optimisation till the annealing temperature reached the floor temperature. In this set of experiments, we observe that peak test time performance is impervious to the choice of batch size and that this peak performance is marginally better than the best BLEU score obtained during sentence sampling (45.4 vs 45.2).

With a decay rate of 0.9, while the annealing temperature reaches the floor temperature for the tuning experiments with batch sizes 200 and 400, the experiments do not complete within 48 hours of processing when using a batch size of 600. This is reflected by the fact that the held-out performance with batch size of 600 is lower compared to using smaller batch sizes. In contrast, when using batches of 200 and 400 sentences, the best performance on the held-out set matches the peak performances observed with the faster decay rate.

Figure 7.10 allows us to draw two conclusions. Firstly, it indicates that the corpus sampling algorithm is a reasonable method to optimise expected corpus BLEU even

though it approximates the exponential space of all possible translations of a given corpus with only a finite number of samples. Secondly, it shows that the corpus sampling algorithm is robust to the size of the batches used for gradient computation.

To have a better feel as to how corpus sampling compares to sentence sampling, we ran additional corpus sampling experiments on the Arabic-English, French-English and German-English language pairs. We compared deterministic annealing with a decay rate of 0.8 with not using any annealing. The results in Figure 7.10 motivated us to use a decay rate of 0.8 for all annealing experiments and to use batches of 400 sentences when tuning the Arabic-English model. For German-English and French-English, we used batches of 96 and 160 sentences respectively. We made this decision because running the experiments with larger batch sizes is too slow.

Test time conditions are identical to the sentence sampling ones and we measure max derivation, max translation and MBR performance on a held-out set after every 20 iterations of the learner. To account for the variance in results due to the stochasticity of the sampler, we ran training for each condition 5 different times and report the averaged results for French-English, German-English and Arabic-English in Figures 7.11, 7.12 and 7.13 respectively.

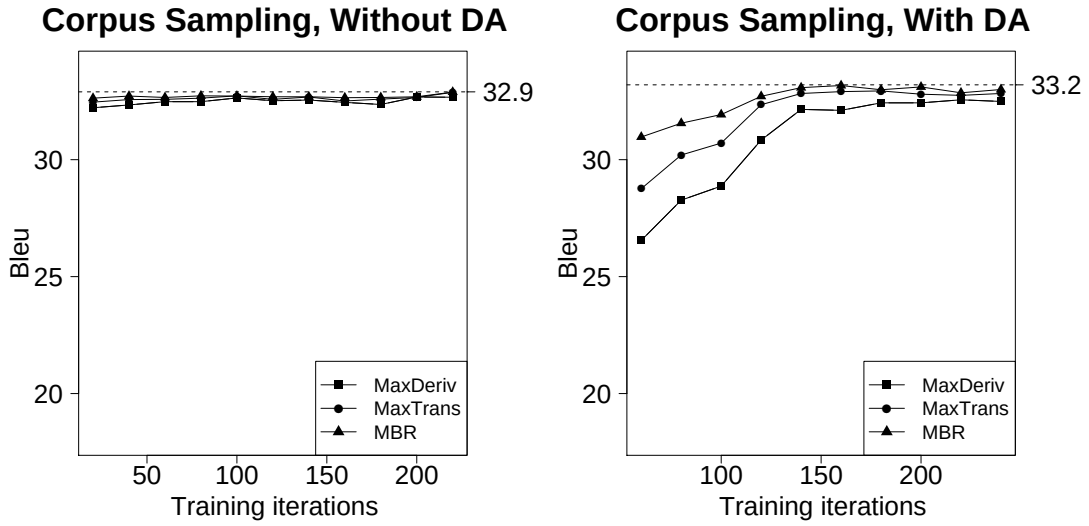


Figure 7.11: Held-out performance for French-English corpus sampling training averaged across 5 training runs. Best scores achieved are indicated by dotted line.

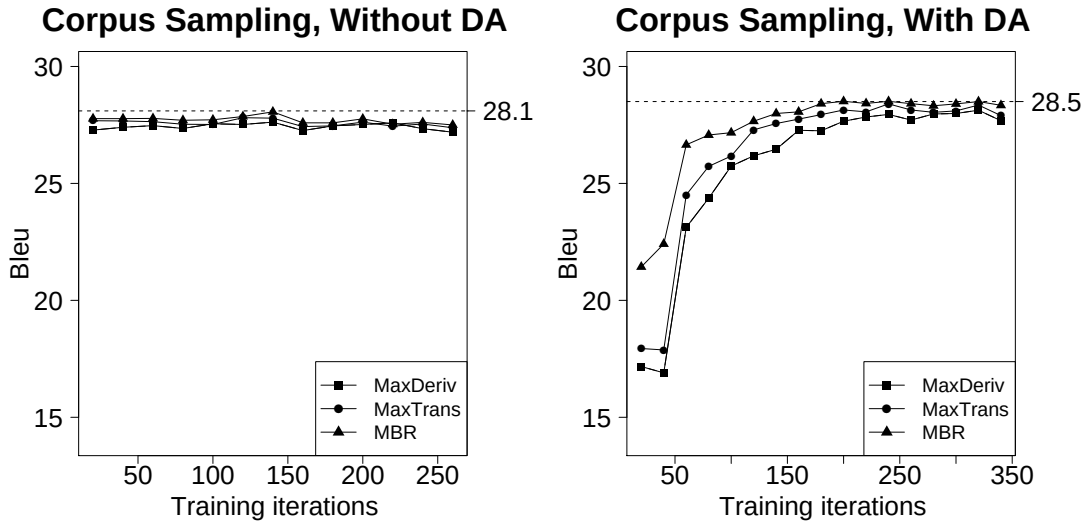


Figure 7.12: Held-out performance for German-English corpus sampling training averaged across 5 training runs. Best scores achieved are indicated by dotted line.

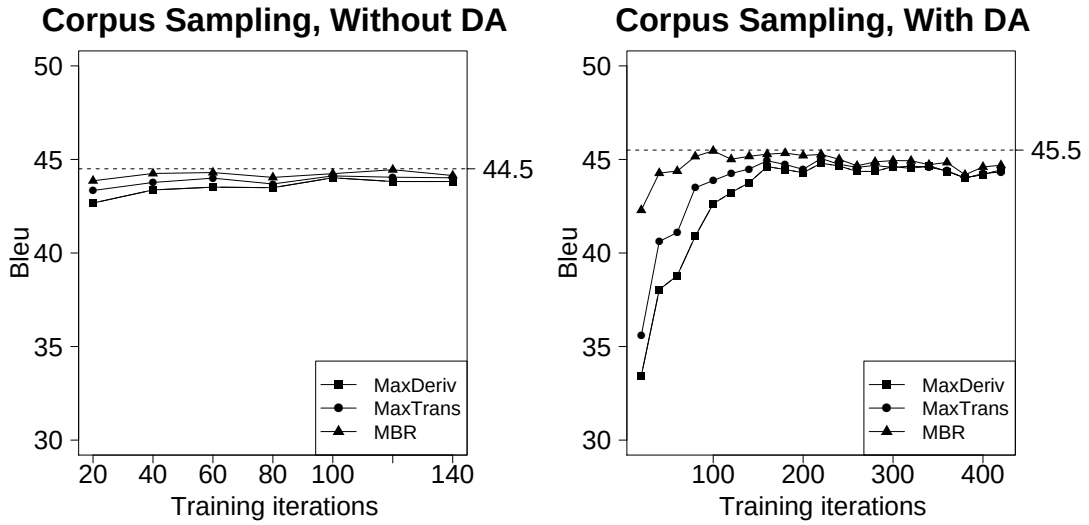


Figure 7.13: Held-out performance for Arabic-English corpus sampling training averaged across 5 training runs. Best scores achieved are indicated by dotted line.

We observe that the shape of the plotted curves for corpus sampling are very similar to the ones for sentence sampling (see Figures 7.6, 7.7 and 7.8) underlying the similarity in the objective functions. However, the peaks for the corpus sampling experiments are higher than for the sentence sampling ones. The benefits of corpus sampling are especially apparent when not using any annealing. In such cases we note that corpus sampling training outperforms sentence sampling by between 0.3 and 0.7% BLEU. When using annealing, the improvements are between 0.1 and 0.3% BLEU. These improvements are small but consistent across 5 different training runs, allowing us to conclude that corpus sampling does give improved translation performance in comparison to sentence sampling.

7.5 Beam Search vs Sampling

In the previous chapter, we proposed the use of sampling as an approximate inference solution for two intractable decision rules: max translation decoding and minimum Bayes risk decoding. However, when used in conjunction with weights optimised using MERT, we found that the sampler's performance lagged behind heuristic-based beam search solutions to the decision rules. We hypothesised that the sampler fared poorly because the MERT objective function, which optimises an unnormalised model for the single best derivation, produces model parameters unsuitable for use with the sampler.

A more suitable objective function is minimum risk training, which we introduced in this chapter. In the previous sections, we showed how to perform unbiased minimum risk training using sampling. Since the risk objective is a difficult one to optimise numerically, we availed of deterministic annealing to help with the optimisation. Finally, we described an algorithm that allowed us to optimise expected BLEU directly rather than an approximation thereof.

Armed with a suitable parameter estimation technique for the sampler, we are now in a position to assess how our sampling-based pipeline compares with the standard beam search pipeline. Let us recall what this latter pipeline consists of. First, the feature weights are optimised using MERT, an efficient line search based batch algorithm which maximises corpus BLEU. Then, a dynamic programming based beam search algorithm is used to find the most likely derivation in the model. The most likely derivation serves an approximation to the most likely string. Finally, either an n-best

list of derivations or even the whole lattice obtained at the end of beam search is used to approximate the evidence space required for MBR decoding.

7.5.1 A Brief Note on Statistical Significance

In this section, we will often be comparing the decoding performance of various competing systems, as measured by BLEU. It is not always clear when a difference in scores between two systems represents a significant difference in their output. A bootstrap resampling method to compute statistical confidence intervals for most automatic metrics, including BLEU, is described in Koehn (2004b). However, the statistical basis that bootstrapping rests on, for example a normal distribution of errors, is not founded for BLEU scores.

A bigger concern about statistical significance tests in general in SMT is that most models are trained using MERT, which is well known to be extremely unstable: different MERT runs produce different weights and these weights produce different translations. These translations can have large variability in their BLEU scores. One might end up with "significant" results purely as a result of the instability of MERT. If a statistical significance test cannot help us distinguish such cases, then it is of limited value.

In the remainder of this section, we mitigate the instability of MERT by running it 10 times on each language pair's tuning set. We use each of the resulting weight sets to decode the test data and report the minimum, maximum, mean and standard deviation of the BLEU scores across decoding runs. For models optimised by expected BLEU training, we run optimisation 5 times. Decoding with the sampler introduces further randomness; we account for it by running the sampler with each weight set 10 different times and report detailed statistics for the results.

7.5.2 Baseline

We use the beam search pipeline as implemented in Moses as our baseline. We performed experiments on the Arabic-English, French-English and German-English language pairs using MT02 as tuning set for Arabic-English and the appropriate DEV 2006 data as the tuning set for the other two language pairs. Each test set was decoded using the max derivation, *n*-best and lattice MBR decision rules. We used MT05 as the Arabic-English test set. For the European language pairs, we used TEST2008A as the in-domain and NEWSDEV2009B as the out-of-domain test sets. Full details on

these datasets, the baseline decoder and the features used in the model are given in Chapter 4.

To account for the instability of MERT training, we ran it 10 times and decoded the test sets with each of the 10 optimised weight sets. We present the best and the worst test set results along with the mean and the standard deviation (σ) of these results in Table 7.3. For *n*-best and lattice MBR decoding, we optimised for the scaling factor using a grid-search on held-out data. For lattice MBR decoding, we optimised the lattice density and set the p and r parameters as per Tromble et al. (2008). For both *n*-best and lattice MBR decoding, the hypothesis set was composed of the top 1000 unique translations produced by the beam search decoder, and the same 1000 translations were used as evidence set for *n*-best MBR.

As Table 7.3 shows, translation results using MERT optimised weights vary significantly from one tuning run to the other, with results varying from a range of 0.3% BLEU (French-English in-domain data) to 1.3% BLEU (German-English out-of-domain data) when performing max derivation decoding.

We compare the max derivation decision rule to MBR decoding. For the Arabic-English test set, MBR decoding brings about a significant improvement in translation performance. *N*-best MBR does better than max derivation and lattice MBR improves upon *n*-best MBR. This result is consistent with the results in the lattice MBR literature (Tromble et al., 2008; Kumar et al., 2009). However, MBR decoding does not help much when translating the in-domain European test sets. This is because the significant overlap between the *n*-grams in the training data and those in the test data produces spiky distributions. As we discussed in Section 7.3.4.2, for such distributions, the benefits of MBR decoding are likely to be mitigated.

MBR decoding *does* help in the case of out-of-domain data. While the improvement in French to English is minimal and only observed when using *n*-best MBR, the more powerful lattice MBR algorithm brings about a significant increase in BLEU on the German-English data set. In particular, the MERT run which produced a BLEU score of 14.9 on max derivation decoding and 15.0 on *n*-best MBR gives a score of 16.0 with lattice MBR: lattice MBR is able to find a consensus solution of significantly higher quality than the 1-best solution under the model. Lattice MBR generally peforms well; however there is a small drop in performance compared to *n*-best MBR on the French-English data sets. This suggests that the hyper-parameters of the lattice MBR algorithm, which we tuned on the held-out set, did not generalise to the unseen test sets.

	Max Derivation				*N*-best MBR				Lattice MBR			
	min	max	mean	σ	min	max	mean	σ	min	max	mean	σ
AR-EN	43.7	44.3	44.0	0.17	44.2	44.5	44.4	0.13	44.4	45.0	**44.6**	0.22
FR-EN In	33.1	33.4	33.3	0.10	33.2	33.6	**33.4**	0.12	33.1	33.4	33.3	0.12
FR-EN Out	19.1	19.6	19.4	0.18	19.3	19.7	**19.5**	0.12	19.2	19.6	19.4	0.16
DE-EN In	27.6	27.9	**27.8**	0.10	27.6	27.9	27.7	0.10	27.6	27.9	27.7	0.10
DE-EN Out	14.9	16.2	15.7	0.33	15.0	16.3	15.7	0.33	16.0	16.4	**16.1**	0.24

Table 7.3: Baseline results - MERT trained models decoded using max derivation, nbest MBR and lattice MBR. MERT was run 10 times for each language pair. We report minimum, maximum, mean and standard deviation of test set BLEU scores across the 10 runs. Best performance on each data set is in bold.

7.5.3 Sampling

Having verified the benefits of deterministic annealing from the experimental results in Section 7.3.4 and 7.4.2, Table 7.4 compares annealed sentence sampling with annealed corpus sampling on our five test sets. To account for sampler variance during both training and decoding, we average scores across 50 runs; 10 runs each using the best weight set from 5 training runs. We run training until the cooling temperature reaches the floor temperature or training has gone on for 48 hours, whichever comes first. During training, the current weight settings are periodically output after every 20 iterations and are then used to decode the held-out set. The weight set which gives the best translation performance on the held-out set is considered the best weight set and is ultimately used for decoding the test set.

The results in Table 7.4 confirm recent findings of (Blunsom et al., 2008; Arun et al., 2009) that max translation improves over max derivation decoding for models trained to account for multiple derivations. We also see that MBR performs best on all test sets establishing that the sampler is best used as an MBR decoder. For MBR decoding, we also report the minimum and maximum scores across the 50 decoding runs, along with the standard deviation of the scores.

The trend observed on held-out sets carry over to the test sets. Table 7.4 shows that corpus sampling does at least as well as sentence sampling on four out of five datasets, with small but consistent improvements on three of them. These results show that corpus sampling is a suitable algorithm for performing expected BLEU training.

	Sentence sampling						Corpus sampling					
	MaxD	MaxT	MBR				MaxD	MaxT	MBR			
Test set	Mean	Mean	Min	Max	Mean	σ	Mean	Mean	Min	Max	Mean	σ
AR-EN	43.0	43.9	44.4	44.9	**44.6**	0.11	41.8	43.1	44.2	44.8	44.5	0.14
FR-EN In	32.2	32.8	32.6	33.2	32.9	0.16	32.5	33.0	33.1	33.3	**33.2**	0.06
FR-EN Out	19.0	19.6	19.5	19.9	19.7	0.09	19.2	19.6	19.7	19.9	**19.8**	0.05
DE-EN In	26.9	27.3	27.4	27.8	27.6	0.07	27.2	27.6	27.6	28.0	**27.8**	0.11
DE-EN Out	15.8	16.3	16.5	16.7	**16.6**	0.07	15.9	16.3	16.4	16.8	**16.6**	0.12

Table 7.4: Comparison of bleu scores for annealed sentence sampling and corpus sampling using 3 different decision rules. The scores are the average across 50 runs; 10 decoding runs each using the best weight set from 5 training runs. For MBR decoding, max, min, mean and standard deviation (σ) are also included. Numbers in bold represent best performance for each data set.

7.5.4 Comparison

We now compare our strongest sampling pipeline, annealed corpus sampling training followed by MBR decoding, with the best results obtained using MERT optimised Moses. The results obtained with MERT correspond to the figures shown in bold in Table 7.3. As before, we average sampling scores across 50 runs; 10 decoding runs each using the best weight set from 5 training runs. The Moses results are obtained by averaging scores from 10 different MERT training runs. Results are shown in Table 7.5.

The sampling pipeline markedly outperforms beam search methods on out-of-domain test sets, with an improvement of 0.3% BLEU in French-English and 0.5% BLEU in German-English. However, there is no improvement and in some cases a slight deterioration when translating in-domain data. We delay further discussion of these results to the next section.

An interesting thing to note is that the sampler results are significantly more stable than those obtained with MERT weights on 4 out of 5 test sets whereas on the 5th test set, both methods give results of almost similar stability. The instability of MERT is a topic of sufficient concern to have generated substantial research interest. The work of Foster and Kuhn (2009) is of special relevance since it performs a thorough investigation of MERT's stability (or lack thereof), finding that test set BLEU scores can vary by 1 % across 10 MERT runs. However, they were unable to come up with an

	MERT/Moses		Sampler	
Test set	Best	σ	MBR	σ
AR-EN MT05	**44.6** (*L*-MBR)	0.22	44.5	0.14
FR-EN In	**33.4** (*N*-MBR)	0.12	33.2	0.06
FR-EN Out	19.5 (*N*-MBR)	0.12	**19.8**	0.05
DE-EN In	**27.8** (MaxD)	0.10	**27.8**	0.11
DE-EN Out	16.1 (*L*-MBR)	0.24	**16.6**	0.12

Table 7.5: Results comparing MERT/Moses pipeline with unified sampler pipeline. Sampler uses corpus sampling during training and MBR decoding at test time. Moses results are averaged across decoding runs using weights from 10 MERT runs and sampler results are averaged across 10 decoding runs for each of 5 different training runs. We report BLEU scores and standard deviation (σ). For Moses results, we indicate the decision rule used. *L*-MBR: Lattice MBR, *N*-MBR: *n*-best MBR and MaxD: max derivation. Scores in bold indicate best performances for the data set.

effective way of reducing this variance. The results in Table 7.5 suggest that gradient-based optimisation of the minimum risk training objective is a parameter estimation technique with lower variance than MERT. We attribute the improved stability to the more powerful optimisation algorithm used by the sampler: the information provided by the gradient steers the model towards better weights. MERT, on the other hand, optimises one feature at a time using line search and therefore does not explore the full feature space as thoroughly.

We next compare the decoding speed for the different flavours of MBR decoding algorithms. The average decoding times are 10 seconds per sentence for Moses *n*-best MBR, 40 seconds per sentence for Moses lattice MBR and 180 seconds per sentence for sampling MBR. Sampling the phrase-based model is expensive, meaning that lattice MBR is still faster (around 4 times) to run than sampling MBR. However, due to the *unified* nature of the training and decoding criterion in our approach, the minimum risk trained weights can be plugged *directly* into the sampler MBR decoder, whereas lattice MBR requires an additional expensive step of tuning the model hyper-parameters.

Some example translations drawn randomly from French-English and German-English out-of-domain experiments, comparing reference translations, the Moses outputs and the sampler outputs are shown in Table 7.6 and 7.7 respectively.

	Output
R	the option to buy 18.49 percent from investment company vatas had already been arranged back in august .
M	the right to buy the per cent of society 18,49 investment had already been decided in vatas august .
S	the right to buy the 18,49 per cent of society investment vatas had already been decided in august .
R	the sports confederation hamburg (hsb) regretted von beust ' s comments .
M	the sports federation in hamburg ( ) the statements of hsb regrettait beust .
S	the sports federation of hamburg (hsb) regrettait declarations of beust .
R	the bankers got $ 7.3 million while fastow , kopper and others skimmed about $ 12.3 million , according to the indictment .
M	according to the accusations , the bankers have obtained $ 7.3 while , and the other fastow kopper would have won $ 12.3 million .
S	according to the accusations , the bankers have obtained $ 7.3 while fastow , kopper and others would have won $ 12.3 million .

Table 7.6: Comparison of reference translation (R), Moses MBR output (M) and Sampler MBR output (S) on 3 randomly chosen sentences from the French-English NEWSDEV2009B test set.

	Output
R	the bank was merely holding the shares for a third party , rumours in the financial market suggested .
M	is the only for the geldhaus shares a third party , it was said on financial market .
S	the geldhaus think the shares only for a third party , it was said on financial market .
R	the in-the-flesh resurrection of harmonia takes place at the berlin house of cultures of the world .
M	the physical resurrection of the house will take place at the berlin harmonia cultures of the world .
S	the physical resurrection of harmonia will take place at the berlin house of the cultures of the world .
R	the british trio was arrested three months later .
M	the british trio was three months later , arrested .
S	the british trio was arrested three months later .

Table 7.7: Comparison of reference translation (R), Moses MBR output (M) and Sampler MBR output (S) on 3 randomly chosen sentences from the German-English NEWSDEV2009B test set.

7.5.5 Discussion

In Table 7.5, we compared our best sampling pipeline consisting of sampling-based minimum risk training and sampling MBR with a beam search pipeline of MERT training followed by lattice-based MBR decoding[2]. We found that the sampling pipeline gave marked improvements on out-of-domain test sets but a slight deterioration on in-domain test sets. Since in most realistic applications of MT, the test data comes from a domain different to that of the training data, these results are particular encouraging.

We hypothesise two possible reasons for these results. Firstly, we speculate that MERT overfits in-domain data. Generally speaking, overfitting tends to be an issue in models where the features capture very specific characteristics of the input and/or output. In our model, the features are mostly log probability scores and are few; thus, overfitting seems unlikely.

However, note that one of the features in the model is the phrase penalty feature. When the phrase penalty feature weight is positive, the model prefers translations made of a large number of short phrases whereas a small number of long phrases is preferred if the feature weight is negative. We noticed that in 15 out of the 20 MERT optimised weights obtained from the FR-EN and DE-EN tuning runs, the weight for this feature is **negative**. Intuitively this makes sense: MERT optimises for single-best derivations and generally, in the in-domain data used for MERT tuning, the 1-best derivations tend to be hypotheses which use a small number of phrases. By assigning a negative value to the feature, a model which favours the use of a small number of phrases during translation is learnt.

Figure 7.3 evidenced that the sampler can obtain a very accurate estimate of the expectation of the phrase penalty feature; therefore, we expect that minimum risk training with the sampler should provide good weights for this feature. In fact, we found that expected BLEU training **always** assigned **positive** weights to the phrase penalty feature.

Recall that the minimum risk training criterion optimises a softer objective than MERT: instead of moving the probability mass towards the 1-best derivation, each derivation in the model is assigned a share of the probability mass proportional to its gain with respect to the reference translations. A positive weight for the phrase penalty feature suggests that while the 1-best derivation uses a small number of phrases, there are many other high quality derivations which use a larger number of phrases.

[2]We consider n-best MBR as a specialised variant of lattice MBR.

By favouring the use of a large number of phrases during translation, minimum risk training ultimately increases the model's ability to generalise. This could explain the improved translation performance on out-of-domain data since out-of-domain translation tends to require the use of a larger number of short phrases.

An alternate explanation is that, compared to in-domain test sets where most of the *n*-grams have already been seen in the training corpus, the majority of out-of-domain *n*-grams have rarely or not been seen. This has the consequence that there is a high level of uncertainty during translation. As a result, rather than relying on the 1-best derivation of the model, a search for a consensus translation, such as the MBR translation, finds a solution of better quality. Of course, MBR decoding of out-of-domain data was tried for MERT trained models too. However, recall that MERT weights need to be rescaled prior to MBR decoding: there is no guarantee that the scaling factor used is optimal for decoding out-of-domain data.

7.5.5.1 Sampler Decoding: MERT vs Minimum Risk Training

In order to tease apart the effects of minimum risk training and MBR decoding, we ran additional decoding experiments where for the sake of completeness, we decoded all 5 of our test sets. We begin by keeping the decoder constant and varying the feature weights. In Table 7.8, we compare using rescaled MERT trained weights with expected BLEU optimised weights while using the sampler as decoder. We used weights from 5 MERT and from 5 expected BLEU runs, running the decoder 10 times with each weights. We report the mean sampling MBR BLEU scores across the 50 decoding runs for each parameter estimation technique.

On out-of-domain test sets, the expected BLEU weights give substantial improvements over MERT weights, thus adding credence to the hypothesis that minimum risk training generalises better. However, since improvements are also obtained on in-domain test sets, the most apt conclusion to draw is that MERT trained weights are simply not suitable for use with the sampler.

7.5.5.2 Moses Decoding: MERT vs Minimum Risk Training

Next, we compare MERT and expected BLEU weights using Moses as decoder. We perform max derivation, *n*-best MBR and lattice MBR decoding. For *n*-best and lattice MBR decoding with MERT weights, we optimised for the scaling factor using a grid-search on held-out data. For similar experiments with expected BLEU weights, no

Test set	MERT	Expected BLEU
AR-EN	44.0	44.5
FR-EN In	33.2	33.2
FR-EN Out	19.0	19.8
DE-EN In	27.5	27.8
DE-EN Out	16.3	16.6

Table 7.8: Comparing sampling MBR BLEU scores when run with MERT optimised and expected BLEU trained weights. Scores are averaged across 10 decoding runs for each of 5 different training runs.

rescaling is required since the weights are already scaled appropriately. Prior to lattice MBR decoding, we optimised the lattice density and set the p and r parameters as per (Tromble et al., 2008). In both n-best and lattice MBR decoding experiments, the hypothesis set consisted of the top 1000 unique translations produced by the beam search decoder. The same 1000 translations were used as evidence set for n-best MBR. We used weights from 5 different MERT and from 5 different expected BLEU optimisation runs.

The results are shown in Table 7.9. The improvement in out-of-domain results across all decision rules when using expected BLEU weights suggests that minimum risk training is in fact a more appropriate objective function to be optimised when generalisation is important. On in-domain data, the performance of max derivation is generally better using MERT weights confirming our belief that MERT weights, while lacking generalisation, do well when decoding in-domain data.

Except for in-domain French-English, the best BLEU scores are obtained when using lattice MBR decoding in conjunction with expected BLEU weights. Given that lattice MBR with MERT weights fails to produce similar improvements, it seems plausible that the minimum risk training criterion might be a better fit for lattice MBR than the MERT criterion. An analogous observation is made by Pauls et al. (2009) who find that a training objective function which accounts for the entire distribution outperforms MERT when used with a lattice MBR like consensus decoding algorithm.

An alternate explanation as to why lattice MBR with MERT weights underperforms is that a weight rescaling step is required prior to decoding. The rescaling factor is usually optimised on a held-out set. There is no guarantee that the rescaled weights generalise to unseen data. On the other hand, the minimum risk training algorithm as

Training	MERT						Expected BLEU					
Decoding	MaxD		N-MBR		L-MBR		MaxD		N-MBR		L-MBR	
	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ
AR-EN	44.0	0.17	44.4	0.13	44.6	0.22	44.1	0.08	44.4	0.06	**44.7**	0.15
FR-EN In	33.3	0.10	**33.4**	0.12	33.3	0.12	33.1	0.04	33.1	0.04	33.3	0.05
FR-EN Out	19.4	0.18	19.5	0.12	19.4	0.16	19.5	0.10	19.7	0.12	**19.8**	0.08
DE-EN In	27.8	0.10	27.7	0.10	27.7	0.10	27.7	0.10	27.8	0.09	**28.0**	0.08
DE-EN Out	15.7	0.33	15.7	0.33	16.1	0.24	16.2	0.20	16.4	0.18	**16.6**	0.12

Table 7.9: Moses under max derivation (MaxD), N-best MBR (N-MBR) and lattice MBR (L-MBR) decoding regimes with MERT and expected BLEU trained weights. Results are averaged across decoding runs using weights from 5 MERT and 5 expected BLEU training runs.We report mean BLEU scores and standard deviation (σ). Scores in bold are best performances for the data set.

implemented with the sampler is probabilistic and scales the weights automatically to the appropriate scale.

At the beginning of this section, we hypothesised that the improvement obtained on out-of-domain data when using the sampling pipeline can be due to either better parameter estimation or due to the use of the MBR decision rule. From Table 7.9, we see that while out-of-domain 1-best decoding results with expected BLEU trained weights are better than the corresponding results with MERT weights, the combination of MBR decoding and expected BLEU weights does even better: However, as Table 7.9 shows, provided that the weights have been optimised so that the distribution over derivations in the model is appropriately shaped, the MBR decision rule is generally beneficial both for in-domain and out-of-domain data.

Similar to the figures in Table 7.5, the results with sampler trained expected BLEU weights are more stable than with MERT weights, underlying the usefulness of gradient-based optimisation. The stability of the results can be seen by small standard deviation across expected BLEU decoding results compared to results with MERT weights.

7.5.5.3 Minimum Risk Training: Sampler vs Moses

We have so far seen that both the sampler and Moses benefit from minimum risk trained weights. In a final set of experiments, we keep the weights constant and instead compare two alternate MBR decoding algorithms: sampling-based MBR versus lattice

	Sampling MBR				Lattice MBR			
Test Set	Min	Max	Mean	σ	Min	Max	Mean	σ
AR-EN	44.2	44.8	44.5	0.14	44.5	44.9	**44.7**	0.15
FR-EN In	33.1	33.3	33.2	0.06	33.2	33.3	**33.3**	0.05
FR-EN Out	19.7	19.9	**19.8**	0.05	19.6	19.8	**19.8**	0.08
DE-EN In	27.6	28.0	27.8	0.11	27.9	28.1	**28.0**	0.08
DE-EN Out	16.4	16.8	**16.6**	0.12	16.5	16.8	**16.6**	0.12

Table 7.10: Comparison of Sampling MBR and Lattice MBR decoding as measured by BLEU. For sampling MBR, scores are the average across 50 runs; 10 decoding runs each using the best weight set from 5 training runs. For lattice MBR, scores are averaged across 1 decoding run for each of the 5 training runs. We report minimum, maximum and the standard deviation of the scores across all decoding runs. Best averaged results are indicated in bold.

MBR. Since MERT weights are unsuitable for use with the sampler, we use expected BLEU trained weights.

Lattice MBR and sampling MBR are both solutions to the same problem, which is to accurately estimate the risk of each solution in the exponential translation search space. Lattice MBR computes this risk by using an efficient dynamic program which involves all the derivations in the pruned search space whereas sampling MBR does so by using small number of derivations sampled from the posterior distribution of derivations in the search space.

Table 7.10 presents results from experiments comparing the two decoding techniques. We used the weights from 5 different expected BLEU optimisation runs. Lattice MBR is a deterministic algorithm: given a weight vector, if the hyperparameters of the algorithm stay constant, then the output will always be the same. This is not the case for sampling MBR; therefore, we ran 10 decoding runs with each of the 5 feature weights. We report the minimum, the maximum and the standard deviation of the scores across all decoding runs. The evidence space in lattice MBR naturally consists of the entire lattice whereas in sampling MBR it is made of 10,000 derivations sampled from the distribution. Both lattice and sampling MBR use hypothesis spaces containing the 1,000 most probable translations: the former does so by extracting the top 1,000 distinct derivations from the lattice while the latter uses the 1,000 most probable translations in the evidence space as per their Monte Carlo estimate of $p(\mathbf{e}|\mathbf{f})$.

Comparing the mean MBR scores, we observe that on out-of-domain data, both sampling MBR and lattice MBR give identical results. On the other hand, lattice MBR gives better results on in-domain data; however, we found that sampling MBR matches lattice MBR on all test sets when the evidence space is increased from 10,000 to 100,000 derivations.

These results also serve as a sanity check: given the same feature weights, both algorithms converge to the same result. Note that lattice MBR uses as evidence set the whole lattice, consisting of roughly 10^{70} derivations (see Table 7.1), whereas the evidence set in sampling MBR typically consists of 10,000 derivations sampled from the distribution. Nevertheless, the sampler's limited evidence set is enough to give a good estimate of the probability distribution and therefore a good estimate of the expected BLEU of the translations in the model.

Still, sampling MBR is around 4 times slower than lattice MBR. The results in Table 7.10 suggest that an alternate efficient and accurate way of doing MBR decoding, which maintains the extended view of the distribution provided by sampling MBR, is to first train the feature weights using sampling-based minimum risk training and then perform lattice MBR. This procedure has the added advantage of eschewing the expensive step of tuning the scaling factor of the feature weights, thus reducing the number of hyper-parameters in the algorithm.

7.6 Summary

In this chapter, we used the sampler introduced in Chapter 5 to approximate in a principled manner the intractable problem of minimum risk training. The need for minimum risk training was motivated by the fact that merely applying sampling MBR to MERT optimised models failed to give satisfactory performance. We attributed this result to the mismatch between the MERT training objective which aims to move the probability mass of the distribution towards the *single-best* derivation and the MBR decoding objective which reasons in terms of translations rather than derivations and which aims to find the translation that minimises the expected loss of the model. MBR therefore can be seen as seeking a consensus solution whereas MERT seeks to ensure that the best solution is ranked first, disregarding all other solutions in the search space. The minimum risk training objective, on the other hand, has the same form as the MBR decoding objective, making this consistency of objective across the translation pipeline very appealing.

A key component of minimum risk training as well as many other parameter estimation techniques for log-linear models is the computation of the expectation of the feature values under the distribution. One of the contributions of this chapter is a detailed comparison of various methods used for approximating this expectation. We showed empirically that, if the number of derivations used for the calculation of the expectation is kept constant, a closer approximation to the true expectation is obtained when the derivations are sampled from the entire distribution rather than just from the vicinity of its mode. This result suggests that some of the prior work in MT where the feature expectation has been computed over *n*-best lists might benefit from the inclusion of low probability solutions too.

While sampling provides a close approximation to the true expectation, we found that the closest approximation is obtained when computing the expectation over a packed representation of all the unpruned derivations in the model. This representation is a by-product of dynamic programming based decoding algorithms and typically compactly encodes around 10^{70} derivations for medium sized sentences whereas we usually sample 10^4 derivations. This result indicates that an avenue worth pursuing in the future is to store a packed representation of the derivations encountered during sampling which can subsequently be used for the calculation of expectations.

Our model's performance is evaluated with the standard MT metric, BLEU, which is defined at the corpus level. We experimented with two formulations of minimum risk training. In the first place, we defined the objective at the sentence level (*sentence sampling*) and the objective was aggregated over all input sentences. In this formulation, we approximated BLEU with a sentence-level variant. When aggregated over the corpus, sentence-level BLEU is known not to correlate well with corpus-level BLEU. Instead, we proposed a novel sampling algorithm which allowed us to draw samples at the corpus level (*corpus sampling*) and therefore, unlike approximations used by previous methods, directly use corpus BLEU in our objective function. The training objective is non-convex so we used *deterministic annealing* to smooth the objective thus decreasing the chances of the optimiser to get trapped in local optima.

When using the sampler with MERT trained weights in Chapter 6, we had found that max translation and MBR decoding only marginally outperformed max derivation decoding. With minimum risk trained models though, the experimental results were consistently in keeping with our initial intuition: max translation clearly outperformed max derivation and MBR did best, thus showing the benefits of marginalising over derivations and of taking into account the whole distribution.

The unified sampling pipeline of minimum risk training and MBR decoding was found to do particularly well on out domain translation when compared with the typical SMT pipeline of MERT tuning followed by max derivation or MBR decoding. Further analysis showed that this is because the minimum risk objective generalises better than the 1-best objective of MERT.

Another benefit of our training regime was that the learnt parameters were found to be more stable that MERT optimised parameters, indicating that the gradient-based optimisation technique made possible by the probabilistic formulation of the training objective explores the parameter space in a more systematic manner than the line-search used by MERT.

The overall best decoding results were obtained by running lattice MBR on expected BLEU trained weights, suggesting that minimum risk training criterion is in fact a better fit for MBR than MERT. This approach has the additional benefit that it obviates the need for feature weight rescaling, a step which is required when using MERT weights.

An additional benefit of minimum risk training, not explored in this thesis, is that unlike MERT training, it can scale to a large number of features. We leave this for future work.

Chapter 8

Conclusions and Future Directions

8.1 Summary

Recent advances in statistical machine translation have used beam search methods for approximate inference within probabilistic translation models. Despite their success, these methods compromise the probabilistic interpretation of the underlying model thus limiting the application of probabilistically defined decision rules during training and decoding. As an alternative, this thesis has proposed a novel Monte Carlo sampling framework for theoretically sound approximate probabilistic inference in these models.

The main contributions of the thesis are as follows:

- We developed a Gibbs sampler for sampling derivations from the distribution defined by a phrase-based machine translation model and showed that it effectively explores this distribution. Since the state space, i.e. the space of derivations allowed by the model, does not easily decompose into a graphical model, textbook Gibbs sampling could not be applied. Instead, we used a block Gibbs sampling approach in which a subset of variables in the model are sampled conditioned on the remaining ones. At each iteration of Gibbs sampling, the subset of variables to be sampled are selected by so called Gibbs operators. Each operator defines a set of variables neighbouring the current variable. The Gibbs operators were defined such that the entire state space could be explored by the successive application of each operator at each position in the source sentence.

  Having defined the Gibbs sampler, we demonstrated empirically that in most cases the sampler convergences to the true distribution in the sampling limit thus evidencing that the sampler is drawing samples from the distribution of interest.

This convergence is obtained irrespective of the manner in which the sampler is initialised, showing that the operators allow the sampler to mix well.

Occasionally, the sampler was found to get stuck in local optima. These cases served as stark reminders that the distribution being sampled from is highly multimodal.

- Having ascertained that the Gibbs sampler is able to efficiently explore the state space of derivations, we applied the sampling framework to a variety of inference problems. The sampler can be used to provide an unbiased estimate of any expression that can be written in terms of an expectation of a function defined over the probability distribution being sampled from. We showed that the sampler can therefore be used to obtain estimates of the probability of a derivation, the probability of a translation and the expected gain of a translation. The solutions which maximise each of these terms are the max derivation, max translation and MBR solutions respectively. The sampler thus provides a tractable way for implementing these decoding decision rules.

  The sampler can also be used for minimum risk training, a parameter estimation technique which optimises a probabilistically defined objective function. Since this objective takes into account the entire distribution and has the exact same form as the MBR decision rule used at test time, we considered it to be particularly well suited to the sampler. A key term required during gradient-based optimisation of this training regime is the expectation of the values of the features in the model. Computing this term exactly is intractable so was approximated with sampling. We found that the mix of high probability and low probability derivations obtained through sampling provided a more accurate estimate of the feature expectations than merely using the high probability derivations contained in an n-best list.

  Finally, we compared the novel pipeline of sampling-based minimum risk training and decoding with the standard pipeline of MERT training and dynamic programming-based beam decoding. We found that the sampling pipeline can improve upon the latter. The improvements were noted on out-of-domain test sets, suggesting that minimum risk training generalises better than MERT. We also found that the minimum risk trained feature weights produced more consistent results than MERT trained weights.

The best overall performance was generally obtained when combining minimum risk trained weights with lattice MBR, a dynamic programming based decoding algorithm which estimates the risk of every translation using the entire pruned lattice of a first pass beam decoder. This result suggests that minimum risk training is a better fit for lattice MBR than MERT training. Sampling MBR matched lattice MBR on some data sets and was found to match it on all of them when the sample set was made larger.

In summary, this thesis demonstrates the potential of the proposed sampling-based framework as an alternative to dynamic programming based beam search algorithms for both training and decoding in phrase-based translation models. For these tasks, sampling allies the simplicity of *n*-best list approaches with the extended view of the distribution that lattice-based approaches benefit from.

Additionally, by using Monte Carlo techniques we avoid the biases associated with beam pruning. In doing so, we provide a further tool to the translation community that we envision will allow the development and analysis of increasing theoretically well motivated techniques.

However, one drawback of our framework is its slow runtime. This is because it is computationally expensive to draw samples from the distribution. In the next section, we suggest ways to improve sampling efficiency as well as outline some future research possibilities.

8.2 Future Directions

8.2.1 Sampler Efficiency

We have shown in this thesis that sampling is a practical approximate inference tool for SMT models. However, a concern with the proposed sampling framework is its slow runtime. This has limited the number of samples we can use for efficient training and decoding. Both tasks should improve with more samples; indeed, in Section 7.5.5.3 we observed that sampling MBR can match the decoding performance of lattice MBR by using a larger evidence set than currently used.

Consequently, an obvious area of future work is to speed up sampling. A possible solution can be found by revisiting the algorithm of the *scan* function which was detailed in Algorithm 5.1. The algorithm discards a large number of intermediate samples in order to reduce the correlation between successive samples. One way to

speed up sampling is to retain some of these discarded samples, e.g. we could collect a sample after each operator scan (lines 5, 17 and 20 in Algorithm 5.1) rather than only at the end of all of their scans. This is a straightforward way to get a three-folds speed up in sampling although the impact on sampling performance by having more highly correlated samples needs to be determined.

Another potential way to speed up the algorithm is to replace the sequential scanning procedure whereby each operator systematically traverses the input sentence from left to right with a *random scan* procedure (Levine and Casella, 2006). In random scanning, the operator to sample with and the source position to sample at are themselves sampled from a pre-specified probability distribution over operators and source positions; successive samples are therefore less likely to be correlated. To ensure that autocorrelation is minimised, every m-th sample can be collected. A drawback of this scanning procedure is that it introduces additional hyper-parameters, in the form of the distribution over operators and source positions, to the scanning algorithm.

A common way of improving the performance of an SMT system is by using large order language models (LMs). In the experiments in this thesis, we used trigram LMs but it is not uncommon in large scale tasks to see models using 5-gram, 6-gram or even 7-gram LMs. The use of high order LMs has an impact on the dynamic program used for hypothesis recombination during beam decoding. As the n-gram order increases, fewer recombinations take place so more aggressive pruning is required to maintain an acceptable decoding speed. The n-gram order of the LM used also impacts sampling speed. In fact, the complexity of our sampling algorithm is linear in the n-gram order.

We give a broad sketch of a Metropolis-Hastings approach for reducing this complexity to constant time. In this approach, the block sampling steps are performed using a low order language model. The sample obtained after a full scan is then accepted or rejected based on a Metropolis-Hastings acceptance test. In this Metropolis-Hasting step[1], the proposal distribution is the distribution with the low order LM and the target distribution is the desired high order language model. If the distribution of derivations using low order n-grams is close to that using high order n-grams, the acceptance rate should be high leading to fast convergence. On the other hand, if the two distributions are far apart, the proposed sample will be rejected often causing the sampler to converge slowly. In the best case scenario, the sampling algorithm will be significantly speeded up since instead of calling the high order LM during all the intermediate sampling steps, the latter is called only once per sampler iteration. As this algorithm

[1]This step is performed immediately after line 10 in Algorithm 5.2.

uses the MCMC machinery, it also retains all the theoretical guarantees of sampling approaches.

8.2.2 Feature Engineering

Arguably the most exciting avenue of future research provided by the sampling framework is through its support for feature engineering. Feature engineering is a very active current area of research in SMT. It consists of augmenting the base features found in most log-linear models of SMT with additional features capable of identifying characteristics of the input and the output sentences that are indicative of whether the output is good or not.

Feature engineering for SMT was hindered for years due a lack of alternative to MERT, a training algorithm which does not scale to a large number of features. Recent work using margin-based optimisation algorithms has shown that the translation performance of some syntax-based linear translation models (Watanabe et al., 2007; Chiang et al., 2008b, 2009) can be improved with the addition of a large number of sparse features, therefore demonstrating the promise of this research avenue. Phrase-based translation models should benefit from feature engineering too. The sampling framework offers support for such a pursuit by allowing the use of gradient-based optimisation algorithms which scale to a large number of features.

The sampling framework offers another advantage. Recall that for beam decoding methods to be computationally efficient, the features characterising the steps in the derivation must be either computable independently of each other or with only limited local context (as in the case of the language model or distortion costs). This has led to a situation where entire classes of potentially useful features are not considered because they would be impractical to integrate into a dynamic programming based translation system. In the sampling framework, this restriction is lifted. Any function of $h(\mathbf{e}, \mathbf{f}, \mathbf{d})$, local or global, may participate in the translation model subject only to its own computability.

The sampling approach presented in this thesis therefore offers the SMT community a unique framework for improving machine translation quality by exploring a vast number of both local and global features.

8.2.3 Syntax-based Translation Models

The sampling framework proposed in this thesis has many benefits. In addition to enabling tractable unbiased minimum risk training and decoding, it also allows the verification of the utility of standard approximation techniques such as the dynamic programming based max derivation decoding employed by most SMT systems. As such, it would be useful to have a sampler for syntax-based models too.

We give a brief sketch of how this sampler would look like for a synchronous contex-free grammar (SCFG) based translation model. For this model, given an exhaustive translation forest, i.e. a packed representation of *all* the derivations in the search space, a fast top-down recursive algorithm for sampling derivations exists and is described in Blunsom and Osborne (2008). However, such an exhaustive translation forest is possible only for models without a language model; the addition of a language model causes the packed representation to be too large to be practical.

Denoting the forest for the model without a language model as the -LM forest, an algorithm for sampling derivations from a SCFG model with a language model can make use of the algorithm for sampling with -LM forest. Samples can be drawn from the latter and then rescored with a language model. A Metropolis-Hastings step is then used to accept or reject the proposed derivation.

Bibliography

Aarts, E. H. L. and Laarhoven, P. J. M. V. (1987). *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43.

Arun, A., Dyer, C., Haddow, B., Blunsom, P., Lopez, A., and Koehn, P. (2009). Monte Carlo Inference and Maximization for Phrase-based Translation. In *Proceedings of CoNLL*, pages 102–110, Boulder, Colorado. Association for Computational Linguistics.

Arun, A., Dyer, C., Haddow, B., Blunsom, P., Lopez, A., and Koehn, P. (2010a). Monte Carlo techniques for phrase-based translation. In *Special Issue of the Machine Translation journal,*. Springer.

Arun, A., Haddow, B., and Koehn, P. (2010b). A unified approach to minimum risk training and decoding. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 365–374, Uppsala, Sweden. Association for Computational Linguistics.

Arun, A. and Koehn, P. (2007). Online Learning Methods For Discriminative Training of Phrase Based Statistical Machine Translation. In *Proceedings of MT Summit XI*, Copenhagen, Denmark.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition.

Blunsom, P., Cohn, T., Dyer, C., and Osborne, M. (2009). A Gibbs Sampler for Phrasal Synchronous Grammar Induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 782–790, Suntec, Singapore. Association for Computational Linguistics.

Blunsom, P., Cohn, T., and Osborne, M. (2008). A Discriminative Latent Variable Model for Statistical Machine Translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio. Association for Computational Linguistics.

Blunsom, P. and Osborne, M. (2008). Probabilistic Inference for Machine Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 215–223, Honolulu, Hawaii. Association for Computational Linguistics.

Bouchard-Côté, A., Petrov, S., and Klein, D. (2009). Randomized Pruning: Efficiently Calculating Expectations in Large Dynamic Programs. In *Advances in Neural Information Processing Systems 22*, pages 144–152.

Brown, P., Cocke, J., Della-Pietra, S., Della-Pietra, V., Jelinek, F., Lafferty, J., Mercer, R., and Roossin, P. (1990). A Statistical Approach to Machine Translation. *Computational Linguistics*, 16:79–85.

Brown, P. F., Della Pietra, V. J., Della Pietra, S. A., and Mercer, R. L. (1993). The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Callison-Burch, C., Koehn, P., Monz, C., and Schroeder, J. (2009). Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece. Association for Computational Linguistics.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluating the Role of BLEU in Machine Translation Research. In *In EACL*, pages 249–256.

Chen, S. F. and Rosenfeld, R. (1999). A Gaussian Prior for Smoothing Maximum Entropy Models. Technical report.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270, Morristown, NJ, USA. Association for Computational Linguistics.

Chiang, D. (2007). Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228.

Chiang, D., DeNeefe, S., Chan, Y. S., and Ng, H. T. (2008a). Decomposability of Translation Metrics for Improved Evaluation and Efficient Algorithms. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 610–619, Honolulu, Hawaii. Association for Computational Linguistics.

Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *In North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT*.

Chiang, D., Marton, Y., and Resnik, P. (2008b). Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, Hawaii. Association for Computational Linguistics.

Cohn, T. and Blunsom, P. (2009). A Bayesian Model of Syntax-Directed Tree to String Grammar Induction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 352–361, Singapore.

Cohn, T., Goldwater, S., and Blunsom, P. (2009). Inducing Compact but Accurate Tree-Substitution Grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 548–556, Boulder, Colorado.

DeNero, J., Bouchard-Côté, A., and Klein, D. (2008). Sampling Alignment Structure under a Bayesian Translation Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Honolulu, Hawaii. Association for Computational Linguistics.

DeNero, J., Chiang, D., and Knight, K. (2009). Fast Consensus Decoding over Translation Forests. In *Proceedings of the Joint Conference of the 47th Annual*

Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 567–575, Suntec, Singapore. Association for Computational Linguistics.

Eisner, J. (2003). Learning Non-Isomorphic Tree Mappings for Machine Translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, Sapporo, Japan. Association for Computational Linguistics.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan. Association for Computational Linguistics.

Finkel, J. R., Manning, C. D., and Ng, A. Y. (2006). Solving the Problem of Cascading Errors: Approximate Bayesian Inference for Linguistic Annotation Pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, Sydney, Australia. Association for Computational Linguistics.

Foster, G. and Kuhn, R. (2009). Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 242–249, Athens, Greece. Association for Computational Linguistics.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia. Association for Computational Linguistics.

Gauvain, J.-l. and Lee, C.-h. (1994). Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298.

Gelman, A. and Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Germann, U., Jahr, M., Knight, K., Marcu, D., and Yamada, K. (2001). Fast Decoding and Optimal Decoding for Machine Translation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 228–235, Toulouse, France. Association for Computational Linguistics.

Geyer, C. J. (1992). Practical Markov Chain Monte Carlo. *Statistical Science*, 7(4):473–483.

Goldwater, S. and Griffiths, T. (2007). A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic. Association for Computational Linguistics.

Goldwater, S., Griffiths, T. L., and Johnson, M. (2006). Contextual Dependencies in Unsupervised Word Segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia. Association for Computational Linguistics.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109.

Ittycheriah, A. and Roukos, S. (2007). Direct translation model 2. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 57–64, Rochester, New York. Association for Computational Linguistics.

Jensen, C. S., Kong, A., and Kjaerulff, U. (1993). Blocking Gibbs Sampling in Very Large Probabilistic Expert Systems. *International Journal of Human Computer Studies. Special Issue on Real-World Applications of Uncertain Reasoning*, 42:647–666.

Johnson, H., Martin, J., Foster, G., and Kuhn, R. (2007a). Improving Translation Quality by Discarding Most of the Phrasetable. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 967–975, Prague, Czech Republic. Association for Computational Linguistics.

Johnson, M., Griffiths, T., and Goldwater, S. (2007b). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York. Association for Computational Linguistics.

Kirkpatrick, S., Gelatt, C. D., Jr., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220:671–680.

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In *Proc. of IEEE Internation Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

Knight, K. (1999). Decoding complexity in word-replacement translation models. *Comput. Linguist.*, 25(4):607–615.

Koehn, P. (2004a). Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*.

Koehn, P. (2004b). Statistical significance tests for machine translation evaluation. In Lin, D. and Wu, D., editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Koehn, P., Hoang, H., Mayne, A. B., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL Demonstration Session*, pages 177–180.

Koehn, P., Och, F., and Marcu, D. (2003). Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 48–54, Morristown, NJ, USA.

Kumar, S. and Byrne, W. (2004). Minimum Bayes-Risk Decoding for Statistical Machine Translation. In Susan Dumais, D. M. and Roukos, S., editors, *HLT-NAACL 2004: Main Proceedings*, pages 169–176, Boston, Massachusetts, USA. Association for Computational Linguistics.

Kumar, S., Macherey, W., Dyer, C., and Och, F. (2009). Efficient Minimum Error Rate Training and Minimum Bayes-Risk Decoding for Translation Hypergraphs and Lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the*

ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 163–171, Suntec, Singapore. Association for Computational Linguistics.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.

Langlais, P., Gotti, F., and Patry, A. (2007). A Greedy Decoder for Phrase-Based Statistical Machine Translation. In *11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2007)*, pages 104–113, Skovde, Sweden.

Levine, R. A. and Casella, G. (2006). Optimizing random scan gibbs samplers. *J. Multivar. Anal.*, 97(10):2071–2100.

Lewis, P. M. and Stearns, R. E. (1966). Syntax directed transduction. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:21–35.

Li, Z., Callison-Burch, C., Dyer, C., Ganitkevitch, J., Khudanpur, S., Schwartz, L., Thornton, W. N. G., Weese, J., and Zaidan, O. F. (2009a). Joshua: an open source toolkit for parsing-based machine translation. In *StatMT '09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Morristown, NJ, USA. Association for Computational Linguistics.

Li, Z. and Eisner, J. (2009). First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 40–51, Singapore.

Li, Z., Eisner, J., and Khudanpur, S. (2009b). Variational Decoding for Statistical Machine Translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 593–601.

Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. (2006). An End-to-End Discriminative Approach to Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia. Association for Computational Linguistics.

Lin, C.-Y. and Och, F. J. (2004). ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation . In *Proceedings of Coling 2004*, pages 501–507, Geneva, Switzerland. COLING.

Liu, D. C. and Nocedal, J. (1989). On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45(3):503–528.

Lopez, A. (2008). Statistical machine translation. *ACM Comput. Surv.*, 40(3):1–49.

Lopez, A. (2009). Translation as Weighted Deduction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 532–540, Athens, Greece. Association for Computational Linguistics.

Lopez, A. (2010). Markov chain Monte Carlo in very large structured search spaces. Unpublished draft.

Macherey, W., Och, F., Thayer, I., and Uszkoreit, J. (2008). Lattice-based Minimum Error Rate Training for Statistical Machine Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734, Honolulu, Hawaii. Association for Computational Linguistics.

Marcu, D., Wang, W., Echihabi, A., and Knight, K. (2006). SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 44–52, Sydney, Australia. Association for Computational Linguistics.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 133–139, Morristown, NJ, USA. Association for Computational Linguistics.

May, J. and Knight, K. (2006). A Better N-Best List: Practical Determinization of Weighted Finite Tree Automata. In Khudanpur, S. and Roark, B., editors, *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, volume Main Proceedings, pages 351–358, Brooklyn, New York. Association for Computational Linguistics.

Mcdonald, R. (2006). Discriminative sentence compression with soft syntactic constraints. In *In Proc. EACL*.

Mcdonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *In Proc. ACL*, pages 91–98.

Melamed, I. D., Green, R., and Turian, J. P. (2003). Precision and recall of machine translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 61–63, Morristown, NJ, USA. Association for Computational Linguistics.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341.

Neal, R. (1993). Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical report, Dept. of Computer Science, University of Toronto.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. Association for Computational Linguistics.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., and Radev, D. (2004). A smorgasbord of features for statistical machine translation. In Susan Dumais, D. M. and Roukos, S., editors, *HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. In *Proc. of the Conference on Empirical Methods in*

Natural Language Processing and Very Large Corpora, pages 20–28, University of Maryland, College Park, MD.

Och, F. J., Ueffing, N., and Ney, H. (2001). An efficient A* search algorithm for statistical machine translation. In *Proceedings of the workshop on Data-driven methods in machine translation*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pauls, A., Denero, J., and Klein, D. (2009). Consensus Training for Consensus Decoding in Machine Translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1418–1427, Singapore. Association for Computational Linguistics.

Quirk, C., Menezes, A., and Cherry, C. (2005). Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 271–279, Morristown, NJ, USA. Association for Computational Linguistics.

Riedel, S. and Clarke, J. (2009). Revisiting Optimal Decoding for Machine Translation IBM Model 4. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 5–8, Boulder, Colorado. Association for Computational Linguistics.

Robert, C. P. and Casella, G. (2005). *Monte Carlo Statistical Methods*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Rose, K. (1998). Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems. In *Proceedings of the IEEE*, pages 2210–2239.

Schraudolph, N. N. (1999). Local Gain Adaptation in Stochastic Gradient Descent. Technical Report IDSIA-09-99, IDSIA.

Shen, L., Sarkar, A., and Och, F. J. (2004). Discriminative reranking for machine translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Sixtus, S. and Ortmanns, S. (1999). High quality word graphs using forward-backward pruning. In *Proc. of ICASSP*, Phoenix, AZ.

Smith, D. A. and Eisner, J. (2006). Minimum Risk Annealing for Training Log-Linear Models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794, Sydney, Australia. Association for Computational Linguistics.

Smith, N. A. (2006). Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text. Technical report, Johns Hopkins University.

Smith, N. A. and Eisner, J. (2004). Annealing Techniques for Unsupervised Statistical Language Learning. In *In Proc. of ACL*, pages 486–493.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Stolcke, A. (2002). SRILM – an extensible language modeling toolkit. In *Intl. Conf. on Spoken Language Processing*.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Torn, A. and Zhilinskas, A. (1989). *Global optimization*. Springer-Verlag, Berlin ; New York :.

Tromble, R., Kumar, S., Och, F., and Macherey, W. (2008). Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 620–629, Honolulu, Hawaii. Association for Computational Linguistics.

Walsh, B. (2004). Markov Chain Monte Carlo and Gibbs Sampling.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online Large-Margin Training for Statistical Machine Translation. In *Proceedings of the 2007 Joint*

Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 764–773, Prague, Czech Republic. Association for Computational Linguistics.

Zens, R., Hasan, S., and Ney, H. (2007). A Systematic Comparison of Training Criteria for Statistical Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 524–532.

Zollmann, A., Venugopal, A., Och, F., and Ponte, J. (2008). A Systematic Comparison of Phrase-Based, Hierarchical and Syntax-Augmented Statistical MT. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1145–1152, Manchester, UK. Coling 2008 Organizing Committee.