Refinements in Hierarchical Phrase-Based Translation Systems

Juan Miguel Pino

Cambridge University Engineering Department

Dissertation submitted to the University of Cambridge for the degree of Doctor of Philosophy

Declaration

I hereby declare that the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

Acknowledgements

First, I am truly indebted to my supervisor Prof. Bill Byrne. Bill provided relentless support throughout my PhD studies. He taught me how to be a better writer and a better researcher. I am also very grateful that he provided me with invaluable opportunities such as participating in translation competitions or carrying out an internship at Google for three months.

I also would like to thank Dr. Stephen Clark and Dr. Miles Osborne for being part of my thesis committee and providing very valuable feedback on the first submission of this thesis.

Without the lab Computer Officers, Patrick Gosling and Anna Langley, experiments reported in this thesis would not have been possible, I am therefore grateful for all their hard work on the computer systems.

I also wish to thank all my colleagues at the Machine Intelligence Lab. Dr. Adrià de Gispert provided me with critical guidance throughout my research: two chapters of this thesis originated from a very fruitful collaboration with him. I am grateful to Aurelien Waite for all his help on infrastructure and for helping me becoming a better engineer. Thanks to Dr. Gonzalo Iglesias for illuminating discussions on FSTs for machine translation as well as C++ software engineering. Dr. Graeme Blackwood and Dr. Jamie Brunning provided much support at the start of my studies as well as critical translation tools such as word alignment and lattice rescoring tools.

I am also in debt to my office colleagues Matt Shannon, Matt Seigel and Chao Zhang for making PhD studies a pleasurable experience.

Finally, thank you Anna for your infinite patience, guidance and support and for telling me to apply for at least two jobs at the end of my studies. Thank you Sofia and Fania: I will try to teach you many languages, just in case machine translation doesn't improve quickly enough.

Abstract

The relatively recently proposed hierarchical phrase-based translation model for statistical machine translation (SMT) has achieved state-of-the-art performance in numerous recent translation evaluations. Hierarchical phrase-based systems comprise a pipeline of modules with complex interactions. In this thesis, we propose refinements to the hierarchical phrase-based model as well as improvements and analyses in various modules for hierarchical phrase-based systems.

We took the opportunity of increasing amounts of available training data for machine translation as well as existing frameworks for distributed computing in order to build better infrastructure for extraction, estimation and retrieval of hierarchical phrase-based grammars. We design and implement grammar extraction as a series of Hadoop MapReduce jobs. We store the resulting grammar using the HFile format, which offers competitive trade-offs in terms of efficiency and simplicity. We demonstrate improvements over two alternative solutions used in machine translation.

The modular nature of the SMT pipeline, while allowing individual improvements, has the disadvantage that errors committed by one module are propagated to the next. This thesis alleviates this issue between the word alignment module and the grammar extraction and estimation module by considering richer statistics from word alignment models in extraction. We use alignment link and alignment phrase pair posterior probabilities for grammar extraction and estimation and demonstrate translation improvements in Chinese to English translation.

This thesis also proposes refinements in grammar and language modelling both in the context of domain adaptation and in the context of the interaction between first-pass decoding and lattice rescoring. We analyse alternative strategies for grammar and language model cross-domain adaptation. We also study interactions between first-pass and second-pass language model in terms of size and n-gram order. Finally, we analyse two smoothing methods for large 5-gram language model rescoring.

The last two chapters are devoted to the application of phrase-based grammars to the string regeneration task, which we consider as a means to study the fluency of machine translation output. We design and implement a monolingual phrase-based decoder for string regeneration and achieve state-of-the-art performance on this task. By applying our decoder to the output of a hierarchical phrase-based translation system, we are able to recover the same level of translation quality as the translation system.

Contents

Chapter 1

Introduction

1.1 Machine translation

Machine translation is the process of translation from input speech or text in a natural language into another natural language by some kind of automatic system. Real world examples include online services such as Google Translate[1], Bing Translate[2], SDL[3], PROMT[4], etc. Interacting with any online automatic translation service with the expectation of a high quality translation can be a frustrating experience. Indeed, variations in word order across languages and syntax and the use of real world knowledge for puns or idioms make translation a very challenging task.

1.1.1 Challenges for Translation

In order to illustrate the difficulties that arise in translation, we present several examples that make translation challenging for humans, and *a fortiori* for computers. In some languages, some concepts are common enough to be designated by one word, but in another language, an entire sentence may be needed to describe that concept. For example, the word *sobremesa* in Spanish can be translated into English as *the time spent after a meal, talking to the people with whom the meal was shared*.[5] In this situation, a human trans-

[1] https://translate.google.com
[2] https://www.bing.com/translator
[3] http://www.freetranslation.com/
[4] http://www.online-translator.com
[5] http://blog.maptia.com/posts/untranslatable-words-from-other-cultures

lator is left with the choice of keeping the translation short but inexact or long but cumbersome. For a computer, or rather a statistical model, such a situation will represent an outlier in terms of the ratio of the number of words that are translation of each other. Self-referential sentences can also present challenges for translation. For example, the sentence *This sentence has five words* has at least two acceptable translations into Russian from the syntactic and semantic point of view: *В этом предложении пять слов* and *Это предложение состоит из пяти слов*. However, the second translation has six words and therefore cannot be accepted. Another challenge for translation is word ordering. The first sentence of the novel *The Metamorphosis* by Franz Kafka reads *Als Gregor Samsa eines Morgens aus unruhigen Träumen erwachte, fand er sich in seinem Bett zu einem ungeheuren Ungeziefer verwandelt*. One possible English translation is *As Gregor Samsa awoke one morning from uneasy dreams, he found himself transformed in his bed into a gigantic insect-like creature*. In German, the words for *transformed* (*verwandelt*) and *insect* (*Ungeziefer*) come right at the end of the sentence and create an effect of surprise for the reader. In English, however, the verb *transformed* comes in the middle of the sentence and the effect of surprise is not as great. This example demonstrates how variations in word ordering between languages can make translation challenging. Computers have the additional challenge that the choice of word ordering should produce a grammatical sentence. This is a challenge for humans too but computers are particularly bad at producing a grammatical output.

1.1.2 Machine Translation Current Quality

Even though machine translation is a challenging task, it is still useful in a number of situations. For example, machine translation can be used to obtain the *gist*, i.e. the general meaning, of a document in a foreign language. Machine translation can also be used for post-editing: a document in a foreign language is first automatically translated and then the translation is corrected by a human translators. This is precisely the approach taken by translation companies such as Unbabel[6].

Machine translation has therefore gotten to the point where it is actually useful for practical applications, but it is reasonable to ask how far we are from perfect translation. We give indications in terms of the BLEU met-

[6] https://www.unbabel.com

System	1-2-3-4	2-3-4	1-3-4	1-2-4	1-2-3
CUED	38.95	34.80	35.60	35.87	35.28
Reference 1	–	46.19	–	–	–
Reference 2	–	–	47.27	–	–
Reference 3	–	–	–	45.43	–
Reference 4	–	–	–	–	46.90

Table 1.1: BLEU score obtained by the Cambridge University Engineering Department system and by human translators on the MT08 test set. 4 references are provided. The BLEU score is measured against various subsets of references: all references (1-2-3-4), all but the first (2-3-4), etc. The BLEU score for a human reference when the set of reference contains the human reference is not computed and is simply 100.

ric (Papineni et al., 2002), which we will define in Section 2.8.1. For now, we simply need to know that BLEU measures how well a translation hypothesis resembles a set of human translation references. The Cambridge University Engineering Department (CUED) submitted a translation system to the NIST Open Machine Translation 2012 Evaluation.[7] We run this system on the newswire portion of the NIST Open Machine Translation 2008 Evaluation test set (MT08). For this MT08 test set, 4 references are available. We measure the case insensitive BLEU score of the CUED system against the 4 references and against subsets of 3 references. Similarly, we measure the case insensitive BLEU score of each reference against the other references. BLEU scores for humans and for the CUED system are summarised in Table 1.1. On average, CUED obtains a BLEU score of 35.39 against 3 references. On average, human references obtain a BLEU score 46.45 against the other references. We can draw two conclusions from these observations. First, this is evidence that many possible translations are acceptable. Second, because the automatic system performance for this particular setting is approximately 10 BLEU points below human performance, this gives an indication of the quality of state-of-the-art automatic translation. After manual inspection of the system output, we show a few examples of output sentences that are relatively long and still have reasonable quality in Figure 1.1:

[7] `http://www.nist.gov/itl/iad/mig/openmt12.cfm`

Minister of agriculture and social affairs, told AFP: "The prime minister submitted his resignation to Abbas, chairman of the president, and at the same time asked him to form a new cabinet, responsible for handling daily affairs of the new government."

Reference: Minister for Agriculture and Social Affairs Habbash told AFP: "Prime Minister Fayyad offered his resignation to President Abbas. The president accepted it and at the same time also asked him to form a new cabinet government to be responsible for routine administration."

Chavez has ordered government officials to keep an eye on foreigners visiting venezuela's speech, when a person is found to have publicly criticized him or the Venezuelan government, are to be deported.

Reference: Chavez has already ordered government officials to closely monitor the speech of foreigners when they visit Venezuela. If anyone is found publicly criticizing him or the Venezuelan government, they should be all deported.

US-China strategic economic dialogue "focused on the economic, environmental and other issues, the most important thing is that the issue of the RMB exchange rate, the US Congress members are of the view that the value of the Renminbi underestimated.

Reference: The US-China Strategic Economic Dialogue mainly discusses economic, environmental protection and other issues. Yet, the most important is the issue of the renminbi exchange rate. US congressmen believe that the renminbi is excessively undervalued.

Figure 1.1: Example output of a translation system, together with one reference. Even though the sentences are relatively long, they are overall fluent and intelligible. True casing is restored manually.

1.2 Statistical Machine Translation

The current dominant approach to machine translation is a statistical approach. Given an input in a natural language, a statistical model of translation will attempt to predict the best translation according to a statistical criterion, such as the most likely translation under a probabilistic model of translation. The data used to estimate the parameters for such a model consists of a *parallel corpus*, which is a set of sentence pairs in two different natural languages that are translation of each other, and a monolingual corpus which is a set of sentences in the language we would like to translate into.

Parallel data can be obtained from multilingual institutions proceedings such as the United Nations (Franz et al., 2013),the Canadian Hansard (Germann, 2001) or the European Parliament (Koehn, 2005). This type of data is relatively clean since it is created by professional translators. However, it may not match the genre of the input sentences we wish to translate, such as newswire. The importance of having a good match between the data used for training and testing is discussed in Section 5.3. In order to address this concern and also in order to obtain more data, parallel data can also be extracted automatically from *comparable* corpora (Smith et al., 2013). However, the widespread availability of machine translation and the development of automatic techniques to extract parallel corpora automatically increase the risk of having automatically translated output of poor quality present in the training data. These concerns have been acknowledged and addressed by a watermarking algorithm (Venugopal et al., 2011).

1.2.1 The SMT Pipeline

Typically, state-of-the-art SMT systems are organised into a pipeline of deterministic or statistical modules, as shown in Figure 1.2. Parallel text is first preprocessed. This consists of data cleaning and tokenisation. For morphologically poor languages such as English, simple rules for tokenisation, e.g. separate words by white space and punctuation, are usually good enough. For morphologically rich languages, more advanced techniques are needed for effective tokenisation, for example morphological analysis (Habash and Rambow, 2005), in order to combat data sparsity and work with a vocabulary with reasonable size. In the case of languages without word boundaries, such as Chinese, word segmentation techniques need to be applied (Zhang and Clark, 2007), in order to be able to break sentences into words.

The preprocessed parallel text is then word-aligned (see Section 2.3). A word alignment is simply a mapping between source words and target words in a parallel sentence pair. Word alignment models were originally used to describe the translation process and to perform translation (Germann et al., 2001). Word alignment toolkits are available online[8,9] as well as a word to word translation decoder.[10] Nowadays, word alignment models are used as an intermediate step in the translation pipeline. A translation grammar is then extracted and estimated from the word alignments (see Section 2.6.4) and translation is performed under the translation grammar.

The monolingual data is also preprocessed in the same fashion as the target side of the parallel text and one or more language models are estimated from this data (see section Section 2.7). The language models and the translation grammar are used by the translation decoder for translation. In order to optimise the SMT model parameters, alternating decoding and tuning steps are carried out.

Typically, a decoder can output a best possible translation, or an n-best list of translations or even a lattice of translations, which is a compact representation of an n-best list with a very large number n. In the latter case, optional rescoring steps can be carried out in order to include models that are too complex or not robust enough to be included in first pass decoding (see Section 2.10). In particular, we study a model of string regeneration that allows any reordering of the output of the first pass decoder in Chapter 7.

1.3 Contributions

The contributions of this thesis are outlined as follows:

- We describe a novel approach to grammar extraction and estimation. Our approach ties the models of word alignment and grammar extraction and estimation more closely. It results in a more robust extraction and estimation of translation grammars and leads to improvements in translation quality, as demonstrated in Chinese to English translation experiments.

[8] http://mi.eng.cam.ac.uk/~wjb31/distrib/mttkv1
[9] https://code.google.com/p/giza-pp
[10] http://www.isi.edu/licensed-sw/rewrite-decoder

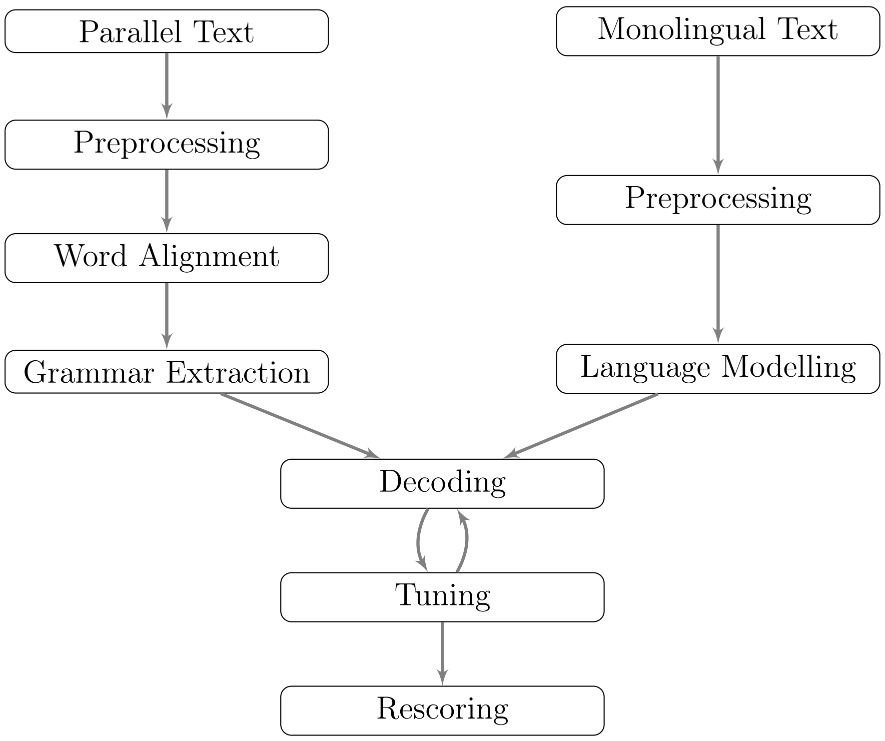

Figure 1.2: Machine Translation System Development Pipeline

- We describe a system that allows the efficient extraction and filtering of very large grammars. This method has been in continued use at CUED and was employed for submissions to the NIST 2012[11] and the WMT 2013 (Bojar et al., 2013) translation evaluations. This was a carefully engineering effort that required detailed understanding of grammar extraction procedures and how to implement them in a Hadoop framework. There are many implementation strategies that can be taken, but obtaining the processing performance needed to support the experiments reported in this thesis requires a careful design process.

- We designed and implemented a system for string regeneration, inspired from phrase-based SMT techniques. We obtain state-of-the-art results in the string regeneration task and demonstrate potential applications to machine translation.

1.4 Organisation of the thesis

We now describe the thesis organisation. In Chapter 2, we review statistical machine translation background: all components of the machine translation pipeline presented in Figure 1.2 are reviewed in detail. In Chapter 3, we present our system for efficient extraction of translation grammars from parallel text and retrieval of translation rules from very large translation grammars. In Chapter 4, we present our novel grammar extraction procedure that makes use of posterior probabilities from word alignment models. We then provide hierarchical phrase-based system building recommendations about what decisions to make in terms of language modelling and grammar design to obtain the best possible systems for translation in Chapter 5. In Chapter 6, we introduce our phrase-based decoder for string regeneration and study fluency through the word reordering task. Finally, in Chapter 7, we apply our regeneration decoder to the output of a machine translation system.

[11] `http://www.nist.gov/itl/iad/mig/openmt12.cfm`

Chapter 2

Statistical Machine Translation

Statistical Machine Translation (SMT) (Brown et al., 1993; Lopez, 2008; Koehn, 2010) has become the dominant approach to machine translation, as increasing amounts of data and computing power have become available. In the SMT paradigm, given a sentence in a source language, conceptually all possible sentences in a target language are assigned a probability, a score, or a cost and the best translation is picked according to a certain decision criterion that relates to these probabilities, scores or costs. The research challenge is to develop models that assign scores that reflect human judgements of translation quality.

In this chapter, we first review the historical background of SMT in Section 2.1. We then present the original source-channel model for SMT in Section 2.2. Word alignment models, which we review in Section 2.3, were introduced within the framework of the source-channel model. The original source-channel model was extended into the log-linear model, presented in Section 2.4. The field of SMT shifted from word-based models to phrase-based models, introduced in Section 2.5, while retaining word-based models in their first formulation as a preliminary step. Phrase-based translation was extended into hierarchical phrase-based translation, which we review in Section 2.6. We then examine various features employed in state-of-the-art decoders in Section 2.6.5. The target language model, which is one of the most important features in translation, is explored in more detail in Section 2.7. In Section 2.8, we review optimisation techniques that are employed in order to tune the decoder parameters. We finally present how finite state transducers can be used in decoding in Section 2.9. Various rescoring procedures are reviewed in Section 2.10.

2.1 Historical Background

In this section, we present a brief historical background of *statistical* machine translation. A more comprehensive account of the history of machine translation in general can be found elsewhere (Hutchins, 1997; Hutchins and Lovtskii, 2000).

Warren Weaver can be considered the father of modern SMT. At a time when the first computers were being developed, he examined their potential application to the problem of machine translation. In his memorandum (Weaver, 1955), he addressed the problem of multiple meanings of a source word by considering the context of that source word, which heralds phrase based translation techniques and the use of context in machine translation. He was also the first to frame machine translation as a source-channel model by considering that a sentence in a foreign language is some form of code that needs to be broken, in analogy to the field of cryptography. Finally, he also emphasised the statistical aspect of machine translation. However, he also predicted that the most successful approaches to machine translation would take advantage of language invariants by using an intermediate language representation in the translation process. Even though state-of-the-art statistical translation systems do not use this kind of approach, we do notice a resurgence in intermediate language representation techniques (Mikolov et al., 2013).

The first successful implementations of Warren Weaver's ideas were carried out by IBM in the 1990s. The source-channel model together with a series of word alignment models were introduced by Brown et al. (1993) while Berger et al. (1996) addressed the problem of multiple meanings using context in a maximum entropy framework. Word-based models were extended into different variants of phrase-based models in 1999 and at the beginning of the century (Och et al., 1999; Koehn et al., 2003; Och and Ney, 2004) and later on into synchronous context-free grammar models (Chiang, 2005, 2007).

2.2 Source-Channel Model

Statistical machine translation was originally framed as a source-channel model (Shannon, 1948; Brown et al., 1990, 1993). Given a foreign sentence $\boldsymbol{f}$, we want to find the original English sentence $\boldsymbol{e}$ that went through a noisy channel and produced $\boldsymbol{f}$. Note that in the source-channel model notation, what we would like to recover—the English sentence—is called the *source* while what is observed—the foreign sentence—is called the *target*. A source-channel model assigns probabilities from source (English) to target (foreign) but in translation, the model is used to infer the source that was most likely to have generated the target.

We do not use this convention here and call the *source* what we are translating from and the *target* what we are translating into. This convention is frequently adopted (Och et al., 1999; Och and Ney, 2002, 2004) in SMT, and more so since SMT has been framed as a log-linear model (see Section 2.4). We use the decision rule in Equation 2.1, which minimises the risk under a zero-one loss function (see Section 2.10.2):

$$
\begin{aligned}
\hat{\boldsymbol{e}} &= \operatorname*{argmax}_{\boldsymbol{e}} p(\boldsymbol{e} \mid \boldsymbol{f}) \\
\hat{\boldsymbol{e}} &= \operatorname*{argmax}_{\boldsymbol{e}} \frac{p(\boldsymbol{f} \mid \boldsymbol{e})\, p(\boldsymbol{e})}{p(\boldsymbol{f})} \quad \text{(Bayes' rule)} \\
\hat{\boldsymbol{e}} &= \operatorname*{argmax}_{\boldsymbol{e}} p(\boldsymbol{f} \mid \boldsymbol{e})\, p(\boldsymbol{e}) \qquad (2.1)
\end{aligned}
$$

$\hat{\boldsymbol{e}}$ is the hypothesis to be selected. $p(\boldsymbol{f} \mid \boldsymbol{e})$ is called the *translation model* while $p(\boldsymbol{e})$ is called the (target) *language model*.

The translation model and the language model are estimated separately for practical reasons: the amount of parallel data used to train the translation model is in general orders of magnitude smaller than the amount of monolingual data used to train the language model. Another justification is that using two separate models makes the translation process modular: improving the translation model may help improve *adequacy*, i.e. how well the meaning of the source text is preserved in the translated text, while improving the language model may help improve *fluency*, i.e. how well-formed the translation is. It is therefore considered preferable to train both a translation model and a language model. In these models, parallel sentence pairs and target sentences are not used directly as parameters because of an obvious sparsity problem. Parallel sentence pairs are further broken down using word-based models (see Section 2.3), phrase-based models (see Section 2.5) and hierarchical phrase-based models (see Section 2.6). For language modelling, sentences are broken down into windows of consecutive words using n-gram language models (see Section 2.7). We will see in the next section how to decompose the translation model using word alignment, which is introduced as a latent variable into the source-channel model.

2.3 Word Alignment

In the previous section, we have briefly described the source-channel model, which describes the translation process. This model cannot be used directly in practice as it has too many parameters, namely all imaginable sentence pairs and target sentences. In order to address this issue, the *alignment* between source words and target words will be introduced as a latent variable in the source channel model.

Given a sentence pair $(\boldsymbol{f}, \boldsymbol{e})$ with source sentence length $J = |\boldsymbol{f}|$ and target sentence length $I = |\boldsymbol{e}|$, a *word alignment* $\boldsymbol{a}$ for this sentence pair is a mapping between the source and target words. In other words, $\boldsymbol{a}$ is a subset of the cross product of the set of source words and their positions and the set of target words and their positions, as defined in Equation 2.2:

$$\boldsymbol{a} \subset \{((f_j, j), (e_i, i)), (j, i) \in [1, J] \times [1, I]\} \tag{2.2}$$

When the context of which sentence pair $(\boldsymbol{f}, \boldsymbol{e})$ is being word-aligned is obvious, we may simply consider source word positions and target word positions. In that case, $\boldsymbol{a}$ is simply defined as a subset (source position, target position), in Equation 2.3:

$$\boldsymbol{a} \subset [1, J] \times [1, I] \tag{2.3}$$

Each element of $\boldsymbol{a}$ is called an *alignment link*. Alignment links between source and target words correspond to semantic or syntactic equivalences shared by these words in the source and target language and in a particular sentence pair. Alignments can present many-to-one and one-to-many mappings as well as reordering as highlighted by crossing links. An example of word alignment is shown in Figure 2.1.

Brown et al. (1993) introduce the alignment $\boldsymbol{a}$ as a latent variable in the translation model $p(\boldsymbol{f} \mid \boldsymbol{e})$, as in Equation 2.4:

$$p(\boldsymbol{f} \mid \boldsymbol{e}) = \sum_{\boldsymbol{a}} p(\boldsymbol{f}, \boldsymbol{a} \mid \boldsymbol{e}) \tag{2.4}$$

We abuse notation by calling $\boldsymbol{a}$ both the latent variable and the set of alignment links, which is an instance of the latent variable. For mathematical convenience and in order to allow simplifications, given a sentence pair $(\boldsymbol{f}, \boldsymbol{e})$

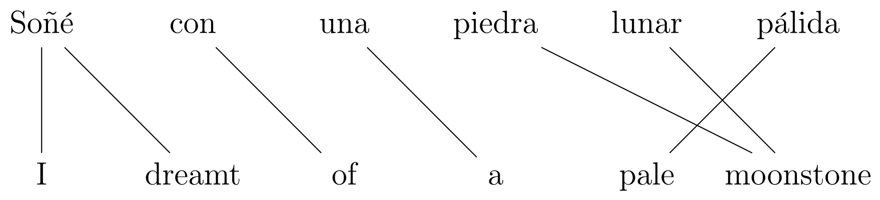

Figure 2.1: Example of word alignment $\boldsymbol{a}$ for a Spanish-English sentence pair. $\boldsymbol{f}$ is the Spanish sentence, $\boldsymbol{e}$ is the English sentence. The source (Spanish) length J is 6 as well as the target (English) length I. This alignment exhibits many-to-one mappings (*I* and *dreamt* align to *Soñé*), one-to-many mappings (*moonstone* aligns to *piedra* and *lunar*), as well as crossing links (the link *pale—pálida* crosses the link *moonstone—lunar*).

with source length J and target length I, $\boldsymbol{a}$ is restricted to be a function from source word positions to target word positions, as in Equation 2.5:

$$\begin{aligned} \boldsymbol{a} : [1, J] &\longrightarrow [0, I] \\ j &\longmapsto a_j \end{aligned} \tag{2.5}$$

The target position zero is included to model source words not aligned to any target word; these unaligned source words are virtually aligned to a so-called *null word*. Note that this definition is not symmetric: it only allows many-to-one mappings from source to target. Various symmetrisation strategies, presented in Section 2.3.2, have been devised to address this limitation. Also note that we did not take into account the null word in our initial definition of alignments in Equation 2.2 because in general, alignments are obtained from symmetrisation heuristics (see Section 2.3.2) where the null word is ignored. We can use the latent variable $\boldsymbol{a}$ to rewrite the translation model in Equation 2.6, with $\boldsymbol{f} = f_1^J$, $\boldsymbol{e} = e_1^I$ and $\boldsymbol{a} = a_1^J$:

$$\begin{aligned} p(f_1^J \mid e_1^I) &= \sum_{a_1^J} p(f_1^J, a_1^J \mid e_1^I) \\ &= \sum_{a_1^J} \prod_{j=1}^{J} p(f_j, a_j \mid f_1^{j-1}, a_1^{j-1}, e_1^I) \\ &= \sum_{a_1^J} \prod_{j=1}^{J} p(f_j \mid f_1^{j-1}, a_1^j, e_1^I)\, p(a_j \mid f_1^{j-1}, a_1^{j-1}, e_1^I) \end{aligned} \tag{2.6}$$

Brown et al. (1993) present a series of five translation models of increasing complexity that parameterise the terms $p(f_j \mid f_1^{j-1}, a_1^j, e_1^I)$ and $p(a_j \mid f_1^{j-1}, a_1^{j-1}, e_1^I)$. Parameter estimation is carried out with the expectation-maximisation algorithm (Dempster et al., 1977). Also based on Equation 2.6, Vogel et al. (1996) introduce an HMM model (Rabiner, 1989) for word alignment and Deng and Byrne (2008) extend the HMM model to a word-to-phrase HMM model. We describe these two models in the following sections.

We have described word alignment models in the context of the source-channel model. In that context, word alignment models can be used directly for word-based decoding.[1] However, nowadays, word alignment models are used as a preliminary step in the machine translation training pipeline, namely prior to rule extraction (see Section 2.5.3 and Section 2.6.4). In that case, the word alignment models are used to produce Viterbi alignments, defined in Equation 2.7:

$$\hat{a}_1^J = \underset{a_1^J}{\operatorname{argmax}}\, p(f_1^J, a_1^J \mid e_1^I) \tag{2.7}$$

One contribution of this thesis is to use alignment posterior probabilities instead of Viterbi alignments for rule extraction (see Chapter 4).

2.3.1 HMM and Word-to-Phrase Alignment Models

We review HMM and word-to-phrase HMM models as these models are used in experiments throughout this thesis. Vogel et al. (1996) introduce an HMM alignment model that treats target word positions as hidden states and source words as observations. The model is written in Equation 2.8:

$$p(f_1^J, a_1^J \mid e_1^I) = \prod_{j=1}^{J} p(a_j \mid a_{j-1}, I)\, p(f_j \mid e_{a_j}) \tag{2.8}$$

Word-to-phrase HMM models (Deng and Byrne, 2008) were designed to capture interesting properties of IBM Model 4 (Brown et al., 1993) in an HMM framework in order to keep alignment and estimation procedures exact. We now present this model in more detail, using our usual source/target convention, which is the reverse than the one adopted in the original publication[2].

[1] e.g. http://www.isi.edu/licensed-sw/rewrite-decoder

[2] $\boldsymbol{s}$ in the publication corresponds to $\boldsymbol{e}$ in this thesis; $\boldsymbol{t}$ corresponds to $\boldsymbol{f}$; J corresponds to J; I corresponds to I.

In the word-to-phrase HMM alignment model, the source sentence $\boldsymbol{f}$ is segmented into source phrases v_1^K. The alignment $\boldsymbol{a}$ is represented by a set of variables $(\phi_1^K, a_1^K, h_1^K, K)$ where:

- K is the number of source phrases that form a segmentation of the source sentence $\boldsymbol{f}$.
- a_1^K is the alignment from target words to source phrases.
- ϕ_1^K indicates the length of each source phrase.
- h_1^K is a *hallucination* sequence that indicates whether a source phrase was generated by the target null word or by a usual target word.

The general form of the model is presented in Equation 2.9:

$$\begin{aligned} p(\boldsymbol{f}, \boldsymbol{a} \mid \boldsymbol{e}) &= p(v_1^K, K, a_1^K, h_1^K, \phi_1^K \mid \boldsymbol{e}) \\ &= p(K \mid J, \boldsymbol{e}) \times \\ &\quad p(a_1^K, \phi_1^K, h_1^K \mid K, J, \boldsymbol{e}) \times \\ &\quad p(v_1^K \mid a_1^K, h_1^K, \phi_1^K, K, J, \boldsymbol{e}) \end{aligned} \tag{2.9}$$

We now review the modelling decisions taken for each of the components from Equation 2.9. The first component is simply modelled by:

$$p(K \mid J, \boldsymbol{e}) = \eta^K \tag{2.10}$$

where η is a threshold that controls the number of segments in the source. The second component is modelled using the Markov assumption:

$$\begin{aligned} p(a_1^K, \phi_1^K, h_1^K \mid K, J, \boldsymbol{e}) &= \prod_{k=1}^{K} p(a_k, h_k, \phi_k \mid a_{k-1}, \phi_{k-1}, h_{k-1}, K, J, \boldsymbol{e}) \\ &= \prod_{k=1}^{K} p(a_k \mid a_{k-1}, h_k, I)\, d(h_k)\, n(\phi_k \mid e_{a_k}) \end{aligned} \tag{2.11}$$

As in the HMM word alignment model a_k depends only on a_{k-1}, the target length I and the binary value h_k. $d(h_k)$ is simply controlled by the parameter p_0 by $d(0) = p_0$. $n(\phi_k \mid e_{a_k})$ is a finite distribution on source phrase length that depends on each target word. This parameter is analogous to the fertility parameter introduced in IBM Model 3 (Brown et al., 1993) and that controls how many source words are aligned to a given target word.

The third component from Equation 2.9 is defined in Equation 2.12 and represents the word-to-phrase translation parameter:

$$p(v_1^K | a_1^K, h_1^K, \phi_1^K, K, J, \boldsymbol{e}) = \prod_{k=1}^{K} p(v_k \mid e_{a_k}, h_k, \phi_k) \tag{2.12}$$

One key contribution from the word-to-phrase HMM model is to use bigram translation probabilities to model one single phrase translation, as shown in Equation 2.13:

$$p(v_k \mid e_{a_k}, h_k, \phi_k) = t_1(v_k[1] \mid h_k \cdot e_{a_k}) \prod_{j=2}^{\phi_k} t_2(v_k[j] \mid v_k[j-1], h_k \cdot e_{a_k}) \tag{2.13}$$

where $h_k \cdot e_{a_k}$ is e_{a_k} if $h_k = 1$ and the null word otherwise, t_1 is a word-to-word translation probability and t_2 is a bigram translation probability.

Figure 2.2 shows a simplified version of the generative story for an HMM word-to-phrase alignment model: first, pick the number of source phrases K according to $P(K \mid J, I)$; then pick a target word given the previously chosen one; finally generate the target phrase from the source word using fertility and bigram translation probabilities. For example, we generate the source phrase *les vaches* from the target word *cows* according to Equation 2.14:

$$p(\textit{les vaches} \mid \textit{cows}) = p(\textit{les} \mid \textit{cows})\, p(\textit{vaches} \mid \textit{cows}, \textit{les}) \tag{2.14}$$

Thus bigram probabilities take into account the context of the target word to some extent.

2.3.2 Symmetrisation Heuristics

We have mentioned that the IBM and HMM alignment models are not symmetric: they only allow a many-to-one mapping from source words to target words. In order to address this issue, one can train alignment models in both source-to-target and target-to-source directions, obtain Viterbi alignments from both models and apply symmetrisation strategies (Och et al., 1999; Och and Ney, 2003; Koehn et al., 2003). Och et al. (1999) designed a first symmetrisation heuristic that was later on dubbed as the *grow* heuristic. Koehn et al. (2003) later extended the *grow* heuristic into the *grow-diag*

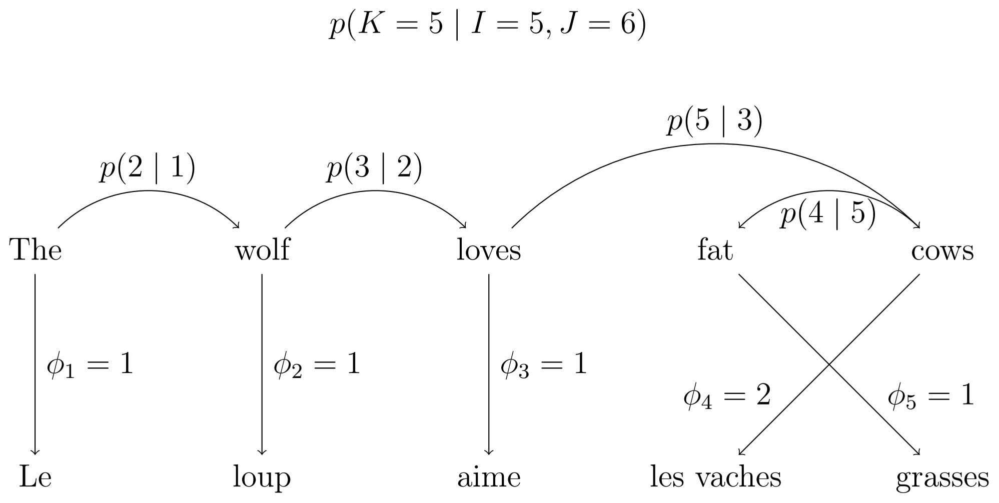

Figure 2.2: Simplified generative story for an HMM word-to-phrase alignment model. Adapted from (Deng and Byrne, 2008). The adjective noun sequence *fat cows* is reordered into the noun adjective sequence *vaches grasses*. The word *cows* has fertility 2 as it is translated into the target phrase *les vaches*.

and *grow-diag-final* heuristics and examined the impact on translation performance for each heuristic.

Alignments from source to target (i.e. in which the alignment is a function from source positions to target positions) and target to source are denoted $\boldsymbol{a}_{f2e}$ and $\boldsymbol{a}_{e2f}$ respectively. Let us consider a sentence pair $(\boldsymbol{f}, \boldsymbol{e})$, and source-to-target and target-to-source Viterbi alignments $\boldsymbol{a}_{f2e}$ and $\boldsymbol{a}_{e2f}$. The *intersection* and *union* heuristics are defined as follows:

- *intersection*: $\boldsymbol{a} = \boldsymbol{a}_{e2f} \cap \boldsymbol{a}_{f2e}$
- *union*: $\boldsymbol{a} = \boldsymbol{a}_{e2f} \cup \boldsymbol{a}_{f2e}$

The *intersection* heuristics typically produces high precision alignments while the *union* heuristics typically produces high recall alignments (Och and Ney, 2003). We now present the *grow* heuristic and its variants, which are based on the initial *intersection* and *union* heuristics. The *grow* heuristic algorithm is presented in Figure 2.3. The input is a sentence pair (f_1^J, e_1^I), a source-to-target alignment $\boldsymbol{a}_{f2e}$ and a target-to-source alignment $\boldsymbol{a}_{e2f}$. The resulting alignment $\boldsymbol{a}$ is initialised with the intersection (line 2). Then alignment links

```
function GROW($f_1^J, e_1^J, \boldsymbol{a}_{f2e}, \boldsymbol{a}_{e2f}$)
    $\boldsymbol{a} \leftarrow \boldsymbol{a}_{f2e} \cap \boldsymbol{a}_{e2f}$
    while true do
        added $\leftarrow$ false
        for $i \in [1, I]$ do
            for $j \in [1, J]$ do
                if $(j, i) \in \boldsymbol{a}$ then
                    for $(k, l) \in$ NEIGHBOURS$((j, i)) \cap (\boldsymbol{a}_{f2e} \cup \boldsymbol{a}_{e2f})$ do
                        if $k$ not aligned in $\boldsymbol{a}$ or $l$ not aligned in $\boldsymbol{a}$ then
                            $\boldsymbol{a} \leftarrow \boldsymbol{a} \cup (k, l)$
                            added $\leftarrow$ true
                        end if
                    end for
                end if
            end for
        end for
        if not added then
            break
        end if
    end while
    return $\boldsymbol{a}$
end function
```

Figure 2.3: Algorithm for the *grow* symmetrisation heuristic (Koehn, 2010). The alignment is initialised from the intersection and alignment links that are neighbours to existing alignment links are iteratively added if the source or the target is not already aligned.

that are in the union and that are neighbours of already existing alignment links (line 8) are considered. If the source or the target word is not already aligned (line 9), then the link is added to the resulting alignment (line 10). This is repeated until no more links are added.

In the *grow* heuristic, neighbours are defined as horizontal or vertical neighbours. If diagonal neighbours are also considered, then the heuristic becomes *grow-diag*. The *grow* heuristic also has an optional step called *final*. Alignment links in the union where the source or the target is not already aligned can also be added to the resulting alignment. If only links in the union where the source *and* the target are not already aligned are considered for the *final* procedure, then the optional step is called *final-and*.

Symmetrisation heuristics have been shown to be beneficial for alignments both in terms of alignment quality as measured by comparing automatic alignments to human alignments and in translation quality when alignments are used as an intermediate step in the translation pipeline.

2.4 Log-Linear Model of Machine Translation

As we have seen in Section 2.2, SMT was historically framed as a source-channel model. As an alternative, Berger et al. (1996) introduce maximum entropy models for natural language processing. In maximum entropy modelling, we wish to estimate a conditional probability $p(\boldsymbol{y} \mid \boldsymbol{x})$. Given a training sample, various feature functions deemed to be relevant are picked. We then constrain p such that the expected value of each feature function f with respect to the empirical distribution is equal to the expected value of f with respect to the model p. Finally, p is chosen among all models that satisfy the constraints defined by the features and such that its entropy is maximum. Berger et al. (1996) show how a maximum entropy model can be parameterised as an exponential, or log-linear model. They apply this model to three machine translation related tasks. First, they use a maximum entropy model to predict the translation of a word using the context for that word. Then, they use a maximum entropy model to predict the target language word order. Finally, they apply maximum entropy modelling in order to predict the source sentence segmentation.

Och et al. (1999) notice that using an erroneous version of the source-channel model, that is using the following equation:

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} p(\boldsymbol{e} \mid \boldsymbol{f})\, p(\boldsymbol{e}) \tag{2.15}$$

gives comparable performance with respect to using the correct formulation of the source-channel model given in Equation 2.1. Then Och and Ney (2002) propose the following log-linear model extension:

$$\begin{aligned}\hat{\boldsymbol{e}} &= \operatorname*{argmax}_{\boldsymbol{e}} p(\boldsymbol{e} \mid \boldsymbol{f}) \\ &= \operatorname*{argmax}_{\boldsymbol{e}} \frac{\exp(\sum_{m=1}^{M} \lambda_m h_m(\boldsymbol{e}, \boldsymbol{f}))}{\sum_{\boldsymbol{e'}} \exp(\sum_{m=1}^{M} \lambda_m h_m(\boldsymbol{e'}, \boldsymbol{f}))} \\ &= \operatorname*{argmax}_{\boldsymbol{e}} \exp(\sum_{m=1}^{M} \lambda_m h_m(\boldsymbol{e}, \boldsymbol{f}))\end{aligned} \tag{2.16}$$

where h_m are called *feature functions* and λ_m are called *feature weights*. The log-linear model is an extension to the source-channel model because it can be reduced to the original source-channel model with the following settings:

- $M = 2$
- $h_1(\boldsymbol{e}, \boldsymbol{f}) = \log(p(\boldsymbol{f}|\boldsymbol{e}))$
- $h_2(\boldsymbol{e}, \boldsymbol{f}) = \log(p(\boldsymbol{e}))$
- $\lambda_1 = \lambda_2 = 1$

Log-linear models were originally trained with the maximum likelihood criterion, which precisely makes them equivalent to maximum entropy models (Berger et al., 1996). More effective training techniques such as minimum error rate training (Och, 2003) were introduced subsequently (see Section 2.8.2). An advantage of minimum error rate training is that the criterion for optimisation and the evaluation metric are consistent. Because minimum error rate training does not require computing a normalisation constant, in practice, SMT models effectively become linear models, with the objective function presented in Equation 2.17:

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} \sum_{m=1}^{M} \lambda_m h_m(\boldsymbol{e}, \boldsymbol{f}) \tag{2.17}$$

2.5 Phrase-Based Translation

So far, we have presented two modelling approaches to machine translation: the original source-channel model and the current log-linear model. We also have presented word alignment models, which were introduced within the source-channel model framework and which are instances of word-based models.

In the phrase-based translation paradigm, the minimal unit of translation consists of phrases. Phrases are sequences of consecutive words, that need not be syntactically or semantically motivated, but nevertheless are used as translation units. Benefits of phrase-based models include:

- effectively disambiguating the translation of a word in a certain local context;
- effective local reordering within a phrase such as the adjective-noun inversion from French to English;
- effective translation of multi-word expressions, such as idioms.

Phrase-based models currently achieve state-of-the-art performance for certain language pairs that do not involve much reordering such as French-English. They can be defined in the source-channel model framework (see Section 2.2) or the log-linear model framework (see Section 2.4). Because the source-channel model is no longer widely used anymore and because it is a special case of a log-linear model, we will focus our presentation on the log-linear model.

There are different variations on the phrase-based translation paradigm. We will focus on two popular approaches, namely the alignment template model (Och et al., 1999; Och and Ney, 2004) and the phrase-based model (Koehn et al., 2003; Koehn, 2010).

2.5.1 Alignment Template Model

The alignment template model uses the log-linear model presented in Equation 2.16 as a starting point and repeated in Equation 2.18:

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{e_1^J} \exp(\sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I)) \tag{2.18}$$

In order to reduce the number of parameters, two latent variables π_1^K and z_1^K are introduced. z_1^K is a sequence of *alignment templates* while π_1^K is a permutation of size K. An alignment template is a triple $(\tilde{F}, \tilde{E}, \tilde{A})$ where $\tilde{F}$ is a sequence of source word classes, $\tilde{E}$ is a sequence of target word classes and $\tilde{A}$ is an alignment between $\tilde{F}$ and $\tilde{E}$. π_1^K together with z_1^K define a source segmentation of f_1^J into source phrases $\tilde{f}_1^K$, a target segmentation of e_1^I into target phrases $\tilde{e}_1^K$ and a bijective mapping between source phrases and target phrases where $\tilde{f}_{\pi_k}$ is mapped to $\tilde{e}_k$ for $k \in [1, K]$. The alignment templates z_1^K are constrained in such a way that the alignment template classes match the word classes. The alignment template translation model is summarised in Figure 2.4.

Using the max approximation and making the feature functions depend on the hidden variables, the translation model can be rewritten in Equation 2.19:

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{e_1^J, z_1^K, \pi_1^K} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I, \pi_1^K, z_1^K)\right) \tag{2.19}$$

2.5.2 Phrase-Based Model

The phrase-based model is similar to the alignment template model but does not make use of source and target word classes. Again, we use the latent variables π_1^K and z_1^K. This time, z_k is defined as a triple $(\tilde{f}, \tilde{e}, \tilde{A})$ where $\tilde{f}$ is a sequence of source words, $\tilde{e}$ is a sequence of target words and $\tilde{A}$ is an alignment between the words in $\tilde{f}$ and $\tilde{e}$. The reason for using the alignment information between phrase pairs is to be able to compute the lexical feature (see Section 2.6.5). Because the lexical feature can be computed by other means than this alignment information (see again Section 2.6.5), it is also possible to simply define z_k as a phrase pair.

We have presented two variants of the phrase-based model. We will now describe how to obtain the phrase pairs used for the latent variable z.

2.5.3 Phrase Pair Extraction

A preliminary step to phrase pair extraction is to obtain word aligned parallel text. One possibility is to train source-to-target and target-to-source word alignment models, obtain Viterbi alignments in both directions, and apply symmetrisation heuristics, as described in Section 2.3.2. Then phrase pairs are extracted from each word aligned sentence pair.

$$\begin{tikzcd}
f_1 \arrow[d] & f_2 \arrow[d] & f_3 \arrow[d] & f_4 \arrow[d] & f_5 \arrow[d] & f_6 \arrow[d] & f_7 \arrow[d] \\
\overline{f}_1 \arrow[dr] & \overline{f}_2 \arrow[d] & \overline{f}_3 \arrow[d] & \overline{f}_4 \arrow[d] & \overline{f}_5 \arrow[d] & \overline{f}_6 \arrow[dl] & \overline{f}_7 \arrow[dll] \\
 & \widetilde{f}_1 \arrow[dr, "z_1"'] & \widetilde{f}_2 \arrow[dl, "z_2"] & \widetilde{f}_3 \arrow[dr, "z_3"'] & \widetilde{f}_4 \arrow[dl, "z_4"] & & \\
 & \widetilde{e}_1 \arrow[dl] \arrow[d] & \widetilde{e}_2 \arrow[d] & \widetilde{e}_3 \arrow[d] \arrow[dr] & \widetilde{e}_4 \arrow[dr] & & \\
\overline{e}_1 \arrow[d] & \overline{e}_2 \arrow[d] & \overline{e}_3 \arrow[d] & \overline{e}_4 \arrow[d] & \overline{e}_5 \arrow[d] & \overline{e}_6 \arrow[d] & \\
e_1 & e_2 & e_3 & e_4 & e_5 & e_6 &
\end{tikzcd}$$

Figure 2.4: Alignment template translation process. Adapted from (Och and Ney, 2004). The source word sequence f_1^7 is first transformed into a source class sequence $\overline{f}_1^7$. The source classes are then segmented into source phrases $\widetilde{f}_1^4$. The source phrases are then reordered and aligned to target phrases $\widetilde{e}_1^4$. For example, the source phrase $\widetilde{f}_1$ is aligned to the target phrase $\widetilde{e}_2$ through z_1. This means that the permutation π_1^4 has value $\pi_2 = 1$. This also means that z_1 define a word alignment between the source words f_1 and f_2 and the target word e_3. Finally, the target phrases $\widetilde{e}_1^4$, which encode the target class sequence $\overline{e}_1^6$ generate the target word sequence e_1^6.

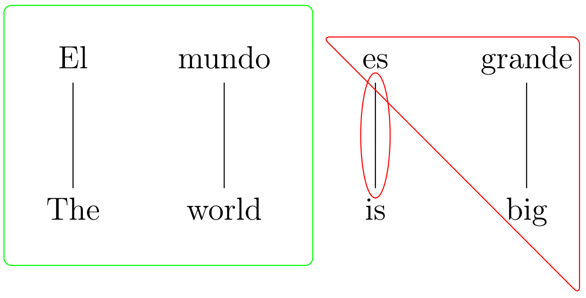

Figure 2.5: Rule extraction for a sentence pair. For example, the phrase (El mundo, The world) is extracted. The phrase pair (es grande, big) is not extracted because it is not consistent with the alignment.

Let us consider a sentence pair $(\boldsymbol{f}, \boldsymbol{e})$ and an alignment $\boldsymbol{a}$. We extract all phrase pairs that are *consistent* with the alignment. This means that we extract all phrase pairs $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$ that satisfy Equation 2.20 and Equation 2.21:

$$\forall (j, i) \in \boldsymbol{a}, (j \in [j_1, j_2] \Leftrightarrow i \in [i_1, i_2]) \tag{2.20}$$

$$[j_1, j_2] \times [i_1, i_2] \cap \boldsymbol{a} \neq \emptyset \tag{2.21}$$

Equation 2.20 requires that no alignment link be between a word inside the phrase pair and a word outside the phrase pair. Equation 2.21 requires that there be at least one alignment link between a source word in the source phrase and a target word in the target phrase. For example, in Figure 2.5, the phrase pair ⟨*El mundo*, *The world*⟩ is extracted while the phrase pair ⟨*es grande*, *big*⟩ is not because the word *es* (*is*) is aligned outside the phrase pair. Note that the consistency constraint sometimes refers to only Equation 2.20 rather than both Equation 2.20 and Equation 2.21.

2.5.4 Phrase-Based Decoding

We have introduced two types of phrase-based models and described a technique to extract phrase-pairs, which are an essential component of these models. We will now describe the decoding process, which is an effective means to obtain the optimal translation of a source sentence.

We first describe a naive strategy for decoding, in order to motivate the need for a more efficient decoder. A naive decoder may follow the following steps, given a source sentence $\boldsymbol{f}$ of length J to be translated:

- Consider all possible segmentations of $\boldsymbol{f}$ into source phrases. There are 2^{J-1} such segmentations.
- For each segmentation, consider all possible permutations of the source phrases. For a segmentation of size K, there are $K!$ such permutations.
- For each permutation of the source phrases, consider all translations of each source phrase, and concatenate the target phrases according to the permutation. The source phrases translations are given by the phrase pairs extracted from the training data (see Section 2.5.3). If there are 10 translations per source phrase, we obtain 10^K possible translation (for the segmentation and permutation considered).
- Rank the translations by their score and pick the highest scoring translation.

We can see that the search space is too large for this naive approach to be feasible. We now present a practical solution to the decoding process in phrase-based translation. We will first introduce the translation process. Then, we will describe how translation hypotheses are built incrementally. We will then motivate the need for pruning and how pruning is carried out. Finally, we will describe how future cost estimation may reduce search errors.

Translation Process Given a source sentence, the translation process is to iteratively pick a source phrase that has not been translated yet, translate that source phrase into a target phrase and append the target phrase to the translation, and repeat this process until the source sentence has been entirely covered by source phrases. While the process is not complete, the concatenation of target phrases is called a *partial hypothesis*.

Hypothesis Expansion The decoding process starts from an initial empty partial hypothesis. This empty partial hypothesis is extended by picking source phrases, appending their translations to the empty hypothesis. At this stage, we have obtained several partial hypotheses. The partial hypotheses are repeatedly extended until all source words have been covered. Partial hypotheses are represented by states that contain the information necessary to compute the cost of an extension. If we use an n-gram language model as a feature, the state will encode the cost of the partial hypothesis and the last $n-1$ words of the partial hypothesis.

Hypothesis Recombination When two partial hypotheses share the same $n - 1$ words, only the partial hypothesis with the lower cost can lead to the best final hypothesis. Therefore, the partial hypothesis with higher cost can be discarded, or alternatively, it is possible to make these two partial hypotheses share the same state for rescoring purposes.

Stack Based Decoding and Pruning The decoding search space is very large as seen above. Approximations therefore need to be made for an effective search. The partial hypotheses are grouped in *stacks* by the number of source words covered. This allows pruning. Each time a hypothesis expansion produces a hypothesis that belongs to a certain stack, that stack is pruned. There are two commonly used types of pruning: histogram pruning and threshold pruning (Koehn, 2010). Histogram pruning enforces a maximum number of partial hypotheses in each stack. Threshold pruning examines the cost of the best partial hypothesis in a stack and discards all partial hypotheses in that stack whose cost is greater than the best cost plus a threshold. Histogram pruning provides a simple guarantee in terms of computational complexity. On the other hand, because it relies on cost, threshold pruning ensures a consistent quality for partial hypotheses that survive the pruning threshold.

Future Cost Partial hypotheses that cover the same number of source words are grouped together for pruning purposes. However, their cost may not be directly comparable, for example partial hypotheses that correspond to the translation of frequent words in the source might have a smaller cost than partial hypotheses that correspond to the translation of rare words in the source. To address this issue, a future cost that represents how difficult it is to translate the rest of the sentence is added to the model cost of each partial hypothesis.

2.6 Hierarchical Phrase-Based Translation

In the previous section, we have described the phrase-based translation paradigm. In this section, we present the hierarchical phrase-based translation model. This model relies on the synchronous context free grammar formalism. Reordering between source and target languages is modelled by the grammar rather than by an *ad hoc* reordering model, although using both the grammar and a reordering model may be beneficial (Huck et al., 2013). A closely related formalism, inversion transduction grammars, was previously introduced (Wu, 1995, 1997). However, because hierarchical phrase-based grammar only contain lexicalised rules, translation decoding is more practical. An alternative to hierarchical phrase-based translation that also models discontinuous phrases but does not use the same grammar formalism has also been introduced recently (Galley and Manning, 2010).

2.6.1 Introduction and Motivation

Phrase-based models generally impose a limit on the size of the phrase pairs extracted while, in decoding, phrases are typically reordered within a certain limit. These restrictions may be problematic for language pairs such as German-English or Chinese-English that allow arbitrary reordering in some instances. Hierarchical phrase-based translation was introduced in order to overcome the reordering limitations from phrase-based models (Chiang, 2005, 2007). For example, the Chinese sentence with English gloss in Figure 2.6 requires "nested" reordering (Chiang, 2007):

- The phrase *with North Korea have diplomatic relations* must be reordered into the phrase *have diplomatic relations with North Korea.*
- The phrase *few countries one of* must be reordered into the phrase *one of (the) few countries.*
- After the two previous segments are reordered, *have diplomatic relations with North Korea that one of the few countries* must be reordered into *one of the few countries that have diplomatic relations with North Korea.*

Phrase-based systems can model this type of movement but they require very long phrase pairs, which is impractical to rely upon because of data sparsity. On the other hand, hierarchical phrase-based grammars do model this type of movement using shorter phrase pairs but with more complex rules.

2.6.2 Hierarchical Grammar

2.6.2.1 Hierarchical Grammar Definition

A hierarchical phrase-based grammar, or hierarchical grammar, or Hiero grammar, which is a particular instance of a synchronous context free gram-

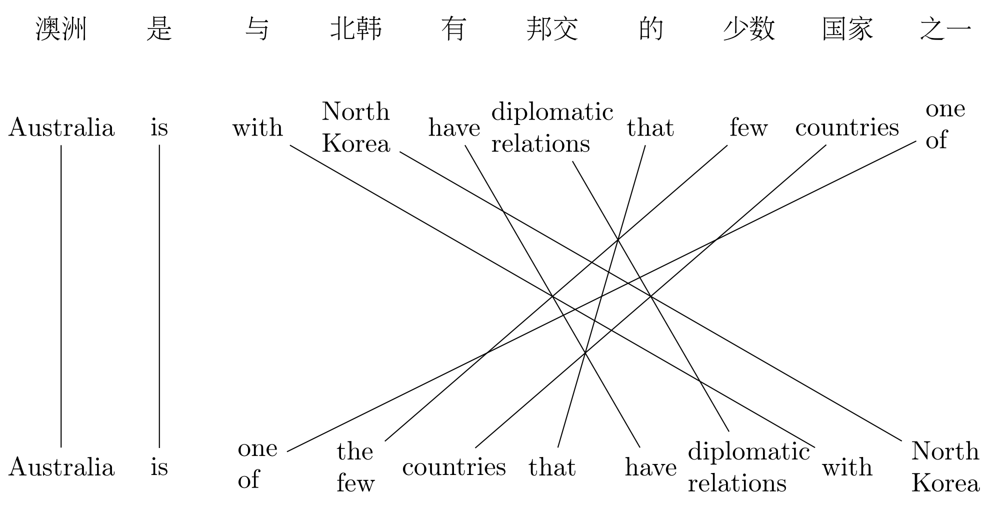

Figure 2.6: Example of Chinese sentence that needs nested reordering in order to be translated into English. Adapted from (Chiang, 2007).

mar (Lewis II and Stearns, 1968; Aho and Ullman, 1969), is a set of rewrite rules of the following type:

$$X \to \langle \gamma, \alpha, \sim \rangle$$

where X is a nonterminal, γ and α are sequences of terminals and nonterminals and $\sim$ is an alignment between nonterminals. Terminals that appear in γ are words in the source language while terminals that appear in α are words in the target language. Nonterminals are chosen from a finite set disjoint from the set of terminals. The alignment between nonterminals indicates which nonterminals in the source and target languages correspond to each other. The alignment $\sim$ can be written with a set of matching indices.

2.6.2.2 Types of Rules

A rule may or may not contain any nonterminals. A rule without nonterminals (on the right hand side) is called a *phrase-based rule*. A rule with nonterminals is called a *hierarchical rule*. A hierarchical grammar also usu-

$$
\begin{aligned}
R_1 &: S \rightarrow \langle X, X \rangle \\
R_2 &: S \rightarrow \langle SX, SX \rangle \\
R_3 &: X \rightarrow \langle X_1 \text{ 的 } X_2,\ \text{the } X_2 \text{ that } X_1 \rangle \\
R_4 &: X \rightarrow \langle X \text{ 之一},\ \text{one of } X \rangle \\
R_5 &: X \rightarrow \langle \text{与 } X_1 \text{ 有 } X_2,\ \text{have } X_2 \text{ with } X_1 \rangle \\
R_6 &: X \rightarrow \langle \text{澳洲},\ \text{Australia} \rangle \\
R_7 &: X \rightarrow \langle \text{是},\ \text{is} \rangle \\
R_8 &: X \rightarrow \langle \text{北韩},\ \text{North Korea} \rangle \\
R_9 &: X \rightarrow \langle \text{邦交},\ \text{diplomatic relations} \rangle \\
R_{10} &: X \rightarrow \langle \text{少数 国家},\ \text{few countries} \rangle
\end{aligned}
$$

Figure 2.7: Example of hierarchical grammar. Adapted from (Chiang, 2007). With this grammar, it is possible to obtain a derivation for the sentence pair from Figure 2.6. The derivation is shown in Figure 2.8.

ally contains the following rules called *glue* rules:

$$
\begin{aligned}
S \rightarrow& \langle X, X \rangle \\
S \rightarrow& \langle SX, SX \rangle
\end{aligned}
$$

with S the start symbol. The first glue rule is necessary to be able to start a derivation. The second glue rule allows concatenation of phrase-based or hierarchical rules.

2.6.2.3 Hierarchical Grammar Example

Let us consider for example the grammar in Figure 2.7 where each rewrite rule is given a name R_i. With this grammar, it is possible to write a derivation, i.e. a sequence of rules, that rewrites the start symbol S into the sentence pair presented in Section 2.6.1 (Chiang, 2007). For example we can apply the derivation $R_2, R_2, R_1, R_6, R_7, R_4, R_3, R_{10}, R_5, R_9, R_8$ as in Figure 2.8.

$$
\begin{aligned}
S &\to \langle SX, SX\rangle \\
&\to \langle SX_1X_2, SX_1X_2\rangle \\
&\to \langle X_1X_2X_3, X_1X_2X_3\rangle \\
&\to \langle \text{澳洲 } X_2X_3, \text{Australia } X_2X_3\rangle \\
&\to \langle \text{澳洲 是 } X_3, \text{Australia is } X_3\rangle \\
&\to \langle \text{澳洲 是 } X \text{ 之一}, \text{Australia is one of } X\rangle \\
&\to \langle \text{澳洲 是 } X_1 \text{ 的 } X_2 \text{ zhiyi}, \text{Australia is one of the } X_2 \text{ that } X_1\rangle \\
&\to \langle \text{澳洲 是 } X_1 \text{ 的 少数 国家 之一}, \text{Australia is one of the few countries that } X_1\rangle \\
&\to \langle \text{澳洲 是 与 } X_1 \text{ 有 } X_2 \text{ 的 少数 国家 之一}, \\
&\quad \text{Australia is one of the few countries that have } X_2 \text{ with } X_1\rangle \\
&\to \langle \text{澳洲 是 与 } X_1 \text{ 有 邦交 的 少数 国家 之一}, \\
&\quad \text{Australia is one of the few countries that have diplomatic relations with } X_1\rangle \\
&\to \langle \text{澳洲 是 与 北韩 有 邦交 的 少数 国家 之一}, \\
&\quad \text{Australia is one of the few countries that have diplomatic relations with North Korea}\rangle
\end{aligned}
$$

Figure 2.8: Example of hierarchical grammar derivation. Adapted from (Chiang, 2007). A derivation is simply a sequence of rules that rewrite the start symbol S into a sentence pair. This particular derivation produces the sentence pair from Figure 2.6.

2.6.2.4 Constraints on Hierarchical Grammars

The definition given for a hierarchical grammar is very general. In practice, systems impose constraints on the types of rule the grammar contains for efficiency reasons. We first introduce the concept of rule pattern (Iglesias et al., 2009a) in order to describe these constraints in terms of patterns. A rule pattern is simply a pair of regular expressions that match the source and target sides of a hierarchical rule. For example, the rule $X \rightarrow \langle \textit{Buenas tardes } X, \textit{ Good afternoon } X\rangle$ has a rule pattern $\langle \Sigma^+ X, \Psi^+ X\rangle$ where Σ is the source vocabulary and Ψ is the target vocabulary. For ease of notation, we use the notation w for either Σ^+ or Ψ^+. Thus w simply represents a sequence of terminals. In our example, the pattern for the rule $X \rightarrow \langle \textit{Buenas tardes } X, \textit{ Good afternoon } X\rangle$ is $\langle wX, wX\rangle$. Chiang (2007) defines the following set of pattern-related constraints that must be satisfied by the rules in a hierarchical grammar:

- If a rule $X \rightarrow \langle \gamma, \alpha\rangle$ has a pattern $\langle w, w\rangle$, then $|\gamma| \leq 10$ and $|\alpha| \leq 10$.
- A rule $X \rightarrow \langle \gamma, \alpha\rangle$ must satisfy $|\gamma| \leq 5$.
- Rules have at most 2 nonterminals.
- The source side of a rule cannot have adjacent nonterminals. Setiawan and Resnik (2010) relax this constraint in order to model Chinese-English reordering phenomena that may be not always captured in training because of data sparsity. Note that the target side can still have adjacent nonterminals.

More fine-grained constraints can be defined on rule patterns, which are obtained from rules by replacing terminal sequences by the placeholder w. For example, we use the configuration described in Table 2.1 for all translation experiments in this thesis, unless specified otherwise. This grammar was obtained following a greedy strategy of adding in turn the most beneficial patterns for Arabic-English translation.

Another restriction on hierarchical grammars is the extent of reordering allowed. de Gispert et al. (2010a) investigate the use of shallow-N grammars that precisely control the depth of reordering in translation. We describe here shallow-1 grammars (Iglesias et al., 2009a; de Gispert et al., 2010a) as they will be used for translation experiments in this thesis. Shallow-1 grammars allow only one level of reordering, they do not allow nested reordering as in

⟨**source** , **target**⟩	**include**	⟨**source** , **target**⟩	**include**
$\langle w~X~,~w~X\rangle$	**no**	$\langle X~w~,~w~X\rangle$	yes
$\langle w~X~,~X~w\rangle$	yes	$\langle X~w~,~w~X~w\rangle$	yes
$\langle w~X~,~w~X~w\rangle$	yes	$\langle X~w~,~X~w\rangle$	**no**
$\langle w~X~w~,~w~X\rangle$	yes	$\langle w~X~w~,~X~w\rangle$	yes
$\langle w~X~w~,~w~X~w\rangle$	yes		
$\langle X1~w~X2~,~w~X1~w~X2\rangle$	**no**	$\langle X2~w~X1~,~w~X1~w~X2\rangle$	yes
$\langle X1~w~X2~,~w~X1~w~X2~w\rangle$	**no**	$\langle X2~w~X1~,~w~X1~w~X2~w\rangle$	yes
$\langle X1~w~X2~,~w~X1~X2\rangle$	**no**	$\langle X2~w~X1~,~w~X1~X2\rangle$	yes
$\langle X1~w~X2~,~w~X1~X2~w\rangle$	**no**	$\langle X2~w~X1~,~w~X1~X2~w\rangle$	yes
$\langle X1~w~X2~,~X1~w~X2\rangle$	**no**	$\langle X2~w~X1~,~X1~w~X2\rangle$	yes
$\langle X1~w~X2~,~X1~w~X2~w\rangle$	**no**	$\langle X2~w~X1~,~X1~w~X2~w\rangle$	yes
$\langle X1~w~X2~,~X1~X2~w\rangle$	**no**	$\langle X2~w~X1~,~X1~X2~w\rangle$	yes
$\langle w~X1~w~X2~,~w~X1~w~X2\rangle$	**no**	$\langle w~X2~w~X1~,~w~X1~w~X2\rangle$	yes
$\langle w~X1~w~X2~,~w~X1~w~X2~w\rangle$	yes	$\langle w~X2~w~X1~,~w~X1~w~X2~w\rangle$	yes
$\langle w~X1~w~X2~,~w~X1~X2\rangle$	yes	$\langle w~X2~w~X1~,~w~X1~X2\rangle$	yes
$\langle w~X1~w~X2~,~w~X1~X2~w\rangle$	yes	$\langle w~X2~w~X1~,~w~X1~X2~w\rangle$	yes
$\langle w~X1~w~X2~,~X1~w~X2\rangle$	yes	$\langle w~X2~w~X1~,~X1~w~X2\rangle$	yes
$\langle w~X1~w~X2~,~X1~w~X2~w\rangle$	yes	$\langle w~X2~w~X1~,~X1~w~X2~w\rangle$	yes
$\langle w~X1~w~X2~,~X1~X2~w\rangle$	yes	$\langle w~X2~w~X1~,~X1~X2~w\rangle$	yes
$\langle X1~w~X2~w~,~w~X1~w~X2\rangle$	yes	$\langle X2~w~X1~w~,~w~X1~w~X2\rangle$	yes
$\langle X1~w~X2~w~,~w~X1~w~X2~w\rangle$	yes	$\langle X2~w~X1~w~,~w~X1~w~X2~w\rangle$	yes
$\langle X1~w~X2~w~,~w~X1~X2\rangle$	yes	$\langle X2~w~X1~w~,~w~X1~X2\rangle$	yes
$\langle X1~w~X2~w~,~w~X1~X2~w\rangle$	yes	$\langle X2~w~X1~w~,~w~X1~X2~w\rangle$	yes
$\langle X1~w~X2~w~,~X1~w~X2\rangle$	yes	$\langle X2~w~X1~w~,~X1~w~X2\rangle$	yes
$\langle X1~w~X2~w~,~X1~w~X2~w\rangle$	**no**	$\langle X2~w~X1~w~,~X1~w~X2~w\rangle$	yes
$\langle X1~w~X2~w~,~X1~X2~w\rangle$	yes	$\langle X2~w~X1~w~,~X1~X2~w\rangle$	yes
$\langle w~X1~w~X2~w~,~w~X1~w~X2\rangle$	**no**	$\langle w~X2~w~X1~w~,~w~X1~w~X2\rangle$	yes
$\langle w~X1~w~X2~w~,~w~X1~w~X2~w\rangle$	**no**	$\langle w~X2~w~X1~w~,~w~X1~w~X2~w\rangle$	yes
$\langle w~X1~w~X2~w~,~w~X1~X2\rangle$	**no**	$\langle w~X2~w~X1~w~,~w~X1~X2\rangle$	yes
$\langle w~X1~w~X2~w~,~w~X1~X2~w\rangle$	**no**	$\langle w~X2~w~X1~w~,~w~X1~X2~w\rangle$	yes
$\langle w~X1~w~X2~w~,~X1~w~X2\rangle$	**no**	$\langle w~X2~w~X1~w~,~X1~w~X2\rangle$	yes
$\langle w~X1~w~X2~w~,~X1~w~X2~w\rangle$	**no**	$\langle w~X2~w~X1~w~,~X1~w~X2~w\rangle$	yes
$\langle w~X1~w~X2~w~,~X1~X2~w\rangle$	**no**	$\langle w~X2~w~X1~w~,~X1~X2~w\rangle$	yes

Table 2.1: Rule patterns included in a baseline hierarchical grammar. A pattern is obtained from a rule by replacing terminal sequences by the placeholder w. The decision whether to include each pattern was based on preliminary experiments in Arabic-English. Most beneficial patterns were added incrementally. Unless specified otherwise, this configuration is used in all subsequent translation experiments.

the example presented in Section 2.6.1. A shallow-1 grammar is defined as follows:

$$
\begin{aligned}
&S \rightarrow \langle X, X \rangle \\
&S \rightarrow \langle SX, SX \rangle \\
&X \rightarrow \langle \gamma_s, \alpha_s \rangle (\gamma_s, \alpha_s \in (T \cup \{V\})^+) \\
&X \rightarrow \langle V, V \rangle \\
&V \rightarrow \langle s, t \rangle (s \in T^+, t \in T^*)
\end{aligned}
$$

where S is the start symbol, T is the set of terminals and V is the set of nonterminals. There are two nonterminals apart from the start symbol: X and V. The rule type $X \rightarrow \langle \gamma_s, \alpha_s \rangle$ corresponds to all hierarchical rules. It is possible to apply this type of rule only once in any derivation. Indeed, the right hand side contains only one type of nonterminal, V, which can be rewritten only with a phrasal rule corresponding to the line $V \rightarrow \langle s, t \rangle$. Note that for rules of the type $V \rightarrow \langle s, t \rangle$, t can be the empty word, thus these rules, called *deletion rules*, allow deletion on the target side. Shallow-1 grammar are used for language pairs that do not present much reordering. Shallow-1 grammars were previously shown to work as well as full hierarchical grammars for the Arabic-English language pair (Iglesias et al., 2009a) and for the Spanish-English language pair (Iglesias et al., 2009c). In addition, shallow-1 grammars reduce the search space of the decoder greatly, resulting in a much faster decoding time, a reduced memory use and potentially fewer search errors under the translation grammar.

2.6.3 Log-linear Model for Hierarchical Phrase-Based Translation

We now define in more detail the log-linear model for hierarchical translation, which usually makes a maximum (max) approximation, i.e. replaces a sum by the maximum term in the sum and assumes that the other terms are negligible. We follow the original description (Chiang, 2007). For a sentence pair $(\boldsymbol{f}, \boldsymbol{e})$, let us define $\mathcal{D}$ the set of possible derivations D of this sentence pair under a hierarchical grammar. We will use the following notation for a derivation D:

- the foreign yield $\boldsymbol{f}$. We define $f(D) = \boldsymbol{f}$

- the English yield $\boldsymbol{e}$. We define $e(D) = \boldsymbol{e}$

We can now derive the log-linear model for hierarchical translation:

$$\begin{aligned}
\hat{\mathbf{e}} &= \operatorname*{argmax}_{\boldsymbol{e}} p(\boldsymbol{e} \mid \boldsymbol{f}) \\
&= \operatorname*{argmax}_{\boldsymbol{e}} \sum_{D \in \mathcal{D}} p(D, \boldsymbol{e} \mid \boldsymbol{f}) \text{ (marginalisation)} \\
&= \operatorname*{argmax}_{\boldsymbol{e}} \operatorname*{argmax}_{D \in \mathcal{D}} p(D, \boldsymbol{e} \mid \boldsymbol{f}) \text{ (max approximation)} \\
&= e(\operatorname*{argmax}_{\mathbf{e}, D \in \mathcal{D}} p(D, \mathbf{e} | \mathbf{f})) \\
&= e(\operatorname*{argmax}_{D | f(D) = \mathbf{f}} p(D)) \qquad (2.22)
\end{aligned}$$

Thanks to the max approximation, the distribution over derivations instead of the distribution over English sentences is modelled log-linearly and we obtain finally the following decoding equation:

$$\hat{\mathbf{e}} = e(\operatorname*{argmax}_{D | f(D) = \mathbf{f}} \exp(\sum_{m=1}^{M} \lambda_m h_m(D))) \qquad (2.23)$$

One of the features, the language model, plays a particular role. The language model feature can be written as:

$$h_M(D) = p_{LM}(e(D)) \qquad (2.24)$$

where M is the index of the language model feature, p_{LM} is the language model and $e(D)$ is the English yield of the derivation D. It is not possible to compute the language model using dynamic programming since the language model needs context in order to be computed, therefore the language model feature is typically computed after a parsing step.

Note that Equation 2.23 is an approximation and that there have been attempts to perform marginalisation over the latent variable D while keeping the translation process tractable (Blunsom et al., 2008a; de Gispert et al., 2010a). This can give gains over the max approximation, although subsequent rescoring steps (see Section 2.10) can produce similar performance (de Gispert et al., 2010a).

2.6.4 Rule Extraction

We have so far given the definition of a hierarchical grammar and explained how it is used with statistical models. It is also necessary to extract an appropriate grammar, defined by its rules. The extraction is performed on a parallel corpus. The parallel corpus is first word-aligned, then rules are extracted from the alignment.

We first extract phrase pairs as described in Section 2.5.3. For each extracted phrase pair $\langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle$, we define the following rule: $X \rightarrow \langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle$. These rules are called *initial rules*. We extend the set of initial rules with the following recursion: given a rule $X \rightarrow \langle \gamma, \alpha \rangle$ and an initial rule $X \rightarrow \langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle$ such that $\gamma = \gamma_1 f_{j_1}^{j_2} \gamma_2$ and $\alpha = \alpha_1 e_{i_1}^{i_2} \alpha_2$, then extract the rule $X \rightarrow \langle \gamma_1 X \gamma_2, \alpha_1 X \alpha_2 \rangle$. Note that γ_1 or γ_2, but not both, can be the empty word, and similarly for α_1 and α_2.

2.6.5 Features

The following features are commonly used in log-linear models for machine translation:

- Source-to-target and target-to-source translation models. As described above, the translation process produces a derivation D. A derivation D can be seen as a sequence of n rules $X \rightarrow \langle \alpha_1, \gamma_1 \rangle, ..., X \rightarrow \langle \alpha_n, \gamma_n \rangle$. Then the source-to-target translation model is simply $\prod_{i=1}^{n} p(\gamma_i | \alpha_i)$ where the $p(\gamma_i | \alpha_i)$ are typically estimated using relative frequency, based on the appearance of phrasal and hierarchical rules in the parallel corpus. The target-to-source translation model is symmetric.

- Source-to-target and target-to-source lexical translation model. Typically, the source-to-target lexical translation model $lx(f_1^J, e_1^I)$ is computed as following for a foreign sentence f_1^J translated into the English sentence e_1^I:

$$lx(f_1^J, e_1^I) = \frac{1}{(I+1)^J} \prod_{j=1}^{J} \sum_{i=0}^{I} p_1(e_i | f_j) \tag{2.25}$$

  where p_1 is the word-to-word translation probability in IBM Model 1 and e_0 the null word. The target-to-source lexical translation model is symmetric. One reason to use lexical models in addition to translation models is that translation models are relatively sparse compared to lexical models, thus lexical models can smooth the translation models.

- Number of uses of the glue rule in a derivation. This feature trades off monotonic translation versus reordering.
- Word insertion penalty. This feature controls the length of the output.
- Rule insertion penalty. This feature controls the number of rules used in a derivation to generate a translation. Using many rules is closer to word-to-word translation while using few rules means that longer phrase pairs are used. This feature is similar to the phrase penalty in phrase-based statistical machine translation (Koehn, 2010).
- Word deletion scale factor. This feature controls the number of times a deletion rule is applied.
- Rule count feature. This feature indicates whether a rule occurs once, twice or more in the parallel data (Bender et al., 2007). Thus it indicates how reliable a rule is.

The feature weights are optimised using minimum error rate training (Och, 2003) under the BLEU score (Papineni et al., 2002) (see also Section 2.8.1).

2.7 Language Modelling

We mentioned in Section 2.6.3 that a language model was one of the features of log-linear models for translation. This feature is critical as it helps to obtain a fluent translation output. A language model is a probability distribution over word sequences. It can be used to assign a probability to a sequence of words or to predict the word most likely to appear in a certain context. Language models have applications in fields where the goal is to produce fluent output, such as automatic speech recognition or machine translation. n-gram language models are typically used because they are robust, can be easily trained on large amounts of data and model local grammatical relationships. Since they do not model the language structure nor long distance relationships between words, work has been conducted (Shen et al., 2008) to overcome this issue. We first review n-gram language models, then review different smoothing techniques and we finally describe in more detail two smoothing techniques that are used for experiments in this thesis: Kneser-Ney smoothing and Stupid Backoff smoothing.

2.7.1 n-gram language models

For simplicity, let us first consider the case of a bigram language model. Let us consider a vocabulary V and the two special symbols `<s>` and `</s>` corresponding respectively to the start-of-sentence symbol and the end-of-sentence symbol. We define $W = V \cup \{\texttt{<s>}, \texttt{</s>}\}$. Let us now define the Markov chain $(X_i)_{i \in \mathbb{N}}$ with values in W and transition probability p. p has the following properties:

- $p(X_0 = \texttt{<s>}) = 1$: we always start a sentence with a start-of-sentence symbol.
- $p(X_{n+1} = \texttt{</s>} \mid X_n = \texttt{</s>}) = 1$: once we reach the end of a sentence, we stay in the end-of-sentence state. This is because we do not consider infinite word sequences.
- $p(X_{n+1} = \texttt{<s>} \mid X_n = \texttt{<s>}) = 0$: we cannot stay in the start-of-sentence state and have to transition to either a word or the end-of-sentence state.

A bigram model is defined by the conditional independence assumptions of the Markov chain (X_i) and the translation probability p. A bigram model will therefore assign the probability $p(\boldsymbol{w})$ to a sequence of words $\boldsymbol{w} = w_1 ... w_n$ in Equation 2.26:

$$p(\boldsymbol{w}) = \prod_{i=1}^{n+1} p(w_i \mid w_{i-1}) \tag{2.26}$$

with the convention $w_0 = \texttt{<s>}$ and $w_{n+1} = \texttt{</s>}$.

An n-gram language model can be defined similarly: this time the random variables X_i take values in W^{n-1} instead of W. An n-gram model will assign the probability $p(\boldsymbol{w})$ to a sequence of words $\boldsymbol{w} = w_1 ... w_n$ in Equation 2.27:

$$p(\mathbf{w}) = \prod_{i=1}^{n+1} p(w_i \mid w_{i-n+1}^{i-1}) \tag{2.27}$$

with the same convention that $w_0 = \texttt{<s>}$ and $w_{n+1} = \texttt{</s>}$. Parameters can be trained using maximum likelihood estimation, so the parameter

$p(w_i|w_{i-n+1}^{i-1})$ is computed in Equation 2.28.

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} \tag{2.28}$$

where $c(.)$ counts the number of occurrences of a particular word sequence in the training data. Maximum likelihood estimation assigns zero probability to unseen events, therefore different smoothing strategies have been explored to address this problem.

2.7.2 Back-off and Interpolated Models

Smoothing strategies for language modelling make use of lower order distributions either by backoff or interpolation. The general form of a backoff model (Chen and Goodman, 1998) is presented in Equation 2.29:

$$p_{\text{backoff}}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \alpha(w_i \mid w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) p_{\text{backoff}}(w_i \mid w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases} \tag{2.29}$$

where c is an occurrence count in the monolingual training corpus for an n-gram. The conditions $c(w_{i-n+1}^{i}) > 0$ and $c(w_{i-n+1}^{i}) = 0$ can be replaced by $c(w_{i-n+1}^{i}) \geq \mathit{cutoff}$ and $c(w_{i-n+1}^{i}) < \mathit{cutoff}$ respectively when n-grams with an occurrence less than the *cutoff* threshold are ignored in the training data.

The general form of an interpolated model (Chen and Goodman, 1998) is presented in Equation 2.30:

$$\begin{aligned} p_{\text{interpolate}}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{\text{ML}}(w_i \mid w_{i-n+1}^{i-1}) + \\ (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{\text{interpolate}}(w_i \mid w_{i-n+2}^{i-1}) \end{aligned} \tag{2.30}$$

where:

- p_{ML} is the maximum likelihood estimate.
- $\lambda_{w_{i-n+1}^{i-1}}$ is the interpolation parameter.

The difference between backoff and interpolated model is that interpolated models make use of lower order distributions even when the n-gram counts are greater than zero. However, an interpolated model can be written in the form of a backoff model. Let us define $\alpha(w_i \mid w_{i-n+1}^{i-1})$ in Equation 2.31:

$$\alpha(w_i \mid w_{i-n+1}^{i-1}) = p_{\text{interpolate}}(w_i \mid w_{i-n+1}^{i-1}) \tag{2.31}$$

and $\gamma(w_{i-n+1}^{i-1})$ in Equation 2.32:

$$\gamma(w_{i-n+1}^{i-1}) = 1 - \lambda_{w_{i-n+1}^{i-1}} \tag{2.32}$$

We can see that by plugging in the definitions of $\alpha(w_i \mid w_{i-n+1}^{i-1})$ and $\gamma(w_{i-n+1}^{i-1})$ in Equation 2.29, it is possible to write an interpolated model as a backoff model. This observation is trivial but it is useful in practice in order to make use of the ARPA file format[3] which only supports back-off models.

2.7.3 Modified Kneser-Ney Smoothing

In this section, we present interpolated modified Kneser-Ney smoothing (Chen and Goodman, 1998), which is the most popular smoothing strategy used for machine translation. The motivation for Kneser-Ney smoothing (Kneser and Ney, 1995) as given by Chen and Goodman (1998) is that the use of lower order distributions for smoothing needs to take into account information from the higher order distribution. For example, let us consider a bigram model. We want to assign a probability to the word *Francisco* given its previous word, say *hello*. If we have not seen the bigram *hello Francisco* in the training corpus, we need to back-off to the unigram *Francisco*. We assume that the unigram *Francisco* is very frequent in our corpus but that it is only seen in the context of the bigram *San Francisco*. If we simply use the maximum likelihood estimate of the unigram *Francisco*, we will obtain a relatively high probability for the bigram *hello Francisco*. This should not be the case because the corpus provides evidence that *Francisco* is very unlikely to follow any other word than *San*.

The modified Kneser-Ney smoothed probability is defined in Equation 2.33.

$$p_{\text{KN}}(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{c(w_{i-n+1}^{i-1})} + \gamma(w_{i-n+1}^{i-1}) p_{\text{KN}}(w_i | w_{i-n+2}^{i-1}) \tag{2.33}$$

where D and γ are defined in Equation 2.34 and Equation 2.35.

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_3 & \text{if } c \geq 3 \end{cases} \tag{2.34}$$

[3] http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{c(w_{i-n+1}^{i-1})} \tag{2.35}$$

N_1, N_2 and N_{3+} are defined in Equation 2.36.

$$\begin{aligned} N_1(w_{i-n+1}^{i-1}\bullet) &= |\{w_i : c(w_{i-n+1} w_i) = 1\}| \\ N_2(w_{i-n+1}^{i-1}\bullet) &= |\{w_i : c(w_{i-n+1} w_i) = 2\}| \\ N_{3+}(w_{i-n+1}^{i-1}\bullet) &= |\{w_i : c(w_{i-n+1} w_i) \geq 3\}| \end{aligned} \tag{2.36}$$

2.7.4 Stupid Backoff Smoothing

The Stupid Backoff smoothing scheme (Brants et al., 2007) is similar to the general backoff smoothing scheme presented in Equation 2.29 and presented in Equation 2.37:

$$s_{\text{stupid backoff}}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \alpha\, s_{\text{stupid backoff}}(w_i \mid w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases} \tag{2.37}$$

We use the notation s to indicate that that the Stupid Backoff Smoothed score is not a probability. We note the following differences with respect to the traditional backoff scheme:

- The non backed off score uses no discounting and simply uses relative frequency.
- The backed off score scaling parameter α is independent of the n-gram history.
- As a result, the Stupid Backoff does not define a probability distribution.

Stupid Backoff smoothing was designed to fit the MapReduce framework and build very large language models (see Section 3.1). This scheme will be used in subsequent translation rescoring experiments throughout this thesis.

2.8 Optimisation

2.8.1 Evaluation Metrics

Evaluating the quality of machine translation output is a challenge in that very often many possible translations are acceptable and of equal quality. Human experts seem to be the most qualified for this task, however their service is expensive and time consuming. Automatic metrics have been therefore designed in order to allow rapid development and comparison of machine translation systems. We describe the BLEU metric since this is the most widely used metric and because we use it extensively in our experiments. We refer to publications for alternative metrics such as METEOR (Banerjee and Lavie, 2005), NIST (Doddington, 2002) or TER (Snover et al., 2006).

The BiLingual Evaluation Understudy (BLEU) score (Papineni et al., 2002) is defined as the geometric mean of n-gram modified precisions with respect to one or several translation references times a brevity penalty. More precisely, given S input sentences $s_1, ..., s_S$, we consider a set of S candidate translations $c_1, ..., c_S$. For each input sentence s_i, there is a finite number R_i of reference translations $r_{i,1}, ..., r_{i,R_i}$. The reference translation that is closest in length to the candidate translation c_i is denoted r_{c_i}. The BLEU metric is defined as following:

$$\text{BLEU}(c_1, ..., c_S, r_{1,1}, ...r_{1,R_1}, ..., r_{S,1}, ..., r_{S,R_S}) = \text{BP} \prod_{n=1}^{N} p_n^{\frac{1}{N}} \tag{2.38}$$

BP is the brevity penalty and is defined as:

$$\text{BP} = \exp(\min(0, 1 - \frac{\sum_{i=1}^{S} |r_{c_i}|}{\sum_{i=1}^{S} |c_i|})) \tag{2.39}$$

p_n is the modified n-gram precision and is defined as

$$\frac{\sum_{i=1}^{S} \sum_{g \in c_i} count_{clip}(g, c_i, r_{i,1}, ..., r_{i,R_i})}{\sum_{i=1}^{S} \sum_{g \in c_i} count(g, c_i)} \tag{2.40}$$

where g is an n-gram, $count(g, c_i)$ is the number of times g occurs in c_i and

$$count_{clip}(g, c_i, r_{i,1}, .., r_{i,R_i}) = \min(count(g, c_i), \max(count(g, r_{i,1}), .., count(g, r_{i,R_i}))) \tag{2.41}$$

The maximum n-gram order is usually $N = 4$. The BLEU metric can be computed efficiently, is language independent and correlates well with human judgements, especially for statistical translation systems.

2.8.2 Minimum Error Rate Training

Minimum error rate training (Och, 2003) is a method to optimise the feature weights in a log-linear model for translation with respect to a particular evaluation metric. We present this method assuming that the evaluation metric is BLEU. The objective function is defined as follows:

$$\hat{\lambda}_1^M = \underset{\lambda_1^M}{\operatorname{argmax}} \operatorname{BLEU}(c(\lambda_1^M), \mathbf{r}) \tag{2.42}$$

where $c(\lambda_1^M)$ are the candidate translations obtained using feature weights λ_1^M and $\mathbf{r}$ are the reference translations. Since computing $c(\lambda_1^M)$ requires computing the most likely translations given a set of features, this optimisation problem is complex: it requires two argmax operations. Och therefore approximates the search space of translations with n-best lists given by a decoder. The optimisation problem is then solved by computing, along a particular direction in the feature space, intervals where the BLEU function is constant and choosing for each direction the optimal feature set. It was found that optimising feature weights in a consistent way with the evaluation metric, for example BLEU, could improve the translation quality as measured by this metric significantly.

2.9 Decoding with Finite State Transducers

We use HiFST, a decoder based on finite state transducers (Iglesias et al., 2009b; Iglesias, 2010). The decoder solves Equation 2.23 in three steps. In the first step, the source sentence is parsed using a variant of the CYK algorithm (Chappelier and Rajman, 1998). Each cell in the CYK grid contains a nonterminal symbol and a span represented by a start and end index and backpointers to other cells in the grid. In the second step, a lattice (Ueffing et al., 2002), i.e. a compact representation of the possible translations as a directed acyclic graph where edges are words, is recursively built by following the backpointers in the grid. In the third step, a language model represented as an acceptor is composed with the original lattice to give translations a language model score.

2.10 Lattice Rescoring

An advantage of the HiFST decoder (see Section 2.9) is that it directly generates translation lattices that can be rescored in order to improve translation quality. Rescoring methods are used in machine translation when their integration into the decoder is challenging in terms of memory use. 5-gram language model rescoring and lattice minimum Bayes' risk rescoring are two such methods. On the other hand, if scalability issues can be overcome, integrating complex models into first-pass decoding may provide a simpler and easier to use system. Furthermore, because larger parts of the decoding search space may be explored, single pass methods may also provide translation quality gains, however this is not guaranteed unless there is no pruning during decoding.

2.10.1 5-gram Language Model Lattice Rescoring

Brants et al. (2007) showed that using vast amounts of data for training language models can improve translation. However, using high order n-gram language models trained on a large dataset in decoding can create memory problems. A solution is to use such models to rescore the output lattice of a first-pass translation. These first-pass lattices are generated using a lower order language model.

The language model we use for rescoring is a 5-gram zero cutoff Stupid Backoff language model (see Section 2.7.2). To build this model, n-grams of order up to 5 and containing words in the target side of the parallel corpus are extracted in parallel from the monolingual data, resulting in an n-gram count file. Then, n-grams from the first-pass translation lattice are extracted using counting transducers (Allauzen et al., 2003). Finally, n-grams from the count file are filtered according to the n-grams from the first-pass translation lattice. A sentence specific Stupid Backoff (Brants et al., 2007) 5-gram language model is built using these n-grams and constitutes a feature along with the first-pass language model, the translation model, and a length penalty in a log-linear model that is used for lattice rescoring. The weights are optimised for BLEU on a development set.

2.10.2 Lattice Minimum Bayes' Risk Rescoring

In Section 2.2, we have presented the general objective function for translation in Equation 2.43 as a decision that minimises the number of errors (Bishop, 2006, p. 39-40):

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} p(\boldsymbol{e} \mid \boldsymbol{f}) \tag{2.43}$$

This objective function is a particular case of Minimum Bayes-Risk (MBR) decoding for SMT (Kumar and Byrne, 2004). MBR takes the general form in Equation 2.44:

$$\hat{\boldsymbol{e}} = \operatorname*{argmin}_{\boldsymbol{e}' \in \mathcal{E}} \sum_{\boldsymbol{e} \in \mathcal{E}} L(\boldsymbol{e}, \boldsymbol{e}') P(\boldsymbol{e} \mid \boldsymbol{f}) \tag{2.44}$$

where:

- $\boldsymbol{f}$ is the source sentence
- $\boldsymbol{e}$ and $\boldsymbol{e}'$ are target sentences
- $\mathcal{E}$ is the set of possible translations
- L is a loss function

If the loss function L is defined as a zero-one loss function as in Equation 2.45:

$$\forall \boldsymbol{e}, \boldsymbol{e}' \ L(\boldsymbol{e}, \boldsymbol{e}') = \begin{cases} -1 & \text{if } \boldsymbol{e} = \boldsymbol{e}' \\ 0 & \text{otherwise} \end{cases} \tag{2.45}$$

then we obtain the objective defined in Equation 2.43. In that sense, the minimum error objective is a special case of the MBR objective. If the translation quality is measure by BLEU, then an appropriate function for L is 1−SBLEU where SBLEU is the sentence level BLEU score, defined as BLEU with a corpus containing only one sentence pair. Kumar and Byrne (2004) use this definition of SBLEU for experiments on n-best list MBR rescoring. Even though SBLEU between two hypotheses might be zero, it is less likely that a particular hypothesis have an expected loss of zero. For other applications such as metrics evaluation, it is beneficial to introduce smoothing for sentence-level BLEU (Lin and Och, 2004). Kumar and Byrne (2004) demonstrate gains in translation quality when rescoring n-best lists with the MBR objective function. Gains measured by the metric used to define the loss function are of course more important.

n-best list MBR rescoring requires $O(n)$ sentence-level BLEU computations for each hypothesis, so $O(n^2)$ sentence-level BLEU computations for each source sentence. This restricts the depth of the n-best list; for example Kumar and Byrne (2004) use a 1000-best list but n could be a very large number. Tromble et al. (2008) remedy this issue by introducing a lattice minimum Bayes' risk decoding technique where $\mathcal{E}$ is a lattice of possible translations rather than an n-best list. They use Equation 2.46 as an approximation of Equation 2.44:

$$\hat{e} = \operatorname*{argmax}_{e' \in \mathcal{E}} \left(\theta_0 |e'| + \sum_{u \in \mathcal{N}} \theta_u \#_u(e') p(u|\mathcal{E}) \right) \tag{2.46}$$

where

- $\mathcal{N}$ is the set of all n-grams in the lattice $\mathcal{E}$
- θ_u is an n-gram specific constant
- $\#_u(e')$ is the number of times u occurs in e'
- $p(u|\mathcal{E})$ is the posterior probability of the n-gram u in the lattice $\mathcal{E}$

Decoding is performed using weighted finite state transducers (Tromble et al., 2008). A more efficient implementation was designed subsequently (Blackwood et al., 2010; Blackwood, 2010).

2.10.3 Lattice Minimum Bayes-Risk for System Combination

Machine translation system combination is a paradigm that exploits the differences in strength and weaknesses from different machine translation systems. Given an input source sentence to be translated, the source sentence is translated by several MT systems which produce one or more translations. These translations are then rescored and/or combined with the hope to produce a final translation of equal or better quality than the translations of any of the individual systems.

System combination techniques can be classified by whether the final translation can be a novel hypothesis that has not been generated by any of the individual systems. System combination techniques that can generate a novel hypothesis usually employ *consensus network decoding* (Fiscus, 1997). In consensus network decoding for MT, n-best lists obtained from individual systems are aligned to a reference, e.g. the MBR hypothesis, then the final hypothesis is obtained by majority voting. In practice, it is difficult to apply consensus network decoding to deep n-best lists or to lattices because of the expensive alignment operation.

Another broad class of system combination techniques that do not generate novel hypotheses include n-best list rescoring as well as lattice rescoring techniques. Blackwood (2010) argues that one possible advantage over consensus network decoding is that fluency is more likely to be preserved and that it is possible to exploit larger hypothesis spaces coming from individual systems.

It is possible to extend the lattice Minimum Bayes-Risk decoding strategy presented in Section 2.10.2 for system combination. We consider two systems that produce two translation lattices $\mathcal{E}_1$ and $\mathcal{E}_2$. The statistics $p(u|\mathcal{E})$ are computed by interpolation. The decoding equation becomes:

$$\hat{\boldsymbol{e}} = \underset{\boldsymbol{e'} \in \mathcal{E}_1 \cup \mathcal{E}_2}{\operatorname{argmax}} \left(\theta_0 |\boldsymbol{e'}| + \sum_{u \in \mathcal{N}_1 \cup \mathcal{N}_2} \theta_u \#_u(\boldsymbol{e'}) (\lambda_1 p(u|\mathcal{E}_1) + \lambda_2 p(u|\mathcal{E}_2)) \right) \tag{2.47}$$

The weights (λ_1, λ_2) are such that $\lambda_1 + \lambda_2 = 1$ and are tuned for BLEU on a development set.

Chapter 3

Data Structures for Hierarchical Phrase-Based Translation Grammars

We have seen in Chapter 2 that the main components of an SMT system are the translation model and the language model. The translation model is trained on parallel text while the language model is typically trained on the target side of the parallel text and possibly additional monolingual target text. It has been shown that increasing the amount of parallel text (Pino et al., 2010) and increasing the amount of monolingual text (Brants et al., 2007) is beneficial for translation quality.

Monolingual and parallel text availability and distributed computing techniques have allowed SMT researchers to build ever larger language models and translation models, based on very large amounts of data. However, decoders need to be able to retrieve information, such as n-gram conditional probabilities or translation grammar rules, efficiently from these models to be able to translate an input sentence or a set of input sentences in a reasonable amount of time. For online systems such as the commercial systems mentioned in Chapter 1, translation even needs to be carried out in *real time*, i.e. for example with a speed of 2000 words translated per minute.

Therefore SMT models need to be stored in a data structure that supports efficient querying and is scalable. We introduce a solution that addresses these two requirements and that has the advantage that it reuses an existing framework in order to ease the implementation. We store translation models

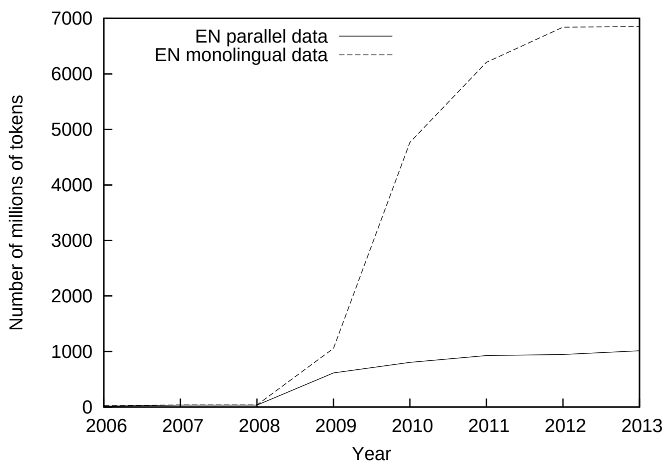

Figure 3.1: Number of English tokens (in millions) in parallel and monolingual data available for the WMT French-English constrained track translation shared task for the years 2006 to 2013.

as a set of key-value pairs in an HFile[1]. We apply this strategy in order to retrieve relevant rules from a hierarchical phrase-based grammar at the test set level. We compare our approach to alternative strategies and show that our approach offers competitive performance in terms of speed, memory and simplicity (Pino et al., 2012). We have made software written for this work available online.[2]

3.1 Introduction

Current machine translation research is characterised by increasing amounts of available data. For example, Figure 3.1 shows that for the WMT machine translation workshop (Bojar et al., 2013) French-English constrained track translation shared task, the English side of parallel data has increased from 13.8M tokens in 2006 to 1012.7M tokens in 2013, and that available English monolingual data has increased from 27.5M tokens to 6852.7M tokens over the same period. Along with growing amounts of data, the use of more power-

[1] `http://hbase.apache.org`
[2] `https://github.com/jmp84/ruleXtract` (branch PBML-2012)

ful computers and distributed computing models such as MapReduce (Dean and Ghemawat, 2008; Lin and Dyer, 2010) has enabled machine translation researchers to build larger statistical machine translation models. MapReduce has thus been applied to various steps of the translation pipeline: word alignment (Dyer et al., 2008), translation model building (Dyer et al., 2008) and language modelling (Brants et al., 2007). We review these applications of MapReduce in more detail in Section 3.2. The challenge is to find effective modelling techniques that can be implemented in the MapReduce framework.

However, once SMT models such as language models and translation grammars are built, either with MapReduce or some other method, the models must be made usable by translation decoders, which only need a fraction of the information contained in those models to be able to translate an input source sentence or a set of input source sentences. For example, in translation from French to English with a phrase-based model (see Section 2.5), given an input sentence *Salut toi*, we only need to know the translation probabilities the translation model assigns to translations of the words *Salut* and *toi* and to translations of the phrase *Salut toi*. Similarly, given a test set, a decoder only needs to retrieve the rules whose source side matches part of one of the source sentences in the test set to be able to generate hypotheses. With large models, simply retrieving relevant rules together with their translation probabilities becomes a challenge.

In the system described by Iglesias et al. (2009b), given a training parallel corpus, rules are extracted and target phrases and hierarchical phrases with counts are stored on disk. Then, given an input test set, rules are extracted from the parallel corpus a second time. Only rules relevant to the test set are retained: source sides of phrase-based rules have to match an n-gram from the test set and consecutive terminals in source sides of hierarchical rules also have to match an n-gram from the test set. As a consequence, some hierarchical rules are never used in decoding. Source sides and rules together with their counts are kept in memory using hash map datastructures, target sides with counts are loaded from disk as precomputed in the first step, so that source-to-target and target-to-source probabilities can be computed.

This method becomes progressively slower with larger amounts of data because it requires extracting rules from the training parallel corpus and recomputing translation probabilities each time we translate a new test set or a new sentence. We would like to improve on this design for more rapid experimentation. We also would like to use a computing infrastructure that requires minimal maintenance. For example, HBase[3], the open source non-relational database implementation of BigTable (Chang et al., 2008), has been applied to the use of distributed language models (Yu, 2008). Rule filtering, i.e. the retrieval of rules relevant to the translation of a test set or a sentence, is an essential step in our pipeline that can be a bottleneck. Our goal is to reduce its processing time from several hours to a few minutes.

In this chapter, we address the problem of retrieving relevant rules with their translation probabilities. We report on investigations into storing the model in the HFile data structure. To our knowledge, this is the first detailed proposed implementation of translation model storage and filtering using HFile data structures. We find that this approach offers a good compromise between speed, performance and ease of implementation. The infrastructure necessary for filtering is lightweight and requires the use of only one machine. We will apply this approach to test set rule filtering prior to decoding. We will discuss alternative strategies as well as their strengths and weaknesses in terms of speed and memory usage. In Section 3.2, we will review approaches that have been used for model building and model filtering. The HFile data structure that is used to store models will be presented in Section 3.3. In Section 3.4, we detail our custom implementation of rule extraction and translation model estimation using MapReduce. Our method and alternative strategies for rule filtering are compared empirically in Section 3.5. We conclude in Section 3.6.

3.2 Related Work

In this section, we review applications of MapReduce to the translation pipeline as well as techniques for storing and retrieving from SMT models.

3.2.1 Applications of MapReduce to SMT

Brants et al. (2007) introduce a new smoothing method called *Stupid Backoff* (see Section 2.7.4). The Stupid Backoff smoothing scheme is recalled in

[3] `http://hbase.apache.org`

Equation 3.1:

$$p_{\text{stupid backoff}}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \alpha \, p_{\text{stupid backoff}}(w_i \mid w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases} \tag{3.1}$$

With respect to the traditional backoff scheme, the Stupid Backoff scheme uses no discounting and simply uses relative frequency for the non backed-off score and the backed-off score scaling parameter is independent of the n-gram history. Therefore this scheme does not define a conditional probability distribution over a word given its history.

The Stupid Backoff language model building and application fit the MapReduce framework well. The input to language model building is a large monolingual corpus. The first step is to build a vocabulary. This is done with the canonical example of word counting with MapReduce. Counts are needed in order to remove words occurring less than a threshold from the vocabulary. The second step is to obtain n-grams and their count. This is done again with MapReduce, and the *map* and *reduce* functions are analogous to the ones defined for word counting but this time unigrams are replaced by n-grams. For n-gram counting, the partition, or sharding, function hashes on the first two words of each n-gram. In addition, unigram counts and the total size of the corpus is available in each partition, i.e. to each reducer. This allows relative frequencies to be computed. Brants et al. (2007) also demonstrate that for large amounts of monolingual data, i.e. above 10 billion tokens, Stupid Backoff smoothing and Kneser-Ney smoothing perform comparably. In addition, only Stupid Backoff smoothing can be scaled to datasets with more than 31 billion tokens. The scalability of Kneser-Ney smoothing has been improved in recent work (Heafield et al., 2013).[4]

Dyer et al. (2008) observe that translation model estimation has become prohibitive on a single core and that existing *ad hoc* parallelisation algorithms may be more fragile than using an existing framework such as the Hadoop implementation of MapReduce.[5] They provide solutions to word alignment model estimation and translation rule extraction and estimation using MapReduce and demonstrate the scalability of their method.

The convenience of the MapReduce framework for parallelisation has led to the building of end-to-end toolkits for entire phrase-based (Gao and Vogel,

[4] see also `http://kheafield.com/code/kenlm/estimation/`, Scalability section

[5] `https://hadoop.apache.org/`

2010) and hierarchical phrase-based models (Venugopal and Zollmann, 2009) for translation using the MapReduce framework.

3.2.2 SMT Models Storage and Retrieval Solutions

We now review techniques appearing in the literature that have been used to store SMT models and to retrieve the information needed in translation from these models. SMT models are usually discrete probabilistic models and can therefore be represented as a set of key-value pairs. To obtain relevant information from a model stored in a certain data structure, a set of keys called a *query set* is formed; each key in this query set is then sought in that datastructure. Strategies include:

- storing the model as a simple data structure in memory
- storing the model in a text file
- storing the model in more complicated data structures such as tries (Fredkin, 1960) (in memory or disk)
- storing fractions of the entire model
- storing data as opposed to a precomputed model
- storing models in a distributed fashion

Each of these strategies is discussed below.

In some cases, it may be possible to fit a model into RAM. In this case the model can be stored as a memory associative array, such as a hash table. In-memory storage allows allows for faster query retrieval than on-disk storage, however only smaller models will fit in memory. In-memory storage has been used to store model parameters between iterations of expectation-maximisation for word alignment (Dyer et al., 2008; Lin and Dyer, 2010).

For larger models, the set of key-value pairs can be stored as a table in a single text file on local disk. Values for keys in the query set can be retrieved by scanning through the entire file. For each key in the file, its membership is tested in the query set. This is the approach adopted in the *Joshua 5.0* decoder (Post et al., 2013)[6], which uses regular expressions or

[6] Inferred from the decoder training scripts available at `http://joshua-decoder.org/5.0/index.html`.

n-grams to test membership (see Section 3.5.4). Venugopal and Zollmann (2009) use MapReduce to scan a file concurrently: the *map* function tests if the vocabulary of a rule matches the vocabulary of a test set.

The model can also be stored using a trie associative array (Fredkin, 1960). A trie is a type of tree where each node represents a shared prefix of a set of keys represented by the child nodes. Each node only stores the prefix it represents. The keys are therefore compactly encoded in the structure of the trie itself. Querying the trie is a $\mathcal{O}(\log(n))$ operation, where n is the number of keys in the dataset. The trie may also be small enough to fit in physical memory to further reduce querying time. Zens and Ney (2007) use tries to store a phrase-based grammar. All the source phrases are represented in a trie stored on disk. Only relevant parts of the trie are loaded into memory when a source phrase is sought. Ganitkevitch et al. (2012) extend this approach to store hierarchical phrase-based grammars. Both source sides and target sides are stored in a packed representation of a trie. Packed tries have also been applied for storing language models (Pauls and Klein, 2011; Heafield, 2011).

It is also possible to create a much smaller approximate version of the model. Talbot and Osborne (2007a) represent a set of n-grams as a Bloom filter (Bloom, 1970). They first use a standard Bloom filter to define a binary feature that indicates whether an n-gram was seen in a monolingual corpus. They also use a Bloom filter to encode n-grams together with quantised counts in order to define a multinomial feature, that is a feature with a finite set of possible values—in this case the quantised values. Both these data structures can substantially reduce the disk space and memory usage with respect to lossless representations of language models. However, they allow false positives for n-gram membership queries and overestimates of n-gram quantised counts. The authors demonstrate that the features they introduce are useful in translation, despite this lack of exactness. In a related publication (Talbot and Osborne, 2007b), the authors demonstrate how these techniques can be used as a replacement to represent a smoothed language model. Talbot and Brants (2008) use a Bloomier filter (Chazelle et al., 2004) to represent n-grams and necessary statistics such as probabilities and back-off weights. Unlike Bloom filters that only support Boolean characteristic functions on a set, Bloomier filters support arbitrary functions, in this case a mapping between an n-gram and its statistics. False positives can still occur but for true positives, correct statistics are retrieved.

Guthrie and Hepple (2010) propose an extension to previous work on randomised language models (Talbot and Osborne, 2007b) which prevents the random corruption of model parameters but does not stop the random assignment of parameters to unseen n-grams. Levenberg and Osborne (2009) extend randomised language models to stream-based language models. Another way of building a smaller approximate version of a model is to retain items with high frequency counts from a stream of data (Manku and Motwani, 2002). This technique has been applied to language modelling (Goyal et al., 2009) and translation rule extraction (Przywara and Bojar, 2011).

Instead of doing some precomputation on a dataset, it is possible to compute the sufficient statistics at query time using a suffix array (Manber and Myers, 1990), so that the model can be estimated only when needed. A suffix array is a sequence of pointers to each suffix in a training corpus. The sequence is sorted with respect to the lexicographic order of the referenced suffixes. Suffix arrays have been used for computing statistics for language models (Zhang and Vogel, 2006), phrase-based systems (Callison-Burch et al., 2005; Zhang and Vogel, 2005), and hierarchical phrase-based systems (Lopez, 2007). Callison-Burch et al. (2005) store both the source side and the target side of a parallel corpus in two suffix arrays. They also maintain an index between a position in the source or target side of the corpus and the sentence number. During decoding, all occurrences of a source phrase are located in the suffix array representing the source side of the corpus. This produces a source marginal count. For each of these occurrences, the corresponding sentence pair and word alignment are retrieved and rules are extracted. This produces a rule count, which is normalised with the source marginal count to produce a source-to-target probability. Lopez (2007) extends this approach to address the grammar extraction and estimation of hierarchical rules. Note that the use of suffix arrays for translation model estimation only supports the computation of source-to-target probabilities.

Finally, some approaches store language models in a distributed fashion. We saw in Section 3.2.1 how a Stupid Backoff language model can be built. After relative frequencies have been computed, another partition function that hashes on the last two words of an n-gram is applied so that backoff operations can be done within the same partition. At decoding time, the decoder architecture is modified in order to request a batch of n-grams rather than a single n-gram. The Stupid Backoff language model building can be scaled to a corpus of 2 trillion tokens and the language model distributed application can be made real time.

Zhang et al. (2006) propose a distributed large language model backed by suffix arrays. HBase has also been used to build a distributed language infrastructure (Yu, 2008). The method we propose to use is closely related to the latter but we use a more lightweight infrastructure than HBase. In addition, it is also possible to apply our method to language model querying (Pino et al., 2012), which demonstrates the flexibility of the infrastructure.

Note that there are many alternative solutions to HBase/HFile for storage of a large number of key-value pairs, such as Berkeley DB[7] or Cassandra[8]. The purpose of this chapter is not to compare these alternatives but rather to compare existing solutions employed in the machine translation community to one of these solutions.

3.3 HFile Description

We now describe the data structure we use to store models and we review relevant features to the design of our system. To store a model represented as key-value pairs, we use the HFile file format,[9] which is an open source reimplementation of the SSTable file format (Chang et al., 2008). The HFile format is a lookup table with key and value columns. The entries are free to be an arbitrary string of bytes of any length. The table is sorted lexicographically by the key byte string for efficient record retrieval by key.

3.3.1 HFile Internal Structure

High Level Structure As can be seen in Figure 3.2, the HFile internal structure is divided into blocks. There are various types of block, depending on the kind of information a block contains. In order to distinguish block types, the first 8 bytes of a block will indicate its type. The blocks are grouped into three parts. The first part (top) is organised into data blocks interleaved with leaf data index blocks and Bloom blocks. The second part (middle) consists of intermediate data index blocks. The third part (bottom) consists of metadata: root data index blocks, a file information block and a Bloom filter index block.

[7] https://oss.oracle.com/berkeley-db.html
[8] http://cassandra.apache.org
[9] http://hbase.apache.org

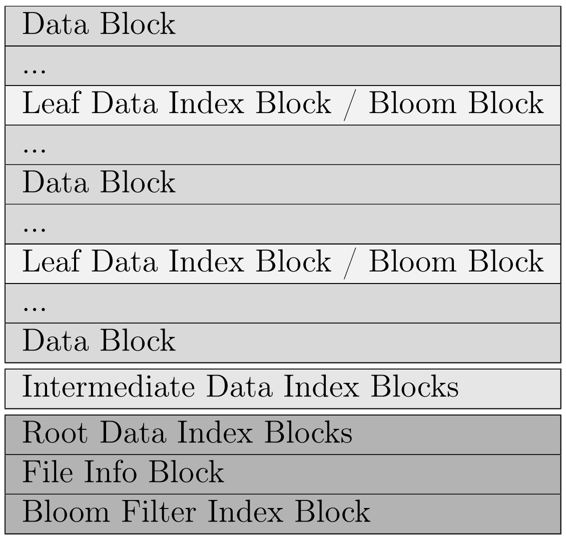

Figure 3.2: HFile internal structure.[11] The structure is divided into three parts. In the first part, data blocks are interspersed with leaf data index blocks and Bloom blocks. The second part contains intermediate data index blocks. The third part consists of metadata: root data index blocks, file format information and a Bloom filter index block.

Index Blocks The root data index blocks map keys to their relevant intermediate data index block, indicated by an offset. In turn, an intermediate data index block maps keys to their relevant leaf data index block, again indicated by an offset. Finally, a leaf data index block maps keys to their relevant data block, still indicated by an offset. The same mechanism is employed for the Bloom filter. The Bloom filter index block maps keys to their relevant Bloom block.

Block Size Configuration The block size is configurable, with a default size of 64KB. Note that HFile blocks are not to be confused with Hadoop Distributed File System (HDFS) blocks whose default size is 64MB.

Block Compression The HFile format allows for the blocks to be compressed. The choice of compression codec is selected when the file is created. We choose the GZip compression codec (Deutsch and Gailly, 1996) for all our experiments.

3.3.2 Record Retrieval

When the HFile is opened for reading, the root data index blocks are loaded into memory. To retrieve a value from the HFile given a key, the appropriate intermediate index block is located by a binary search through the root data index. Binary searches are conducted on the intermediate and leaf index blocks to identify the data block that contains the key. The identified data block is then loaded off the disk into memory and the key-value record is retrieved by scanning the data block sequentially.

3.3.3 Bloom Filter Optimisation for Query Retrieval

It is possible to query for a key that is not contained in the HFile. This very frequently happens in translation because of language data sparsity: in our case, because keys are n-grams of terminals and nonterminals, the number of possible keys is exponential in the size of the vocabulary, and many of these will not have been observed in any rule extracted from training data.

Querying the existence of a key is expensive as it involves all the operations described in Section 3.3.2: loading and binary searching the root data index, loading and binary searching an intermediate data index block, loading and binary searching a leaf data index block and finally loading and scanning a data block.

For fast existence check queries, the HFile format allows the inclusion of an optional Bloom filter (Bloom, 1970). A Bloom filter provides a probabilistic, memory efficient representation of the key set with a $\mathcal{O}(1)$ membership test operation. The Bloom filter may provide a false positive, but never a false negative, for existence of a key in the HFile. Therefore, it is safe to use in our setting as some keys will be looked up in the HFile even though they are not present, but the situation where keys that are in the HFile are not looked up will not happen.

For a large HFile, the Bloom filter may also be very large. Therefore the Bloom filter is also organised into blocks called Bloom blocks. Each block contains a smaller Bloom filter that covers a range of keys in the HFile. Similar to the root data index, a Bloom index is constructed. To check for the existence of a key, a binary search is conducted on the Bloom index, the relevant Bloom block is loaded, and the membership test performed.

[11] After `http://hbase.apache.org/book/book.html` (simplified)

Unlike work on Bloom filter language models (Talbot and Osborne, 2007a,b), this filter only tests the existence of a key and does not return any statistics from the value. If a membership test is positive, a usual key search will still be carried out. During the execution of a query, two keys may reference the same index or Bloom blocks. To prevent these blocks from being repeatedly loaded from disk, the blocks are cached after reading.

3.3.4 Local Disk Optimisation

The HFile format is designed to be used with HDFS, a distributed file system based on the Google File System (Ghemawat et al., 2003). Large files are split into HDFS blocks that are stored on many nodes in a cluster. However, the HFile format can also be used completely independently of HDFS. If its size is smaller than local disk space, the entire HFile can be stored on the local disk of one machine and accessed through the machine's local file system. We report in Section 3.5 that using local disk is faster than using HDFS.

3.3.5 Query sorting optimisation

Prior to HFile lookup, we sort keys in the query set lexicographically. If two keys in the set of queries are contained in the same block, then the block is only loaded once. In addition, the computer hardware and operating system allow further automatic improvements to the query execution. Examples of these automatic improvements include reduced disk seek time, caching data from disk by the operating system,[12] or CPU caching data from main memory (Patterson and Hennessy, 2009).

3.4 Hierarchical Rule Extraction with MapReduce

In the previous section, we have described the HFile format. We will use this format to store a translation grammar. In this section, we describe how we generate this grammar and produce the HFile that represents the grammar. We describe a MapReduce implementation of hierarchical grammar

[12] The Linux Documentation Project, The File System, `http://tldp.org`

extraction and estimation. The implementation is an extension of *Method 3* described by Dyer et al. (2008), which was originally developed to estimate translation probabilities for phrase-based models.

MapReduce (Dean and Ghemawat, 2008) is a framework designed for processing large amounts of data in a distributed fashion on a computer cluster. In this framework, the programmer needs to implement a function called *map* and a function called *reduce*. The *map* function is defined in Equation 3.2:

$$\text{map} : (k_1, v_1) \longmapsto [(k_2, v_2)] \tag{3.2}$$

where (k_1, v_1) is a key-value pair and $[(k_2, v_2)]$ is a list of key-value pairs. We use the $[\,]$ notation to denote a list. Keys and values are arbitrary byte sequences. As the key-value pairs (k_2, v_2) are computed by the *map function*, the MapReduce framework groups all values v_2 by key k_2 in order to provide an input $(k_2, [v_2])$ to the *reduce* function, defined in Equation 3.3:

$$\text{reduce} : (k_2, [v_2]) \longmapsto [(k_3, v_3)] \tag{3.3}$$

The input to the *reduce* function is a key and a list of values. The *reduce* function generates a list of key-value pairs.

Using this notation, we can describe the grammar extraction and estimation as a series of MapReduce jobs, summarised in Figure 3.3. This architecture is similar to the one described in the original publication (Dyer et al., 2008). For source-to-target and target-to-source probability computation, we used Method 3 described in that publication, which was reported to be faster than other methods. Keeping each feature computation as a separate MapReduce job makes the integration of a new feature more convenient.

3.4.1 Rule Extraction

The first step in grammar extraction and estimation is rule extraction. For this step, v_1 represents a sentence pair together with a word alignment and k_1 contains metadata about the parallel corpus such as what collection the sentence pair comes from. This kind of information is useful for domain adaptation as demonstrated in Section 5.5. The *map* function simply extracts hierarchical rules for that sentence pair and outputs the rules. k_2 is a rule and v_2 contains the metadata from k_1. For rule extraction, there is no reduce step. Equation 3.4 illustrates the *map* function for rule extraction:

$$map : (\text{newswire}, (\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{a})) \longmapsto [(r_1, \text{newswire}), (r_2, \text{newswire}), ...] \tag{3.4}$$

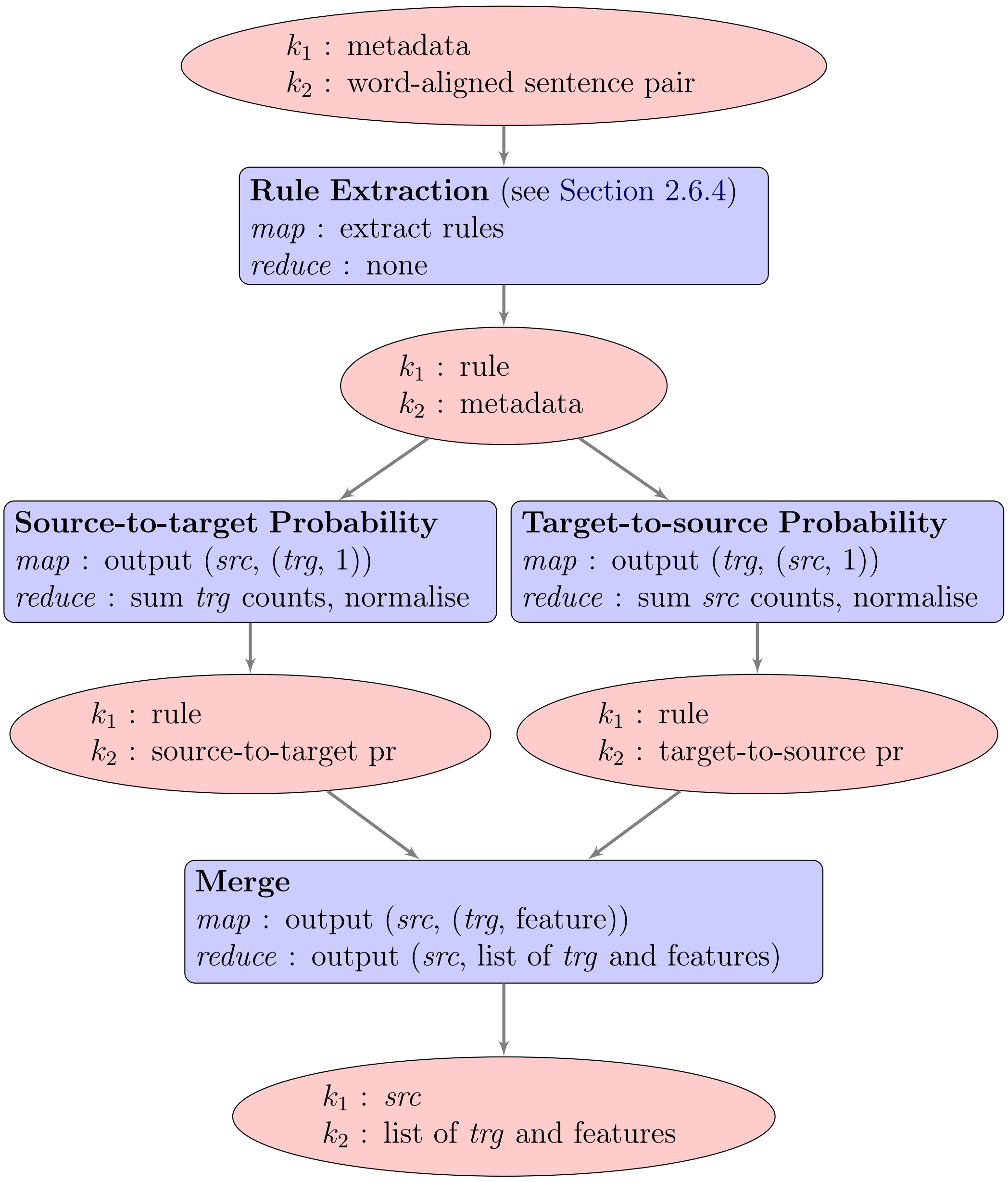
k_1 : metadata
k_2 : word-aligned sentence pair
Rule Extraction (see Section 2.6.4)
map : extract rules
reduce : none
k_1 : rule
k_2 : metadata
Source-to-target Probability
map : output (*src*, (*trg*, 1))
reduce : sum *trg* counts, normalise
Target-to-source Probability
map : output (*trg*, (*src*, 1))
reduce : sum *src* counts, normalise
k_1 : rule
k_2 : source-to-target pr
k_1 : rule
k_2 : target-to-source pr
Merge
map : output (*src*, (*trg*, feature))
reduce : output (*src*, list of *trg* and features)
k_1 : *src*
k_2 : list of *trg* and features

Figure 3.3: Translation grammar extraction and estimation pipeline as a series of MapReduce jobs. Ellipses represent (intermediate) input/output of MapReduce jobs. Rectangles represent MapReduce jobs. Source, target and probability are abbreviated into src, trg and pr. Rules are first extracted, then the source-to-target and target-to-source translation models are estimated. Finally, the features, i.e. the translation models, are merged and the final output has rules sources as keys and lists of targets with features as values.

3.4.2 Corpus-Level Feature Computation

Rule extraction is followed by several MapReduce jobs that are run simultaneously in order to compute *corpus-level* features. Corpus-level features are features related to each rule and the computation of which requires looking at the entire set of extracted rules. In this description, we only consider source-to-target and target-to-source probabilities, but many other corpus-level features are possible, for example features that relate to the word alignment from which rules were extracted. We use the metadata stored in the output of the extraction job in order to compute collection specific probabilities. Collection specific probabilities are probabilities computed over a specific subset of the training corpus. The metadata indicates whether a rule was extracted from that specific subset.

Let us consider first the collection-specific source-to-target probability computation. The input to the *map* function is the output of the extraction job, that is a rule together with metadata. The *map* function checks that the rule comes from the collection we are interested in and if so, outputs the source of the rule as a key k_2 and the target of the rule together with a count of one as a value v_2. Targets (and counts) corresponding to the same source are grouped together by the MapReduce framework. The input to the *reduce* function is a source (k_2) and a list of targets with counts ($[v_2]$). The *reduce* function aggregates the counts for the distinct targets and then normalises those counts with the total count corresponding to the source k_2. The output of the *reduce* function is a list of pairs (rule, probability).

The target-to-source probability computation is very similar. Apart from inverting the role played by source and target, the *map* and *reduce* function also need to change the rule format for rules that invert the order of nonterminals in the source and the target, for example $X \rightarrow \langle X_1 a X_2, X_2 b X_1 \rangle$. For this type of rule, the *map* function has to invert the order of nonterminals in both the source and target so that probabilities are computed correctly. For example, the *map* function will output the target of the rule written as $X_1 b X_2$ and the source of the rule written as $X_2 a X_1$. The *reduce* function will restore the original notation.

We have described the computation of collection specific translation probabilities. The source-to-target and target-to-source probabilities over the entire corpus are computed in an identical manner with the collection simply being the entire corpus. Figure 3.3 illustrates only two corpus-level features, however, in practice, more features are computed: source-to-target and target-to-source probability for each collection and for the entire parallel text, as well as lexical features (see Section 2.6.5).

3.4.3 Feature Merging

Once all MapReduce features have been computed, a MapReduce job that merges the features is called. The input key k_1 to the *map* function is a rule and the input value v_1 is a feature. The *map* function outputs the source of the rule as a key k_2 and the target of the rule together with the feature as value v_2. Targets and features corresponding to the same source are grouped together by the MapReduce framework. The input to the *reduce* function is a source (k_2) and a list of targets with features ($[v_2]$). The *reduce* function simply merges the features for identical targets. The output key k_3 is the source k_2 and the output value v_3 is a list of targets with all computed features.

The output of this merging job is converted to an HFile format. Because the keys in the HFile format need to be sorted, we also want the output of the merging job to be sorted by key. One way to achieve this is to specify only one reducer for the job but this solution is very slow because only one core will be used for the reduce step. A better way is to use a *TotalOrderPartitioner*. We first use a small amount of data so that it may be feasible to carry out rule extraction with only one reducer in the merge step. We use the output of this small scale extraction as input to an *InputSampler*. The sampler will generate a partition file for the partitioner. The use of a *TotalOrderPartitioner* will guarantee that all keys are sorted after the reduce step.

3.5 Hierarchical Rule Filtering for Translation

In the previous section, we have described how to obtain an HFile that represents a hierarchical translation grammar. We will now describe how this datastructure can be used to retrieve rules for a given test set efficiently. We describe our system called *ruleXtract*, and compare it to other methods through time and memory measurements.

3.5.1 Task Description

Given a test set, defined as a set of sentences to be translated, and a hierarchical phrase-based translation grammar, we would like to retrieve all the relevant rules from the model. A phrase-based rule in the HFile is relevant if its source is a subsequence of a sentence in the test set. A hierarchical rule in the HFile is relevant if its source, where nonterminals are instantiated into terminals, is a subsequence of a sentence in the test set. For example, with a test set containing one sentence *Salut toi*, the phrase-based rules with sources *Salut*, *toi*, *Salut toi* are relevant and the hierarchical rules with sources *Salut X* and *X toi* are relevant, because instantiating *Salut X* into *Salut toi* produces a subsequence of the test sentence and similarly for *X toi*.

3.5.2 HFile for Hierarchical Phrase-Based Grammars

Given a test set and an HFile storing a hierarchical phrase-based grammar, we first generate queries from the test set, then retrieve relevant rules along with their corpus-level features from the HFile. To generate queries, we have a set of allowed *source patterns* and instantiate these patterns against the test set. Rule patterns were defined in Section 2.6.2. A source pattern is simply the source side of a rule pattern. Here, we briefly redefine source patterns. A source pattern is a regular expression that matches the source side of a rule. For example, the pattern Σ^+X, with Σ the source vocabulary, represents a rule source side containing a sequence of terminals followed by the nonterminal X. If the input sentence is *Salut à toi*, the pattern will be instantiated as *Salut X*, *Salut à X* and *à X*. We impose the following constraints on source pattern instantiation where the first four relate to constraints in extraction and the last one relates to a decoding constraint:

- s_{max}: maximum number of terminals for phrase-based rules.
- $s_{\text{max elements}}$: maximum number of terminals and nonterminals.
- $s_{\text{max terminals}}$: maximum number of consecutive terminals for hierarchical rules.
- $s_{\text{max NT}}$: maximum nonterminal span in a hierarchical rule.
- $s_{\text{max span}}$: maximum span for the rule. This constraint is not to be confused with the previous $s_{\text{max NT}}$ constraint. While $s_{\text{max NT}}$ represents

the maximum span of a nonterminal on the right hand side of a rule, $s_{\text{max span}}$ represents the maximum span of an entire rule, or alternatively the maximum span of the nonterminal on the left hand side of the rule. Because the latter constraint is enforced by the decoder, we also enforce it in source pattern instantiation so that rules that will never be used by the decoder are not generated.

The source pattern instances are then sorted for more efficient HFile lookup (see Section 3.3.5). Each query is then looked up in the HFile and if present, an HFile record is retrieved. We now compare our approach to similar approaches whose aim is to obtain rules which are relevant to a test set.

3.5.3 Suffix Arrays for Hierarchical Phrase-Based Grammars

One alternative strategy for building a set specific grammar is to use suffix arrays introduced in Section 3.2. We use the *cdec* software (Dyer et al., 2010) implementation of suffix arrays for hierarchical phrase-based rule extraction. The *cdec* implementation is based on earlier work (Lopez, 2007) which extends suffix array based rule extraction from phrase-based system to hierarchical phrase-based systems (see again Section 3.2).

Given a test set, a set of source pattern instances is generated similarly to what is done for *ruleXtract*. These source pattern instances are then sought in a suffix array compiled from the source side of a parallel corpus. Rules are extracted using the word alignment, and source-to-target probabilities are then computed as described in Section 3.2.

3.5.4 Text File Representation of Hierarchical Phrase-Based Grammars

We now describe the *Joshua* decoder (Weese et al., 2011) implementation for storing and retrieving from a translation model. The first implementation variant, which we call *Joshua*, stores the translation model in a text file. Given a test set, each word in the test set vocabulary is mapped to the list of sentences in which it appears. Then, each rule in the translation model is compiled to a regular expression, and each sentence that contains at least a vocabulary word of the rule is matched against this regular expression. If at least one match is successful, the rule is retained.

A faster version is provided that matches consecutive terminals in the source side of a rule to the set of n-grams extracted from the test set. We call this version *Joshua Fast*. A parallel version also exists that chunks the grammar file and distributes each chunk processing as a separate process on a cluster running Sun Grid Engine (Gentzsch, 2001). We call this version *Joshua Parallel*. The parallel version using the faster matching algorithm is called *Joshua Fast Parallel*.

3.5.5 Experimental Design

In this section, we describe our experimental setup in terms of data and various configurations for grammar extraction, grammar filtering and time and memory measurements.

Parallel Text We use a small parallel corpus of 750,950 word-aligned sentence pairs and a larger corpus of 9,221,421 word-aligned sentence pairs from the NIST'12 Chinese-English evaluation, in order to investigate how systems perform with varying amounts of parallel text and whether those systems are robust when dealing with larger data sets.

Grammar Extraction From the parallel corpora, we extract hierarchical grammars with the source-to-target probability feature only. This is done because we do not want feature computation to introduce variability in timing results when comparing different strategies and software implementations. In addition, not all systems can generate the same set of features. For example, the suffix array implementation of rule extraction is not able to generate target-to-source probabilities. Note that in practice for translation decoding, given a vector of parameters, we could simply replace multiple features in the translation model by a single value representing the dot product of the features with the parameter vector. However, we need to keep separate feature values for optimisation (see Section 2.8.2).

The rule extraction constraints described in Section 3.5.2 are set to a maximum source phrase length s_{max} of 9, a maximum source element (terminal and nonterminal) length $s_{\text{max elements}}$ of 5, a maximum number of source consecutive terminals $s_{\text{max terminals}}$ of 5 and a maximum source nonterminal span $s_{\text{max NT}}$ of 10. These settings are standard in all our translation systems and give competitive performance in terms of translation quality.

With these settings, the small grammar contains approximately 60M rules while the larger grammar contains approximately 726M rules. The grammars we obtain are converted to the *Joshua* format in order to be able to filter them with the *Joshua* decoder. We do not generate a hierarchical grammar with *Joshua* simply because we want to compare the performance of rule retrieval from the same grammar.

Grammar filtering For *ruleXtract*, we use the following constraints for source pattern instantiation: the maximum source phrase length s_{max} is set to 9, the maximum source element (terminal and nonterminal) length $s_{\text{max elements}}$ is set to 5, the maximum number of source consecutive terminals $s_{\text{max terminals}}$ is set to 5, the maximum source nonterminal span $s_{\text{max NT}}$ is set to ∞ and the maximum rule span $s_{\text{max span}}$ is set to ∞. The first three constraints are standard in all our translation systems. We use the value ∞ for $s_{\text{max NT}}$ and $s_{\text{max span}}$ so that the number of rules obtained after filtering is identical between *ruleXtract* and *Joshua*. Thus, time and memory measurements are comparable.

Measurements We report time measurements for query processing and query retrieval and the total time used to obtain a set specific rule file for a test set of 1755 Chinese sentences and 51008 tokens. We also report peak memory usage. For *ruleXtract*, query processing involves generating source pattern instances and sorting them according to the HFile sorting order. If we use a Bloom filter, it also involves pre-filtering the queries with the Bloom filter. Query retrieval involves HFile lookup. For the *Joshua* configurations, query processing involves indexing the test set and generating test set *n*-grams and query retrieval involves regular expression matching.

For the *Joshua Parallel* configurations, we use 110 jobs for the larger grammar on a cluster of 9 machines. For this latter configuration, we report the maximum time spent on a job (not the sum) and the maximum memory usage by a job.

Hardware configuration the machine used for query processing has 94GB of memory and an Intel Xeon X5650 CPU. The distributed file system is hosted on the querying machine and other machines with the same specification, which are used to generate the HFile.

Our experimental setup was designed for accurate comparisons between alternative strategies for grammar filtering. In practice, an end-to-end translation system need not use this exact setup. For example the grammar extracted by *Joshua* is in general smaller than the grammar extracted by *ruleXtract* because *Joshua* includes target side constraints. On the other hand, *ruleXtract* uses threshold cutoffs, such as a minimum source-to-target translation probability, in order to reduce the size of the test set specific grammar.

3.5.6 Results and Discussion

Results are summarised in Table 3.1, from which we can draw the following observations:

Speed The *Total Time* column shows that *ruleXtract* is competitive with alternative strategies in terms of speed.

Memory The *Peak Memory* column shows that both *ruleXtract* and *Joshua* are memory hungry, with a usage of up to 40GB. In the case of *ruleXtract*, this is because we keep all source pattern instances for all test set sentences are kept in memory. In the case of *Joshua*, this is due to a caching optimisation: rule source sides that have already been considered relevant to the test set are stored in a hash map.

Local disk optimisation Comparing *HDFS* and *Local* rows (row 1 vs. row 3, row 2 vs. row 4, row 7 vs. row 9, row 8 vs. row 10), we can see that using the local filesystem as opposed to HDFS gives a small decrease in query retrieval time, which becomes more substantial for the larger grammar. This is because when the HFile is stored on HDFS, it is composed of several HDFS blocks that can be located on different data nodes. Since the HFile size is smaller than the disk space on each data node, it is preferable to store the HFile on local disk. However, because the HFile was generated on HDFS through a MapReduce job, this requires copying or moving the HFile from the HDFS file system to the local disk file system prior to running grammar retrieval.

Small Grammar						
Row	**System**	**Query Processing**	**Query Retrieval**	**Total Time**	**Peak Memory**	**# Rules**
1	*ruleXtract HDFS*	9m1s	7m36s	16m40s	40.8G	6435124
2	*ruleXtract Bloom, HDFS*	8m57s	2m16s	11m15s	39.9G	6435124
3	*ruleXtract Local*	8m54s	7m33s	16m30s	40.4G	6435124
4	*ruleXtract Bloom, Local*	8m50s	2m19s	11m11s	38.8G	6435124
5	*Joshua*	0.9s	29m51s	29m54s	42.2G	6435124
6	*Joshua Fast*	0.9s	7m25s	7m28s	40.1G	7493178
Large Grammar						
Row	**System**	**Query Processing**	**Query Retrieval**	**Total Time**	**Peak Memory**	**# Rules**
7	*ruleXtract HDFS*	8m56s	22m18s	31m17s	42.2G	47978228
8	*ruleXtract Bloom, HDFS*	9m12	15m33s	24m49s	40.7G	47978228
9	*ruleXtract Local*	8m55s	21m3s	30m1s	41.6G	47978228
10	*ruleXtract Bloom, Local*	9m0s	14m43s	23m46s	40.6G	47978228
11	*Joshua*	0.9s	out of memory	out of memory	out of memory	out of memory
12	*Joshua Fast*	0.9s	out of memory	out of memory	out of memory	out of memory
13	*Joshua No Cache*	0.9s	537m10s	537m11s	10.1G	47978228
14	*Joshua Fast No Cache*	0.9s	78m53s	78m54s	10.1G	83339443
15	*Joshua Parallel*	total time for slowest job: 43m36s			4G	47978228
16	*Joshua Fast Parallel*	total time for slowest job: 44m29s			4G	83339443

Table 3.1: Time and memory measurements for rule filtering with different strategies for a small and a large grammar. Various configurations for *ruleXtract* and *Joshua* are compared for time and memory usage. The *fast* configuration for *Joshua* filters rules by matching consecutive terminals to test set n-grams, which explains that the number of rules obtained is higher than in all other configurations. This also means that some filtered rules are never used in decoding. Measurements were carried out twice for each condition and the second run was recorded. Thus, measurements reported take advantage of operating system caching.

Bloom filter optimisation Comparing rows with and without *Bloom* (row 1 vs. row 2, row 3 vs. row 4, row 7 vs. row 8 and row 9 vs. row 10), we can see that the use of a Bloom filter gives an important decrease in query retrieval time. This is due to the fact that the number of source pattern instances queries is 31,552,746 and after Bloom filtering, the number of queries is 1,146,554 for the small grammar and 2,309,680 for the larger grammar, reducing the number of time consuming HFile lookups respectively by 96% and 93%. Note that Bloom filters increase query processing time only for the large grammar and more so when using HDFS.

Parallelisation In order to run *Joshua* on the larger grammar and avoid memory problems, we needed to use parallelisation, which provided competitive speeds and a low memory footprint (maximum 4G per job) (see rows 15 and 16).

For comparison, *cdec*'s total processing time is 57m40s for the small grammar, which is significantly slower than the other methods. However, we do not include *cdec*'s suffix array implementation performance in Table 3.1 because the suffix array method involves much on-the-fly computation, such as rule extraction and source-to-target probability estimation, that has already been precomputed in the case of *Joshua* and *ruleXtract*, making direct comparison difficult.

Despite this apparent slowness, the use of suffix array methods for rule extraction favours rapid experimentation because no time consuming precomputation is required. With our method, the precomputation involved consists in generating an HFile. This takes about 20 minutes for the small corpus and about 5 hours for the larger corpus. With the suffix array method, compiling the small parallel corpus into a suffix array data structure takes 2m49s and extracting a grammar for the test set from that corpus takes 57m40s as mentioned. Compiling the large parallel corpus into a suffix array takes 84m29s and extracting a grammar for our test set takes 569m54s. This gives an indication of the total time needed to extract a grammar for an arbitrary corpus and an arbitrary test set.

In our case, even though HFile generation might be relatively time consuming (5 hours for the larger corpus), once the HFile is generated, it is possible to quickly extract grammars for arbitrary test sets and arbitrary filtering parameters for rapid experimentation. This approach favours repeated experimentation with varying development and test sets.

Small Grammar			
System	**Extraction Time**	**Retrieval Time**	**Total Time**
cdec	2m49s	57m40s	1h29s
ruleXtract	20m	11m11s	31m11s
Large Grammar			
cdec	1h24m29s	9h29m54s	10h54m23s
ruleXtract	5h	23m46s	5h23m46s

Table 3.2: Timings for *cdec* and *ruleXtract* for both rule extraction and retrieval.

Table 3.2 compares timings between suffix array rule extraction and retrieval as implemented by *cdec* and our system. For *cdec*, the extraction time corresponds to corpus compilation. For both the small and large grammars, *ruleXtract* performs twice as fast.

The *ruleXtract* system works in batch mode and dividing the number of words in the test set by the total time in the best configuration (*ruleXtract, Bloom, Local* row 10) for the large grammar yields a speed of 35.8 words per second which is a real time system speed for batch processing tasks in which latency has little effect. However, running the system in that configuration gives a speed of 2.5 words per second for the longest sentence in the test set (135 words) and 1.3 words per second for a sentence of length 20. This may be due to several factors. First, different sentences in a test set may share the same source pattern instances, which saves computation time for batch processing. Second, slow I/O operations such as opening the HFile and reading the HFile metadata is shared by all test sentences in batch processing. Finally, processing only one sentence at a time may not benefit from caching (see Section 3.3.3 and Section 3.3.5) as much as processing an entire test set. Future work will be dedicated to reduce latency and obtain an actual real time system at the translation instance level.

3.6 Conclusion

We have presented techniques to build large translation grammars and to filter these grammars to a given test set efficiently. Our strategy is relatively easy to implement, it is flexible in that it can be applied to other tasks such as language modelling, and it does not require extensive computing resources as it is run on one machine. We have demonstrated that its performance in terms of speed and memory usage is competitive with other current alternative approaches.

There are several possible extensions to our current infrastructure. First, rule retrieval can be parallelised by source sentence in order to provide sentence specific grammars rather than test set specific grammar. Rule retrieval can also be parallelised by processing queries in parallel. The HFile format does not support concurrent reading with multithreading, but it allows multiprocessing reading, which makes this option suitable for MapReduce. Finally, in order to provide a real time system, causes of latency may be tackled; for example optimising the source pattern instance creation phase would help reduce latency.

Chapter 4

Hierarchical Phrase-Based Grammar Extraction from Alignment Posterior Probabilities

In Chapter 3, we have described how to exploit the MapReduce framework and the HFile format in order to generate very large translation grammars and retrieve rules from these grammars efficiently. The main contribution was at the infrastructure level rather at the modelling level. In this chapter, we will attempt to improve models for translation grammar extraction.

Standard practice in SMT systems is to decouple the word alignment phase from the rule extraction phase. Typically, word alignment models are only used to obtain a set of alignment links, then those alignment links determine constraints that are followed in the rule extraction step (see Section 2.6.4). In this chapter, we attempt to leverage the information contained in alignment models by extracting rules from alignment posterior probabilities (de Gispert et al., 2010b). These statistics are computed from the HMM alignment model (Vogel et al., 1996) and they are used both to generate constraints for rule extraction and for translation model estimation.

This chapter presents two rule extraction methods. With the first method, rules are extracted from alignment link posterior probabilities. With the second method, rules are extracted from alignment posterior probabilities over phrase pairs. We demonstrate improvements on a medium scale Chinese-English task with these methods. We also investigate how best to exploit source-to-target and target-to-source alignment models.

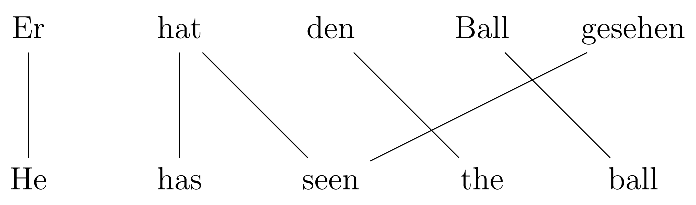

Figure 4.1: German-English word-aligned sentence pair. The spurious alignment link between the German word *hat* (*has*) and the English word *seen* prevents the phrase pair ⟨*hat*, *has*⟩ to be extracted from this sentence pair.

4.1 Introduction

In state-of-the-art SMT systems, rules are extracted from word-aligned parallel text. The alignments are typically generated by applying symmetrisation heuristics (see Section 2.3.2) to Viterbi alignments (see Equation 2.7) obtained from source-to-target and target-to-source word alignment models. Additional information that these models could provide, such as posterior probabilities over alignment links, is not used. For example, let us consider the word-aligned German-English sentence pair in Figure 4.1. Intuitively, we can tell that there is a spurious alignment link between the German word *hat* (*has*) and the English word *seen*. This link will prevent the extraction of the useful phrase pair ⟨*hat*, *has*⟩ from this sentence pair. However, it is possible that the *posterior probability* of this spurious link according to the alignment model is relatively low. We hypothesise that posterior probability information from alignment models is more reliable than the links obtained from Viterbi alignment.

In this chapter, we use HMM alignment models (see Section 2.3.1) to generate the statistics needed to both extract rules and estimate the translation models. We hypothesise that this tighter coupling between alignment and translation models will provide better translation quality.

We will evaluate the grammar we obtain in two ways. First, we will assess the grammar's ability to generate a reference translation from a source sentence. This is determined by the type of reordering allowed by the grammar and by the choice of translations for each source side of a rule. We will then evaluate translation quality provided by this grammar.

Conceptually, our extraction method consists in extracting all possible phrase pairs and hierarchical phrase pairs given a sentence pair and selecting only those that satisfy certain statistical criteria related to alignment posterior probabilities. For example, we can select phrase pairs that contain a link with a high posterior probability; or we can select phrase pairs that contain a link with a high posterior probability and that have a high phrase pair posterior probability. The selection process determines which rules the grammar will contain and will therefore define the ability of the grammar to generate a reference translation given a source sentence. We can also use statistics from alignment models to estimate translation models in a novel way. In this work, we will use phrase pair posterior probability instead of integer counts to estimate translation models.

4.2 Related Work

The limitations of extracting translation rules from Viterbi alignments, i.e. that potentially useful information from the alignment models is ignored, has been addressed previously. Venugopal et al. (2008) extract rules from n-best lists of alignments and n-best lists of syntactic parses for a syntax-augmented hierarchical system (Zollmann and Venugopal, 2006). In the alignment step, an n-best list of alignments $\boldsymbol{a_1}, ..., \boldsymbol{a_n}$ is produced with posterior probabilities $p(\boldsymbol{a_1} \mid \boldsymbol{f}, \boldsymbol{e}), ..., p(\boldsymbol{a_n} \mid \boldsymbol{f}, \boldsymbol{e})$. These posteriors are normalised to produce probabilities $\hat{p}(\boldsymbol{a_1}), ..., \hat{p}(\boldsymbol{a_n})$. Similarly, probabilities $\hat{p}(\boldsymbol{\pi_1}), ..., \hat{p}(\boldsymbol{\pi_{n'}})$ are obtained for an n'-best list of parses $\boldsymbol{\pi_1}, ..., \boldsymbol{\pi_{n'}}$. For each alignment $\boldsymbol{a_i}$ and parse $\boldsymbol{\pi_j}$, syntax-augmented hierarchical rules are extracted with a count $\hat{p}(\boldsymbol{a_i})\,\hat{p}(\boldsymbol{\pi_j})$.

Alignment n-best lists have also been used to create a structure called *weighted alignment matrix* (Liu et al., 2009). Probabilities $\hat{p}(\boldsymbol{a_1}), ..., \hat{p}(\boldsymbol{a_n})$ for n-best alignments $\boldsymbol{a_1}, ..., \boldsymbol{a_n}$ are computed as previously (Venugopal et al., 2008). Then, for each word pair (f_j, e_i), the alignment link posterior probability $p_m(j, i)$ is computed in Equation 4.1.

$$p_m(j, i) = \sum_{k=1}^{n} \hat{p}(\boldsymbol{a_k}) \delta(\boldsymbol{a_k}, i, j) \tag{4.1}$$

$\delta(\boldsymbol{a_k}, i, j)$ indicates whether there is a link between i and j in the alignment $\boldsymbol{a_k}$. Given a sentence pair, all phrase pairs with a maximum source length and a maximum target length that contain a link with a posterior greater than zero are extracted. The fractional counts assigned to these phrase pairs are computed in terms of the link posteriors and then used to estimate the translation models by relative frequency. The fractional count computation approximates the posterior probability of all alignments consistent with the phrase pair. Our method also uses link posterior probabilities to constrain the extraction but the posteriors are computed exactly rather than approximated. In addition, posterior probabilities of consistent alignments is also computed exactly. Finally, our method is also applied to hierarchical phrase-based translation.

Alignment posterior probabilities without approximation have also been used. For a given test set, Deng and Byrne (2008) first extract phrase pairs in a standard manner. Then, source phrases in the test set that do not have any corresponding target in the list of extracted phrase pairs are selected. For each of these source phrases, sentence pairs where the source phrase occurs are considered. For each such sentence pair, all target phrases in the target sentence are assigned phrase pair posterior probabilities (see Section 4.3.4) according to the source-to-target and target-to-source alignment models, then ranked by the geometric average of the two probabilities. The top phrase pair is retained if its scores are above specific thresholds. Our definition of phrase pair posterior probabilities and the procedure to compute them are directly inspired by the work we just described. However, we do not use the word-to-phrase HMM model but the simpler word-to-word HMM model. In addition, our method is applied to hierarchical phrase-based grammars rather than simpler phrase-based grammars. Finally, our grammar extraction scheme does not consist in first extracting a standard grammar and then augmenting the grammar with additional rules: we modify the extraction procedure to directly extract a hierarchical grammar from alignment link posterior probabilities or phrase pair posterior probabilities.

Kumar et al. (2007) also use exact computation of alignment link posteriors in a different application setting. First, instead of using the Viterbi criterion for word alignment reminded in Equation 4.2,

$$\hat{\boldsymbol{a}} = \operatorname*{argmax}_{\boldsymbol{a}} p(\boldsymbol{f}, \boldsymbol{a} \mid \boldsymbol{e}) \tag{4.2}$$

the maximum a posteriori criterion (Matusov et al., 2004), shown in Equation 4.3, is used:

$$\hat{a}_j = \operatorname*{argmax}_{i} p(a_j = i \mid \boldsymbol{f}, \boldsymbol{e}) \tag{4.3}$$

Then, given a parallel corpus for three languages F, G, E, the link posteriors for the language pair (F, E) are computed in terms of the posteriors for the language pair (F, G) and (G, E). G is called a *bridge* language. The motivation is that alignments for the F-G language pair and the G-E language pair may inform alignment for F-E. Multiple bridge languages are used and produce corresponding posterior matrices. The matrices are interpolated and alignments are extracted for each bridge language and for the interpolation. Translation gains are obtained in system combination.

We also note approaches to tighter coupling between hierarchical phrase-based grammars and alignments or even direct modelling of phrase alignment. Marcu and Wong (2002) introduce a joint phrase-based model that does not make use of word alignments. In this generative model, a sentence pair is produced by concatenating phrase pairs, or so-called *concepts*. The authors consider a simpler model with only joint phrase pair translation probabilities and a more complex model with translation and distortion probabilities. The parameter are trained with an approximate version of the expectation-maximisation algorithm (Dempster et al., 1977). Experiments demonstrate translation improvements over IBM Model 4. Birch et al. (2006) constrain this model in order to be able to apply it to larger parallel corpora. When searching for a set of phrase pairs to cover a training sentence pair, phrase pairs that are consistent with the intersection of Viterbi alignments (see Section 2.3.2) are considered first; other phrase pairs are considered only when the sentence pair cannot be covered entirely. Results close to standard phrase-based models are obtained.

DeNero and Klein (2008) prove that phrase alignment is an NP-hard problem. Given a sentence pair $(\boldsymbol{f}, \boldsymbol{e})$, a bijective phrase alignment $\boldsymbol{a}$ is defined as a bijective mapping between source phrases that form a partition of $\boldsymbol{f}$ and target phrases that form a partition of $\boldsymbol{e}$. A scoring function ϕ is also defined that assigns a real-valued score to any phrase pair $\langle$source phrase, target phrase$\rangle$. The score of a bijective phrase alignment is simply the product of the scores of its phrase pairs. Given $(\boldsymbol{f}, \boldsymbol{e}, \phi)$, the phrase alignment optimisation problem is to find the best scoring alignment. DeNero and Klein (2008) show that this problem is NP-hard by showing that the corresponding decision problem is NP-complete via reduction of the SAT problem. We give here an indication of the size of the search space. The number of possible source partitions is $2^{|\boldsymbol{f}|-1}$. Given a source partition with $K+1$ phrases, there are $(K+1)!$ possible permutation of the source phrases and $2^{\binom{|e|-1}{K}}$ possible target partitions with $K + 1$ phrases. In conclusion, there is little hope to solve the phrase alignment problem exactly.

Saers and Wu (2009) report an improvement on a phrase-based system where word alignment has been trained with an inversion transduction grammar rather than IBM or HMM models. Phrase alignment is directly modelled with an inversion transduction grammar. The phrase alignment search space is more restrictive than the space considered in DeNero and Klein (2008) and the expectation maximisation algorithm can be carried out in $O(n^6)$ where n is the number of tokens in a sentence. Pauls et al. (2010) also use an inversion transduction grammar to directly align phrases to nodes in a string-to-tree model. Bayesian methods have also been developed to induce a grammar directly from an unaligned parallel corpus (Blunsom et al., 2008b, 2009). Finally, Cmejrek et al. (2009) extract rules directly from bilingual chart parses of the parallel corpus without using word alignments. We take a different approach in that we aim to start with very strong alignment models and use them to guide grammar extraction.

Finally, some work on smoothing, which could be complementary to the approach taken in this thesis, has been conducted to address the shortcomings of relative frequency estimation for translation models. Foster et al. (2006) conduct an extensive series of experiments that either replace the relative frequency estimated phrase table by a smoothed phrase table or add the smoothed phrase table as a feature and observe improvement in translation quality.

4.3 Rule Extraction

In Section 4.2, we have reviewed approaches that “widen” the translation pipeline by using alignment n-best lists. We have also reviewed applications of exact computation of alignment posterior probabilities and attempts to directly model phrase alignment. We will now describe our grammar extraction methods, based on exact computation of alignment posterior probabilities under an alignment model. As in previous work (Hopkins et al., 2011), we first present a general approach that encompasses both standard methods based on rule extraction from Viterbi alignments as well as our methods. For clarity of presentation, we first describe our methods in the simpler case of phrase-based rule extraction, then extend them to hierarchical phrase-based rule extraction.

```
1: function ExtractRules($f_1^J, e_1^I, \boldsymbol{a}$)
2:     for $1 \leq j_1 \leq j_2 \leq J$ do
3:         for $1 \leq i_1 \leq i_2 \leq I$ do
4:             if SourceConstraints($f_{j_1}^{j_2}$)
                 $\wedge$ AlignConstraints($f_{j_1}^{j_2}, e_{i_1}^{i_2}, \boldsymbol{a}$)
                 $\wedge$ TargetConstraints($e_{i_1}^{i_2}$) then
5:                 Extract($X \rightarrow \langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle$, Count($f_{j_1}^{j_2}, e_{i_1}^{i_2}$))
6:             end if
7:         end for
8:     end for
9: end function
```

Figure 4.2: General procedure for phrase-based rule extraction: both traditional rule extraction from Viterbi alignment and our method are instances of this procedure.

4.3.1 General Framework for Rule Extraction

We first describe a general method for the extraction of phrase-based rules. An extension of this procedure for hierarchical rules is described in Section 4.3.5. The algorithm is described in Figure 4.2. Given a sentence pair (f_1^J, e_1^I), for each source index pair (j_1, j_2) defining a source phrase $f_{j_1}^{j_2}$ (line 2), for each target index pair (i_1, i_2) defining a target phrase $e_{i_1}^{i_2}$ (line 3), if source constraints, target constraints and alignment constraints are satisfied (line 4), then the phrase pair $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$ is extracted with a certain count (line 5): the phrase pair is added to the list of phrase pairs used in translation, and the count will be used subsequently to compute translation models by relative frequency. The purpose of the constraints is to obtain a manageable number of rules. If we did not impose constraints, we would extract $\frac{I(I+1)J(J+1)}{4}$ (not necessarily distinct) rules for the sentence pair (f_1^J, e_1^I). During translation, the decoder would need to apply more pruning, which would potentially lead to more search errors and a decrease in translation quality.

We will now refine this general procedure to make it more practical and closer to a possible implementation. Let us call the source constraints $\mathcal{C}_S$, the alignment constraints $\mathcal{C}_A$ and the target constraints $\mathcal{C}_T$. These are Boolean functions used to select phrase pairs. In practice, source constraints are checked on the source phrases before looking at the target phrases. If source constraints are not met, then we need not consider target phrases for that source phrase. In addition, target phrases are only considered if they satisfy alignment constraints with the source phrase and, if they do, we rank them according to a certain ranking function $\mathcal{R}$. Target constraints also depend on the ranking $\mathcal{R}$, for example we can decide to keep only a certain number of target phrases per source phrase. When a phrase pair is extracted, it is assigned a count which will be used to estimate the source-to-target and target-to-source translation models. The counting function is called $\mathcal{C}$. With this notation, we obtain the revised extraction procedure in Figure 4.3. We will now describe different rule extraction strategies in terms of the constraints $\mathcal{C}_S$, $\mathcal{C}_A$, $\mathcal{C}_\mathcal{T}$, the ranking function $\mathcal{R}$ and the counting function $\mathcal{C}$.

4.3.2 Extraction from Viterbi Alignment Links

In this section, we describe the standard extraction procedure within the framework introduced in Section 4.3.1. Common practice takes a fixed set of word alignment links $\boldsymbol{L}$ and extracts rules from this set. Alignment links $\boldsymbol{L}$ are obtained from the alignment model $\boldsymbol{a}$ either by the Viterbi algorithm or by maximum a posteriori estimation (Matusov et al., 2004; Kumar et al., 2007) and possibly using symmetrisation heuristics to combine links obtained from source-to-target and target-to-source alignment models (see Section 2.3.2). We can restate this common approach in the framework proposed in Section 4.3.1 and in Figure 4.3 where constraints, ranking and counting functions are defined as follows:

- source constraints $\mathcal{C}_S(f_{j_1}^{j_2})$:

$$j_2 - j_1 < s_{\max} \tag{4.4}$$

  where $s_{\max}$ is a integer threshold defined experimentally. $s_{\max}$ represents the maximum length of a source phrase.

- alignment constraints $\mathcal{C}_A(f_{j_1}^{j_2}, e_{i_1}^{i_2}, \boldsymbol{a})$:

$$\Big(\forall (j,i) \in \boldsymbol{L}, j \in [j_1, j_2] \Leftrightarrow i \in [i_1, i_2]\Big) \wedge \Big(\boldsymbol{L} \cap [j_1, j_2] \times [i_1, i_2] \neq \emptyset\Big) \tag{4.5}$$

  Alignment constraints have already been described in Section 2.5.3 as the conditions required for phrase pair extraction. The first bracketed constraint requires that there be no alignment link between a word

```
1: **function** EXTRACTRULES($f_1^J, e_1^I, \boldsymbol{a}$)
2:     **for** $1 \leq j_1 \leq j_2 \leq J$ **do**
3:         **if** $\neg\mathcal{C}_S(f_{j_1}^{j_2})$ **then**                    ▷ Source constraints
4:             **continue**
5:         **end if**
6:         $T \leftarrow \emptyset$                                        ▷ Sorted target phrases
7:         **for** $1 \leq i_1 \leq i_2 \leq I$ **do**
8:             **if** $\mathcal{C}_A(f_{j_1}^{j_2}, e_{i_1}^{i_2}, \boldsymbol{a})$ **then**               ▷ Alignment constraints
9:                 $T \leftarrow T \cup e_{i_1}^{i_2}$
10:            **end if**
11:        **end for**
12:        SORT($T, \mathcal{R}$)                  ▷ Target phrases ranked according to $\mathcal{R}$
13:        **for** $e_{i_1}^{i_2} \in T$ **do**
14:            **if** $\mathcal{C}_T(e_{i_1}^{i_2}, T)$ **then**                        ▷ Target constraints
15:                EXTRACT($X \rightarrow \langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle$, $\mathcal{C}(f_{j_1}^{j_2}, e_{i_1}^{i_2})$)
16:            **end if**
17:        **end for**
18:    **end for**
19: **end function**
```

Figure 4.3: General procedure for phrase-based rule extraction. This version is more practical and closer to a possible implementation than the algorithm in Figure 4.2. Source phrases are first considered. Only if source constraints $\mathcal{C}_S$ are satisfied, then target phrases are considered. Targets that satisfy alignment constraints $\mathcal{C}_A$ with their source are ranked by $\mathcal{R}$. Finally, phrase pairs where the target satisfies target constraints can be extracted with a certain count $\mathcal{C}$. Note that the target constraints implicitly depend on the ranking of the targets by R.

inside the phrase pair and a word outside of it. The second bracketed constraint requires that there be at least one alignment link in the phrase pair. Sometimes, an additional constraint specifies that the boundary words in the phrase pair should be aligned. In this work, this constraint is not present. A phrase pair that satisfies Equation 4.5 is said to be *consistent* with the alignment (see Section 2.5.3).

- target constraints $\mathcal{C}_T(e_{i_1}^{i_2}, T)$: no constraint is imposed in this work. Target constraints based on length may be imposed depending on the implementation.

- ranking and counting functions:

$$\mathcal{R}(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = \mathcal{C}(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = 1 \tag{4.6}$$

The above constraints, ranking and counting functions define the standard approach to grammar extraction. In the next sections, we depart from this approach and apply novel functions to rank and count target-side translations according to their quality in the context of each parallel sentence, as defined by the word alignment models. We also depart from common practice in that we do not use a set of links as alignment constraints. We thus have better control over the number of extracted rules as well as the relative frequency estimates of the source-to-target and target-to-source translation models.

4.3.3 Extraction from Posteriors Probabilities over Alignment Links

For presentation, we only consider source-to-target alignment models: the random variable $\boldsymbol{a}$ that models the alignment process takes values in functions from source word positions to target word positions. However, it it possible to apply our method with any directional alignment model. We will use the link posterior probability $p(a_j = i \mid f_1^J, e_1^I)$ to guide rule extraction. This statistic expresses how likely it is that a word f_j in source position j and a word e_i in target position i are aligned given the sentence pair (f_1^J, e_1^I). The link posterior probability can be computed efficiently for Model 1, Model 2 and HMM. In our experiments, we only use the HMM model to compute link posteriors but comparisons between link posteriors obtained from various models may be interesting in the future. We will derive a closed form solution for these models to compute the link posterior probability. Applying the definition of conditional probability, we obtain the general form of the link posterior probability in Equation 4.7.

$$p(a_{j_0} = i_0 \mid f_1^J, e_1^I) = \frac{p(a_{j_0} = i_0, f_1^J \mid e_1^I)}{p(f_1^J \mid e_1^I)} \tag{4.7}$$

Using Equation 4.7, we will now derive the link posterior probability $p_{M_1}(a_{j_0} = i_0 \mid f_1^J, e_1^I)$ for Model 1, $p_{M_2}(a_{j_0} = i_0 \mid f_1^J, e_1^I)$ for Model 2 and $p_{\mathrm{HMM}}(a_{j_0} = i_0 \mid f_1^J, e_1^I)$ for the HMM model.

4.3.3.1 Link Posterior Probability for Model 1

Let us derive $p_{M_1}(a_{j_0} = i_0 \mid f_1^J, e_1^I)$, the link posterior probability under Model 1. We use the notation from (Brown et al., 1993), where t is the word-to-word translation table and ε is a constant. We compute the numerator from Equation 4.7 by marginalising over all possible alignments and inverting sum and product signs to obtain Equation 4.8:

$$\begin{aligned}
&p_{M_1}(a_{j_0} = i_0, f_1^J \mid e_1^I)\\
&= \sum_{a_1=0}^{I} \dots \sum_{a_{j_0-1}=0}^{I} \sum_{a_{j_0+1}=0}^{I} \dots \sum_{a_J=0}^{I} p_{M_1}(a_1 \dots a_{j_0-1} i_0 a_{j_0+1} \dots a_J, f_1^J \mid e_1^I)\\
&= \sum_{a_1=0}^{I} \dots \sum_{a_{j_0-1}=0}^{I} \sum_{a_{j_0+1}=0}^{I} \dots \sum_{a_J=0}^{I} \frac{\varepsilon}{(1+I)^J} t(f_{j_0} \mid e_{i_0}) \prod_{\substack{j=1\\ j\neq j_0}}^{J} t(f_j \mid e_{a_j})\\
&= \frac{\varepsilon}{(1+I)^J} t(f_{j_0} \mid e_{i_0}) \prod_{\substack{j=1\\ j\neq j_0}}^{J} \sum_{i=0}^{I} t(f_j \mid e_i)
\end{aligned} \tag{4.8}$$

We compute the denominator from Equation 4.7 similarly (see Equation (15) in (Brown et al., 1993)) and obtain Equation 4.9:

$$p_{M_1}(f_1^J \mid e_1^I) = \frac{\varepsilon}{(1+I)^J} \prod_{j=1}^{J} \sum_{i=0}^{I} t(f_j \mid e_i) \tag{4.9}$$

After simplification, we obtain Equation 4.10 from Equation 4.8 and Equation 4.9:

$$p_{M_1}(a_{j_0} = i_0 \mid f_1^J, e_1^I) = \frac{t(f_{j_0} | e_{i_0})}{\sum_{i=0}^{I} t(f_{j_0} | e_i)} \tag{4.10}$$

4.3.3.2 Link Posterior Probability for Model 2

We apply the same method to compute $p_{M_2}(a_{j_0} = i_0 \mid f_1^J, e_1^I)$, the link posterior probability for Model 2. We also use notation from (Brown et al., 1993) but we replace the notation for the alignment probability $a(i \mid j, J, I)$ by $p_a(i \mid j, J, I)$ for clarity. We compute the numerator from Equation 4.7 in Equation 4.11:

$$\begin{aligned}
&p_{M_2}(a_{j_0} = i_0, f_1^J \mid e_1^I) \\
&= \sum_{a_1=0}^{I} \dots \sum_{a_{j_0-1}=0}^{I} \sum_{a_{j_0+1}=0}^{I} \dots \sum_{a_J=0}^{I} p_{M_2}(a_1 \dots a_{j_0-1} i_0 a_{j_0+1} \dots a_J, f_1^J \mid e_1^I) \\
&= \sum_{a_1=0}^{I} \dots \sum_{a_{j_0-1}=0}^{I} \sum_{a_{j_0+1}=0}^{I} \dots \sum_{a_J=0}^{I} \varepsilon \; p_a(i_0 \mid j_0, J, I) \; t(f_{j_0} \mid e_{i_0}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \prod_{\substack{j=1 \\ j \neq j_0}}^{J} p_a(a_j \mid j, J, I) \; t(f_j \mid e_{a_j}) \\
&= \varepsilon \; p_a(i_0 \mid j_0, J, I) \; t(f_{j_0} \mid e_{i_0}) \prod_{\substack{j=1 \\ j \neq j_0}}^{J} \sum_{i=0}^{I} p_a(i \mid j, J, I) \; t(f_j \mid e_i)
\end{aligned} \tag{4.11}$$

We compute the denominator from Equation 4.7 similarly and obtain Equation 4.12:

$$p_{M_2}(f_1^J \mid e_1^I) = \varepsilon \prod_{j=1}^{J} \sum_{i=0}^{I} p_a(i \mid j, J, I) \; t(f_j \mid e_i) \tag{4.12}$$

After simplification, we obtain Equation 4.13 from Equation 4.11 and Equation 4.12.

$$p_{M_2}(a_{j_0} = i_0 \mid f_1^J, e_1^I) = \frac{p_a(i_0 \mid j_0, J, I) \; t(f_{j_0} \mid e_{i_0})}{\sum_{i=0}^{I} p_a(i \mid j_0, J, I) \; t(f_{j_0} \mid e_i)} \tag{4.13}$$

4.3.3.3 Link Posterior Probability for the HMM Model

We now derive $p_{\mathrm{HMM}}(a_{j_0} = i_0 | f_1^J, e_1^I)$, the link posterior probability for the HMM model (Vogel et al., 1996; Rabiner, 1989). These derivations are standard once we realise that the observed sequence is the source sentence f_1^J, the hidden sequence is a_1^J and that in addition to standard presentations of HMM, all probabilities are conditioned on the target sentence e_1^I. We compute the numerator from Equation 4.7 in Equation 4.14:

$$\begin{aligned}
p_{\text{HMM}}(a_{j_0} = i_0, f_1^J \mid e_1^I) &= p_{\text{HMM}}(a_{j_0} = i_0, f_1^{j_0}, f_{j_0+1}^J \mid e_1^I) \\
&= p_{\text{HMM}}(f_{j_0+1}^J \mid a_{j_0} = i_0, f_1^{j_0}, e_1^I)\ p_{\text{HMM}}(a_{j_0} = i_0, f_1^{j_0} \mid e_1^I) \\
&= p_{\text{HMM}}(f_{j_0+1}^J \mid a_{j_0} = i_0, e_1^I)\ p_{\text{HMM}}(a_{j_0} = i_0, f_1^{j_0} \mid e_1^I) \\
&= \beta_{j_0}(i_0)\ \alpha_{j_0}(i_0) \qquad (4.14)
\end{aligned}$$

where $\beta_{j_0}(i_0)$ and $\alpha_{j_0}(i_0)$ are respectively the backward and forward HMM probabilities defined in Equation 4.15:

$$\begin{aligned}
\beta_j(i) &= p_{\text{HMM}}(f_{j+1}^J \mid a_j = i, e_1^I) \\
\alpha_j(i) &= p_{\text{HMM}}(a_j = i, f_1^j \mid e_1^I)
\end{aligned} \qquad (4.15)$$

The forward and backward probabilities can be computed recursively as shown in Equation 4.16 and Equation 4.17:

$$\begin{aligned}
\alpha_j(i) &= p_{\text{HMM}}(a_j = i, f_1^j \mid e_1^I) \\
&= \sum_{k=0}^{I} p_{\text{HMM}}(a_j = i, a_{j-1} = k, f_1^{j-1}, f_j \mid e_1^I) \\
&= \sum_{k=0}^{I} p_{\text{HMM}}(f_j \mid a_j = i, a_{j-1} = k, f_1^{j-1}, e_1^I)\ p_{\text{HMM}}(a_j = i, a_{j-1} = k, f_1^{j-1} \mid e_1^I) \\
&= \sum_{k=0}^{I} p_{\text{HMM}}(f_j \mid e_i)\ p_{\text{HMM}}(a_j = i \mid a_{j-1} = k, f_1^{j-1}, e_1^I)\ p_{\text{HMM}}(a_{j-1} = k, f_1^{j-1} \mid e_1^I) \\
&= \sum_{k=0}^{I} p_{\text{HMM}}(f_j \mid e_i)\ p_{\text{HMM}}(a_j = i \mid a_{j-1} = k, I)\ \alpha_{j-1}(k) \qquad (4.16)
\end{aligned}$$

$$\begin{aligned}
\beta_j(i) &= p_{\text{HMM}}(f_{j+1}^J \mid a_j = i, e_1^I) \\
&= \sum_{k=0}^{I} p_{\text{HMM}}(f_{j+2}^J, a_{j+1} = k, f_{j+1} \mid a_j = i, e_1^I) \\
&= \sum_{k=0}^{I} p_{\text{HMM}}(f_{j+2}^J \mid a_{j+1} = k, f_{j+1}, a_j = i, e_1^I)\ p_{\text{HMM}}(a_{j+1} = k, f_{j+1} \mid a_j = i, e_1^I) \\
&= \sum_{k=0}^{I} p_{\text{HMM}}(f_{j+2}^J \mid a_{j+1} = k, e_1^I)\ p_{\text{HMM}}(f_{j+1} \mid a_{j+1} = k, a_j = i, e_1^I) \\
&\quad p_{\text{HMM}}(a_{j+1} = k \mid a_j = i, e_1^I) \\
&= \sum_{k=0}^{I} \beta_{j+1}(k)\ p_{\text{HMM}}(f_{j+1} \mid e_k)\ p_{\text{HMM}}(a_{j+1} = k \mid a_j = i, I) \qquad (4.17)
\end{aligned}$$

The denominator from Equation 4.7 is computed in Equation 4.18:

$$\begin{aligned}
p_{\text{HMM}}(f_1^J \mid e_1^I) &= \sum_{k=0}^{I} p_{\text{HMM}}(a_J = k, f_1^J \mid e_1^I) \\
&= \sum_{k=0}^{I} \alpha_J(k) \qquad (4.18)
\end{aligned}$$

We will use the link posterior probabilities under the HMM model in order to define constraints, ranking and counting functions.

4.3.3.4 Constraints, Ranking and Counting Functions from HMM Link Posterior Probabilities

We use HMM link posterior probabilities computed in Section 4.3.3.3 in order to define constraints, ranking and counting functions:

- source constraints $\mathcal{C}_S(f_{j_1}^{j_2})$:

  $$j_2 - j_1 < s_{\max} \tag{4.19}$$

  This is the same constraint as defined for standard Viterbi extraction in Section 4.3.2.

- alignment constraints $\mathcal{C}_A(f_{j_1}^{j_2}, e_{i_1}^{i_2}, \boldsymbol{a})$:

  $$\exists (j,i) \in [j_1, j_2] \times [i_1, i_2], p(a_j = i \mid f_1^J, e_1^I) > \lambda \tag{4.20}$$

  $$\forall (j,i) \in [1,J] \times [1,I] \cap \{(j,i) : p(a_j = i \mid f_1^J, e_1^I) > \lambda\} \tag{4.21}$$
  $$j \in [j_1, j_2] \Leftrightarrow i \in [i_1, i_2]$$

  where λ is a threshold defined experimentally. Intuitively, λ is a high link posterior probability. The first constraint (Equation 4.20) means that we require at least one link with a high posterior probability in the phrase pair considered. The second constraint (Equation 4.21) means that there should be no link with a high posterior probability that be inconsistent with the phrase pair. Note that these constraints are identical to the Viterbi alignment constraints defined in Section 4.3.2 if we choose $\boldsymbol{L}$ to be the set of all links with high posterior defined in Equation 4.22:

  $$\boldsymbol{L} = \{(j,i) \in [1,J] \times [1,I] : p(a_j = i \mid f_1^J, e_1^I) > \lambda\} \tag{4.22}$$

  Also note that the second constraint does not consider links to the null word (see Section 2.3) relevant. This is because we do not need to include the null word in a translation rule.

- target constraints $\mathcal{C}_T(e_{i_1}^{i_2}, T)$: we pick the first k translation candidates according to the ranking function $\mathcal{R}$.

- ranking function:

  $$\mathcal{R}(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = \prod_{j=j_1}^{j_2} \sum_{i=i_1}^{i_2} \frac{p(a_j = i \mid f_1^J, e_1^I)}{i_2 - i_1 + 1} \tag{4.23}$$

This ranking function is very similar to the score used for lexical features described in Section 2.6.5. Here, we use link posteriors instead of Model 1 translation probabilities. This function favours short target phrases, therefore we do not use it as a counting function. Preliminary experiments found that this function is not appropriate for counting rules and that it gives poor results. We therefore use the same counting function as in standard practice described in Section 4.3.2.

- counting function:
$$\mathcal{C}(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = 1 \tag{4.24}$$

We have described rule extraction from alignment link posterior probabilities. Next, we will describe rule extraction from alignment posterior probabilities over phrase pairs. This method will use the same source constraints, alignment constraints and target constraints but different ranking and counting functions.

4.3.4 Extraction from Posteriors over Phrase Pairs

In the previous section, we defined and gave closed form solutions to alignment link posterior probabilities for Model 1, Model 2 and the HMM model. We can also define alignment posterior probabilities over phrase pairs. Let us consider the phrase pair $\langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle$ in the sentence pair (f_1^J, e_1^I). In Equation 4.25, we define $A(j_1, j_2; i_1, i_2)$, the set of alignments that have no links between inside the phrase pair and outside the phrase pair:

$$A(j_1, j_2; i_1, i_2) = \{a_1^J : a_j \in [i_1, i_2] \Leftrightarrow j \in [j_1, j_2]\} \tag{4.25}$$

Alignments in $A(j_1, j_2; i_1, i_2)$ satisfy the consistency constraint defined in Equation 4.5 but do not require a link in $[j_1, j_2] \times [i_1, i_2]$. The posterior probability of these alignments given the sentence pair is defined in Equation 4.26:

$$\begin{aligned} p(A(j_1, j_2; i_1, i_2) \mid e_1^I, f_1^J) &= \frac{p(f_1^J, A(j_1, j_2; i_1, i_2) \mid e_1^I)}{p(f_1^J \mid e_1^I)} \\ &= \frac{\sum_{a_1^J \in A(j_1, j_2; i_1, i_2)} p(f_1^J, a_1^J \mid e_1^I)}{\sum_{a_1^J} p(f_1^J, a_1^J \mid e_1^I)} \end{aligned} \tag{4.26}$$

We call this quantity the *phrase pair posterior probability*. We will now derive formula for the phrase pair posterior probability in the case of Model 1, Model

2 and the HMM Model. Again, experiments only use phrase pair posteriors computed from the HMM model, but comparing those with the posteriors obtained from Model 1 and Model 2 may be interesting for future research.

4.3.4.1 Phrase Pair Posterior Probability for Model 1

Let us first define J_{in}, J_{out}, I_{in} and I_{out} in Equation 4.27 for a set of indices i_1, i_2, j_1, j_2:

$$\begin{aligned} J_{\text{in}} &= [j_1, j_2] \\ J_{\text{out}} &= [1, J] \setminus J_{\text{in}} \\ I_{\text{in}} &= [i_1, i_2] \\ I_{\text{out}} &= [0, I] \setminus I_{\text{in}} \end{aligned} \tag{4.27}$$

For Model 1, the numerator from Equation 4.26 is obtained in Equation 4.28:

$$\begin{aligned} &\sum_{a_1^J \in A(j_1,j_2;i_1,i_2)} p_{M_1}(f_1^J, a_1^J \mid e_1^I) \\ &= \sum_{a_1 \in I_{\text{out}}} \cdots \sum_{a_{j_1-1} \in I_{\text{out}}} \sum_{a_{j_1} \in I_{\text{in}}} \cdots \sum_{a_{j_2} \in I_{\text{in}}} \sum_{a_{j_2+1} \in I_{\text{out}}} \cdots \sum_{a_J \in I_{\text{out}}} p_{M_1}(a_1^J, f_1^J \mid e_1^I) \\ &= \sum_{a_1 \in I_{\text{out}}} \cdots \sum_{a_{j_1-1} \in I_{\text{out}}} \sum_{a_{j_1} \in I_{\text{in}}} \cdots \sum_{a_{j_2} \in I_{\text{in}}} \sum_{a_{j_2+1} \in I_{\text{out}}} \cdots \sum_{a_J \in I_{\text{out}}} \frac{\varepsilon}{(1+I)^J} \prod_{j=1}^{J} t(f_j \mid e_{a_j}) \\ &= \frac{\varepsilon}{(1+I)^J} \left(\prod_{j \in J_{\text{out}}} \sum_{i \in I_{\text{out}}} t(f_j \mid e_i) \right) \left(\prod_{j \in J_{\text{in}}} \sum_{i \in I_{\text{in}}} t(f_j \mid e_i) \right) \end{aligned} \tag{4.28}$$

The denominator from Equation 4.26 has already been computed in Equation 4.9. Simplifying Equation 4.28 and Equation 4.9, we obtain Equation 4.29:

$$\begin{aligned} &p_{M_1}(A(j_1, j_2; i_1, i_2) \mid e_1^I, f_1^J) \\ &= \left(\prod_{j \in J_{\text{out}}} \sum_{i \in I_{\text{out}}} \frac{t(f_j \mid e_i)}{\sum_{i'=0}^{I} t(f_j \mid e_{i'})} \right) \left(\prod_{j \in J_{\text{in}}} \sum_{i \in I_{\text{in}}} \frac{t(f_j \mid e_i)}{\sum_{i'=0}^{I} t(f_j \mid e_{i'})} \right) \\ &= \left(\prod_{j \in J_{\text{out}}} \sum_{i \in I_{\text{out}}} p_{M_1}(a_j = i \mid f_1^J, e_1^I) \right) \left(\prod_{j \in J_{\text{in}}} \sum_{i \in I_{\text{in}}} p_{M_1}(a_j = i \mid f_1^J, e_1^I) \right) \end{aligned} \tag{4.29}$$

4.3.4.2 Phrase Pair Posterior Probability for Model 2

To avoid repetition, we skip the derivation which is analogous to the derivation for Model 1. We obtain the phrase pair posterior in Equation 4.30:

$$\begin{aligned} & p_{M_2}(A(j_1, j_2; i_1, i_2) \mid e_1^I, f_1^J) \\ &= \left(\prod_{j \in J_{\text{out}}} \sum_{i \in I_{\text{out}}} p_{M_2}(a_j = i \mid f_1^J, e_1^I) \right) \left(\prod_{j \in J_{\text{in}}} \sum_{i \in I_{\text{in}}} p_{M_2}(a_j = i \mid f_1^J, e_1^I) \right) \end{aligned} \tag{4.30}$$

4.3.4.3 Phrase Pair Posterior Probability for HMM

Let us now compute the phrase pair posterior probability for the HMM model. The denominator from Equation 4.26 can be computed using the forward algorithm while the numerator can be computed using a modified forward algorithm (Deng, 2005). Let us define $\tilde{\alpha}_j(i)$, the modified forward probability in Equation 4.31:

$$\tilde{\alpha}_j(i) = p_{\text{HMM}}(A(j_1, j_2; i_1, i_2), f_1^j, a_j = i \mid e_1^I) \tag{4.31}$$

The numerator from Equation 4.26 can be computed in Equation 4.32:

$$p(A(j_1, j_2; i_1, i_2), f_1^J \mid e_1^I) = \sum_{i=0}^{I} \tilde{\alpha}_J(i) \tag{4.32}$$

The denominator from Equation 4.26 can be computed using the regular forward probability in Equation 4.33:

$$p(f_1^J \mid e_1^I) = \sum_{i=0}^{I} \alpha_J(i) \tag{4.33}$$

Like the regular forward probability, the modified forward probability can also be computed recursively. We can also write the modified forward probability as in Equation 4.34:

$$\begin{aligned} \tilde{\alpha}_j(i) &= p_{\text{HMM}}(A(j_1, j_2; i_1, i_2), f_1^j, a_j = i \mid e_1^I) \\ &= \sum_{a_1^j \in A(j_1, j_2; i_1, i_2)} p_{\text{HMM}}(a_1^{j-1}, f_1^j, a_j = i \mid e_1^I) \\ &= \sum_{\substack{a_1^j \in A(j_1, j_2; i_1, i_2) \\ a_j = i}} p_{\text{HMM}}(a_1^{j-1}, f_1^j, a_j = i \mid e_1^I) \end{aligned} \tag{4.34}$$

The computation of $\tilde{\alpha}_j(i)$ is by a constrained forward algorithm where the constraint is given in Equation 4.35. This is because an alignment in $A(j_1, j_2; i_1, i_2)$ cannot have a link from inside the phrase pair to outside the phrase pair (see Equation 4.25):

$$\forall (j, i) \in J_{\text{out}} \times I_{\text{in}} \cup J_{\text{in}} \times I_{\text{out}}, \tilde{\alpha}_j(i) = 0 \tag{4.35}$$

For a link $(j, i) \in J_{\text{out}} \times I_{\text{out}} \cup J_{\text{in}} \times I_{\text{in}}$ that satisfies the constraint from Equation 4.35, we can derive the modified forward probability in Equation 4.36:

$$\begin{aligned}
\tilde{\alpha}_j(i) &= p_{\text{HMM}}(A(j_1, j_2; i_1, i_2), f_1^j, a_j = i \mid e_1^I) \\
&= \sum_{a_1^{j-1} \in A(j_1, j_2; i_1, i_2)} p_{\text{HMM}}(f_j, f_1^{j-1}, a_j = i, a_1^{j-1} \mid e_1^I) \\
&= \sum_{a_1^{j-1} \in A(j_1, j_2; i_1, i_2)} p_{\text{HMM}}(f_j \mid f_1^{j-1}, a_j = i, a_1^{j-1}, e_1^I) \times \\
&\qquad p_{\text{HMM}}(a_j = i \mid f_1^{j-1}, a_1^{j-1}, e_1^I) \times \\
&\qquad p_{\text{HMM}}(f_1^{j-1}, a_1^{j-1} \mid e_1^I) \\
&= p_{\text{HMM}}(f_j \mid e_i) \sum_{a_1^{j-1} \in A(j_1, j_2; i_1, i_2)} p_{\text{HMM}}(a_j = i \mid a_{j-1}, I) \; p_{\text{HMM}}(f_1^{j-1}, a_1^{j-1} \mid e_1^I) \\
&= p_{\text{HMM}}(f_j \mid e_i) \sum_{k=0}^{I} \sum_{\substack{a_1^{j-1} \in A(j_1, j_2; i_1, i_2) \\ a_{j-1} = k}} p_{\text{HMM}}(a_j = i \mid a_{j-1} = k, I) \\
&\qquad p_{\text{HMM}}(f_1^{j-1}, a_{j-1} = k, a_1^{j-2} \mid e_1^I) \\
&= p_{\text{HMM}}(f_j \mid e_i) \sum_{k=0}^{I} p_{\text{HMM}}(a_j = i \mid a_{j-1} = k, I) \\
&\qquad \sum_{\substack{a_1^{j-1} \in A(j_1, j_2; i_1, i_2) \\ a_{j-1} = k}} p_{\text{HMM}}(f_1^{j-1}, a_{j-1} = k, a_1^{j-2} \mid e_1^I) \\
&= p_{\text{HMM}}(f_j \mid e_i) \sum_{k=0}^{I} p_{\text{HMM}}(a_j = i \mid a_{j-1} = k, I) \; \tilde{\alpha}_{j-1}(k)
\end{aligned} \tag{4.36}$$

We will use the phrase pair posterior probabilities under the HMM model in order to define ranking and counting functions.

4.3.4.4 Constraints, Ranking and Counting Functions from HMM Link and Phrase Posterior Probabilities

In order to keep the size of the rule set manageable, we use the same source constraints, alignment constraints and target constraints as for link posterior extraction defined in Section 4.3.3. We use the phrase pair posterior probabilities under the HMM model both for ranking and scoring extracted rules. This approach assigns a fractional count to each extracted rule, which allows finer estimation of the source-to-target and target-to-source translation models. The ranking and counting functions are defined in Equation 4.37:

$$\mathcal{R}(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = \mathcal{C}(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = p_{\mathrm{HMM}}(A(j_1, j_2; i_1, i_2) \mid f_1^j, e_1^J) \tag{4.37}$$

Under the framework described in Section 4.3.1, we have described standard rule extraction from Viterbi alignments and two novel approaches to rule extraction based on link posterior probabilities and phrase pair posterior probabilities. In order to expose concepts with more clarity, we have restricted the presentation to the extraction of phrase based rules as opposed to hierarchical rules. We will now show how to generalise the techniques presented so far to the extraction of hierarchical rules.

4.3.5 Hierarchical Rule Extraction

In this section, we extend the techniques presented so far to hierarchical rule extraction. In order to avoid repetition, we describe these techniques for the rule pattern $\langle wXw, wXw \rangle$ only (see Section 2.6.2.4 for the definition of patterns). We first rewrite the algorithm in Figure 4.3 into the algorithm in Figure 4.4 for this pattern. We will now describe constraints, counting and ranking functions for standard Viterbi extraction and for extraction from alignment posteriors.

4.3.5.1 Hierarchical Rule Extraction from Viterbi Alignment Links

We now describe the constraints, ranking and counting functions for hierarchical rule extraction from Viterbi alignments:

```
1: function ExtractRules($f_1^J, e_1^I, \boldsymbol{a}$)
2:     for $1 \leq j_1 \leq j_2 < j_3 \leq j_4 \leq J$ do
3:         if $\neg\mathcal{C}_S(f_{j_1}^{j_2} X f_{j_3}^{j_4})$ then                    ▷ Source constraints
4:             continue
5:         end if
6:         $T \leftarrow \emptyset$                         ▷ Sorted hierarchical target phrases
7:         for $1 \leq i_1 \leq i_2 < i_3 \leq i_4 \leq I$ do
8:             if $\mathcal{C}_A(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}, \boldsymbol{a})$ then          ▷ Alignment constraints
9:                 $T \leftarrow T \cup e_{i_1}^{i_2} X e_{i_3}^{i_4}$
10:            end if
11:        end for
12:        Sort($T, \mathcal{R}$)  ▷ Hierarchical target phrases ranked according to $\mathcal{R}$
13:        for $e_{i_1}^{i_2} X e_{i_3}^{i_4} \in T$ do
14:            if $\mathcal{C}_T(e_{i_1}^{i_2} X e_{i_3}^{i_4}, T)$ then                    ▷ Target constraints
15:                Extract($X \rightarrow \langle f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4} \rangle$, $\mathcal{C}(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4})$)
16:            end if
17:        end for
18:    end for
19: end function
```

Figure 4.4: General procedure for hierarchical phrase-based rule extraction. The procedure is presented for the pattern $\langle wXw, wXw \rangle$ only. This algorithm is analogous to the algorithm used to extract phrase-based rules and presented in Figure 4.3.

- source constraints $\mathcal{C}_S(f_{j_1}^{j_2} X f_{j_3}^{j_4})$:

$$\begin{aligned}(j_2 - j_1 + 1) + 1 + (j_4 - j_3 + 1) &\le s_{\text{max elements}} \\ j_2 - j_1 &< s_{\text{max terminals}} \\ j_4 - j_3 &< s_{\text{max terminals}} \\ \text{span}(X) &\le s_{\text{max NT}}\end{aligned} \tag{4.38}$$

  We remind definitions introduced in Section 3.5.2 for the thresholds in Equation 4.38. $s_{\text{max elements}}$ is the maximum number of terminals and nonterminals in the source. $s_{\text{max terminals}}$ is the maximum number of consecutive terminals in the source. $s_{\text{max NT}}$ is the maximum *span* for a nonterminal, i.e. the maximum number of terminals covered by a nonterminal. These thresholds are defined experimentally.

- alignment constraints $\mathcal{C}_A(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}, \boldsymbol{a})$:

$$\begin{aligned}&\forall (j,i) \in \boldsymbol{L}, j \in [j_1, j_4] \Leftrightarrow i \in [i_1, i_4] \\ &\boldsymbol{L} \cap [j_1, j_4] \times [i_1, i_4] \neq \emptyset \\ &\forall (j,i) \in \boldsymbol{L}, j \in (j_2, j_3) \Leftrightarrow i \in (i_2, i_3) \\ &\boldsymbol{L} \cap \{j_2 + 1\} \times (i_2, i_3) \neq \emptyset \\ &\boldsymbol{L} \cap \{j_3 - 1\} \times (i_2, i_3) \neq \emptyset \\ &\boldsymbol{L} \cap (j_2, j_3) \times \{i_2 + 1\} \neq \emptyset \\ &\boldsymbol{L} \cap (j_2, j_3) \times \{i_3 - 1\} \neq \emptyset\end{aligned} \tag{4.39}$$

  The first two constraints require that that the phrase pair $(f_{j_1}^{j_4}, e_{i_1}^{i_4})$ be consistent with the links $\boldsymbol{L}$. The third constraint requires that there be no link from inside the phrase pair $(f_{j_2+1}^{j_3-1}, e_{i_2+1}^{i_3-1})$, which corresponds to the nonterminal X, to outside the phrase pair. The last four constraints mean that the boundary words $f_{j_2+1}, f_{j_3-1}, e_{i_2+1}, e_{i_3-1}$ are not unaligned.

- target constraints $\mathcal{C}_T(e_{i_1}^{i_2} X e_{i_3}^{i_4}, T)$: no constraint. Again, other implementations may impose for example length-based constraints on the target.

- ranking and counting functions:

$$\mathcal{R}(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}) = \mathcal{C}(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}) = 1 \tag{4.40}$$

4.3.5.2 Hierarchical Rule Extraction from Link Posterior Probabilities

We now describe the constraints, ranking and counting functions for the link posterior extraction of hierarchical rules:

- source constraints $\mathcal{C}_S(f_{j_1}^{j_2} X f_{j_3}^{j_4})$:

$$\begin{aligned} j_2 - j_1 &< s_{\text{hier max}} \\ j_4 - j_3 &< s_{\text{hier max}} \\ j_3 - j_2 &< s_{\text{max NT}} \end{aligned} \tag{4.41}$$

  $s_{\text{hier max}}$ is the maximum length for consecutive terminals in the source. $s_{\text{max NT}}$ is the maximum number of terminals covered by a nonterminal. These thresholds are defined experimentally.

- For alignment constraints, let us first define J_{in}, J_{out}, I_{in} and I_{out} in Equation 4.42 for a set of indices i_1, i_2, i_3, i_4, j_1, j_2, j_3, j_4:

$$\begin{aligned} J_{\text{in}} &= [j_1, j_2] \cup [j_3, j_4] \\ J_{\text{out}} &= [1, J] \setminus J_{\text{in}} \\ I_{\text{in}} &= [i_1, i_2] \cup [i_3, i_4] \\ I_{\text{out}} &= [0, I] \setminus I_{\text{in}} \end{aligned} \tag{4.42}$$

  Alignment constraints $\mathcal{C}_A(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}, \boldsymbol{a})$ are defined in Equation 4.43.

$$\begin{aligned} &\exists (j,i) \in (j_2, j_3) \times (i_2, i_3), p(a_j = i \mid f_1^J, e_1^I) > \lambda \\ &\exists (j,i) \in [j_1, j_2] \times I_{\text{in}}, p(a_j = i \mid f_1^J, e_1^I) > \lambda \\ &\exists (j,i) \in [j_3, j_4] \times I_{\text{in}}, p(a_j = i \mid f_1^J, e_1^I) > \lambda \\ &\exists (j,i) \in J_{\text{in}} \times [i_1, i_2], p(a_j = i \mid f_1^J, e_1^I) > \lambda \\ &\exists (j,i) \in J_{\text{in}} \times [i_3, i_4], p(a_j = i \mid f_1^J, e_1^I) > \lambda \\ &\forall (j,i) \in [1, J] \times [1, I] \cap \{(j,i) : p(a_j = i \mid f_1^J, e_1^I) > \lambda\} \\ &\quad j \in (j_2, j_3) \Leftrightarrow i \in (i_2, i_3) \\ &\quad j \in J_{\text{in}} \Leftrightarrow i \in I_{\text{in}} \end{aligned} \tag{4.43}$$

- target constraints $\mathcal{C}_T(e_{i_1}^{i_2} X e_{i_3}^{i_4}, T)$:

$$\begin{aligned} i_2 - i_1 &< t_{\text{hier max}} \\ i_4 - i_3 &< t_{\text{hier max}} \end{aligned} \tag{4.44}$$

$t_{\text{hier max}}$ is the maximum number of consecutive terminals in the target side. Another constraint requires to pick the first k targets per source according to the ranking function $\mathcal{R}$. In the experiments to follow, $k = 3$.

- ranking function:

$$\mathcal{R}(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}) = \prod_{j \in J_{\text{in}}} \sum_{i \in I_{\text{in}}} \frac{p(a_j = i \mid f_1^J, e_1^I)}{i_2 - i_1 + i_4 - i_3 + 2} \tag{4.45}$$

- counting function:

$$\mathcal{C}(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}) = 1 \tag{4.46}$$

4.3.5.3 Hierarchical Rule Extraction from Phrase Posterior Probabilities

For hierarchical rule extraction based on phrase pair posteriors, we use the same constraints as for hierarchical rule extraction based on link posteriors. The ranking and counting functions are defined in terms of alignment posterior probabilities over hierarchical phrase pairs.

We define $A(j_1, j_2; j_3, j_4; i_1, i_2; i_3, i_4)$ in Equation 4.47.

$$A(j_1, j_2; j_3, j_4; i_1, i_2; i_3, i_4) = \{a_1^J : a_j \in I_{\text{in}} \Leftrightarrow j \in J_{\text{in}}\} \tag{4.47}$$

where I_{in} and J_{in} have been defined in Equation 4.42. We can now define the ranking and counting functions in Equation 4.48.

$$\begin{aligned} \mathcal{R}(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}) &= p_{\text{HMM}}(A(j_1, j_2; j_3, j_4; i_1, i_2; i_3, i_4) \mid f_1^j, e_1^J) \\ \mathcal{C}(f_{j_1}^{j_2} X f_{j_3}^{j_4}, e_{i_1}^{i_2} X e_{i_3}^{i_4}) &= p_{\text{HMM}}(A(j_1, j_2; j_3, j_4; i_1, i_2; i_3, i_4) \mid f_1^j, e_1^J) \end{aligned} \tag{4.48}$$

In order to compute these functions for the HMM model, we use the same constrained forward algorithm from Section 4.3.4. This time, the constraints are given by Equation 4.49.

$$\forall (j, i) \in J_{\text{out}} \times I_{\text{in}} \cup J_{\text{in}} \times I_{\text{out}}, \tilde{\alpha}_j(i) = 0 \tag{4.49}$$

where I_{in}, J_{in}, I_{out} and J_{out} have been defined in Equation 4.42.

We have presented a framework to extract phrase based rules and hierarchical rules. This framework encompasses standard extraction from Viterbi alignment links as well as two novel methods that use link posterior probabilities and phrase pair posterior probabilities. We will now evaluate our original methods, first by assessing the expressiveness of the grammars extracted with these methods and then by measuring translation quality.

G_0	G_1	G_2	G_3	G_4
$S \rightarrow \langle X,X \rangle$	$X \rightarrow \langle w\ X,X\ w \rangle$	$X \rightarrow \langle w\ X,X\ w \rangle$	$X \rightarrow \langle w\ X,X\ w \rangle$	$X \rightarrow \langle w\ X,X\ w \rangle$
$S \rightarrow \langle S\ X,S\ X \rangle$	$X \rightarrow \langle X\ w,w\ X \rangle$	$X \rightarrow \langle X\ w,w\ X \rangle$	$X \rightarrow \langle X\ w,w\ X \rangle$	$X \rightarrow \langle X\ w,w\ X \rangle$
$X \rightarrow \langle w,w \rangle$		$X \rightarrow \langle w\ X,w\ X \rangle$	$X \rightarrow \langle w\ X,w\ X \rangle$	$X \rightarrow \langle X_1 w X_2,w X_2 X_1 \rangle$
			$X \rightarrow \langle w\ X\ w,w\ X\ w \rangle$	$X \rightarrow \langle X_1 w X_2,X_2 X_1 w \rangle$

Table 4.1: Hierarchical phrase-based grammars containing different types of rules. The grammar expressiveness is greater as more types of rules are included. In addition to the rules shown in the respective columns, G_1, G_2, G_3 and G_4 also contain the rules of G_0.

4.4 Experiments

In this section, we carry out experiments in order to demonstrate the effectiveness of our original grammar extraction method. We will assess the quality of the grammars extracted in terms of how expressive these grammars are, and in terms of translation quality. In Section 4.4.1, we define the patterns to be included in the grammars used for experiments. In Section 4.4.2, we describe our experimental setup. In Section 4.4.3 and Section 4.4.4, we report results for grammar expressiveness and translation quality. In Section 4.4.5, we compare our two original methods: link posterior extraction vs. phrase pair posterior extraction. Finally, in Section 4.4.6, we explore various methods to exploit information from source-to-target and target-to-source alignment models.

4.4.1 Experimental Setup: Grammar Pattern Definition

In this section we define the hierarchical phrase-based grammars we use for translation experiments. Each grammar is defined by the patterns it contains (see Section 2.6.2.4). We first start with a basic grammar G_0 defined in the left-most column of Table 4.1. G_0 is a monotonic phrase-based translation grammar. It includes all phrase-based rules, represented by the rule pattern $X \rightarrow \langle w, w \rangle$, and the two glue rules that allow concatenation.

Our approach is to extend this grammar by successively incorporating sets of hierarchical rules, or patterns. The goal is to obtain a grammar with few rule patterns but capable of generating a large collection of translation candidates for a given input sentence.

Patterns were added to the base grammar G_0 according to their frequency of use in a translation experiment. We analysed one iteration of minimum error rate training. For each sentence, the decoder generates a 1000-best list of hypotheses. For each hypothesis, the decoder also generates its single best derivation. We extracted rules from the single best derivations of the single best hypotheses and analysed their pattern frequency. Table 4.2 shows the most frequently used rule patterns. We assume that if a rule pattern is used frequently, this means that it is needed in a grammar to produce an acceptable translation.

⟨source , target⟩
$\langle w \ , \ w \rangle$
$\langle w\ X\ w \ , \ w\ X\ w \rangle$
$\langle w\ X \ , \ X\ w \rangle$
$\langle X\ w \ , \ w\ X \rangle$
$\langle X2\ w\ X1 \ , \ X1\ X2\ w \rangle$
$\langle X2\ w\ X1 \ , \ w\ X1\ X2 \rangle$
$\langle X2\ w\ X1 \ , \ X1\ w\ X2 \rangle$
$\langle X\ w \ , \ w\ X\ w \rangle$
$\langle X2\ w\ X1 \ , \ X1\ w\ X2\ w \rangle$
$\langle X2\ w\ X1 \ , \ w\ X1\ w\ X2 \rangle$
$\langle w\ X1\ w\ X2 \ , \ w\ X1\ X2 \rangle$
$\langle w\ X \ , \ w\ X\ w \rangle$
$\langle w\ X\ w \ , \ w\ X \rangle$
$\langle X1\ w\ X2\ w \ , \ X1\ w\ X2 \rangle$

Table 4.2: Most frequently used rule patterns. Frequency was measured by running the decoder on a development set, and counting rule patterns on single best derivations of single best hypotheses. We hypothesise that a frequent rule pattern is needed to produce a translation with good quality.

The grammars used in experiments are summarised in Table 4.1. Each of these grammars will first be evaluated for *expressiveness*: we will measure how often a target sentence can be recovered from a source sentence in decoding. We will also evaluate these grammars for translation quality.

4.4.2 Experimental Setup

We report on experiments in Chinese to English translation. Parallel training data consists of the collections listed in Table 4.3. This is approximately 50M words per language. We report translation results on a development set

ADSO_v5	LDC2006E26	LDC2007E87
LDC2002L27	LDC2006E34	LDC2008E40
LDC2003E07	LDC2006E85	LDC2008E56
LDC2003E14	LDC2006E92	LDC2008G05
LDC2005E83	LDC2006G05	LDC2009E16
LDC2005T06	LDC2007E06	LDC2009G01
LDC2005T10	LDC2007E101	proj_syndicate
LDC2005T34	LDC2007E103	UMD_NewsExplorer
LDC2006E24	LDC2007E46	Wikipedia

Table 4.3: List of collections used as parallel training data for translation experiments.

tune-nw and a test set *test-nw1*. These contain translations produced by the GALE program and portions of the newswire sections of NIST MT02 through MT06.[1] They contain 1755 sentences and 1671 sentences respectively. Results are also reported on a smaller held-out test set *test-nw2*, containing 60% of the NIST newswire portion of MT06, that is, 369 sentences. In Section 4.4.5, the entire newswire portion of MT06 is used and the test set is named *test-nw3*.

The parallel texts for both language pairs are aligned using MTTK (Deng and Byrne, 2005, 2008). For decoding, we use HiFST (see Section 2.9). The language model is a 4-gram language model estimated over the English side of the parallel text and the AFP and Xinhua portions of the English Gigaword Fourth Edition (LDC2009T13) for the first pass translation, interpolated with a zero-cutoff Stupid Backoff 5-gram (see Section 2.7.4 and Section 3.2) estimated using 10.5B words of English newswire text for 5-gram language model rescoring. In tuning the systems, standard minimum error rate training iterative parameter estimation under BLEU is performed on the development set.

Rules with a source-to-target probability less than 0.01 are discarded. In addition, rules that did not occur more than n_{obs} times are discarded. For Viterbi extraction, $n_{obs} = 1$. For link posterior extraction, $n_{obs} = 2$ and for phrase pair posterior extraction, $n_{obs} = 0.2$. These thresholds offer were defined experimentally and provide competitive performance for the various

[1] http://www.itl.nist.gov/iad/mig/tests/mt

conditions without giving unreasonably slow decoding times. The thresholds introduced in previous sections are defined as follows:

- s_{max} is set to 9 for Viterbi extraction and to 5 for posterior extraction. A preliminary experiment showed that for posterior extraction, setting s_{max} to 9 did not provide any improvements and slowed down decoding times.
- $s_{\text{max elements}}$ is set to 5.
- $s_{\text{max terminals}}$ is set to 5.
- $s_{\text{max NT}}$ is set to 10.
- λ is set to 0.5.
- $s_{\text{hier max}}$ and $t_{\text{hier max}}$ are set to 3.

The thresholds relating to Viterbi extraction (s_{max}, $s_{\text{max elements}}$, $s_{\text{max terminals}}$, $s_{\text{max NT}}$) are standard in our translation systems. λ, $s_{\text{hier max}}$ and $t_{\text{hier max}}$ were chosen experimentally and provide competitive performance without slowing down decoding unreasonably.

4.4.3 Grammar Coverage

We first evaluate the grammars defined in Section 4.4.1 for expressiveness. We measure the ability of different grammars to produce a reference translation given an input sentence. Rules are extracted from the sentence pair we want to align and we run the decoder in alignment mode (de Gispert et al., 2010a), which is equivalent to replacing the language model by an acceptor for the reference translation. We compare the percentage of references that can be successfully produced by grammars G_0, G_1, G_2 and G_3[2] for the following extraction methods:

- **Viterbi (V)**: this is the standard extraction method based on a set of alignment links. We distinguish four cases, depending on the model used to obtain the set of links: source-to-target (**V-st**), target-to-source (**V-ts**), and two common symmetrisation strategies: union (**V-union**) and grow-diag-final (**V-gdf**) (see Section 2.3.2).

[2] We did not include G_4 in coverage analysis

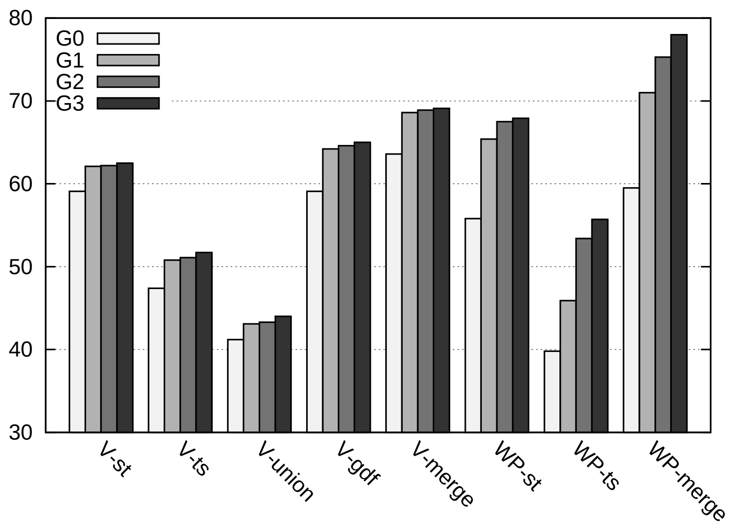

Figure 4.5: Percentage of parallel sentences successfully aligned for various extraction methods and grammars.

- **Word Link Posteriors (WP)**: the extraction method is based on link posteriors described in Section 4.3.3. These rules can be obtained either from the posteriors of the source-to-target (**WP-st**) or the target-to-source (**WP-ts**) alignment models. We apply the constraints described in Section 4.3.3. We do not report alignment percentages when using phrase pair posteriors as they are roughly identical to the **WP** case.

- Finally, in both cases, we also report results when merging the extracted rules in both directions into a single rule set (**V-merge** and **WP-merge**).

Figure 4.5 shows the results obtained for a random selection of 10,000 parallel corpus sentences. As expected, we can see that for any extraction method, the percentage of aligned sentences increases when switching from G_0 to G_1, G_2 and G_3. Posterior-based extraction is shown to outperform standard methods based on a Viterbi set of alignment links for nearly all grammars. The highest alignment percentages are obtained when merging rules obtained under models trained in each direction (**WP-merge**), approximately reaching 80% for grammar G_3.

The maximum rule span ($s_{\text{max span}}$ defined in Section 3.5.2) in alignment was allowed to be 15 words, so as to be similar to translation, where the maximum rule span is 10 words. Relaxing this in alignment to 30 words yields approximately 90% coverage for **WP-merge** under G_3.

We note that if source constraints, alignment constraints and target constraints were not applied, then alignment percentages would be 100% even for G_0, but the extracted grammar would include many noisy rules with poor generalisation power, for example entire sentence pairs.

4.4.4 Translation Results

In this section we investigate the translation performance of each hierarchical grammar, as defined by rules obtained from three rule extraction methods:

- **Viterbi union (V-union)**: standard rule extraction from the union of the source-to-target and target-to-source alignment link sets.
- **Word Posteriors (WP-st)**: extraction based on word posteriors as described in Section 4.3.3. The posteriors are provided by the source-to-target alignment model.
- **Phrase Posteriors (PP-st)**: extraction based on alignment posteriors over phrase pairs, as described in Section 4.3.4, with fractional counts equal to the phrase pair posterior probability under the source-to-target alignment model.

Table 4.4 reports the translation results. It also shows the following decoding statistics as measured on the *tune-nw* set: decoding time in seconds per input word, and number of instances of search pruning per input word. Preliminary experimental work was conducted on grammars G_1 and G_2. Because of slow decoding times, G_4 was not included in this set of experiments; instead, grammar G_4 was used to contrast the **WP** and **PP** methods in Section 4.4.5. As a contrast, we extract rules according to the heuristics introduced by Chiang (2007) and apply the filters described by Iglesias et al. (2009a) to generate a standard hierarchical phrase-based grammar G_H described in Section 2.6.2.4. This uses rules with up to two nonadjacent nonterminals, but excludes identical rule types such as $X \rightarrow \langle w\ X, w\ X \rangle$ or $X \rightarrow \langle w\ X_1\ w\ X_2, w\ X_1\ w\ X_2 \rangle$, which were reported to cause computational difficulties without a clear improvement in translation (Iglesias et al., 2009a).

Grammar	Extraction	# Rules	*tune-nw*			*test-nw1*	*test-nw2*
			time	*prune*	BLEU	BLEU	BLEU
G_H	**V-union**	979k	3.7	0.3	35.1	35.6	37.6
G_1	**V-union**	614k	0.4	0.0	33.6	34.6	36.4
	WP-st	920k	0.9	0.0	34.3	34.8	37.5
	PP-st	894k	1.4	0.0	34.4	35.1	37.7
G_2	**V-union**	735k	1.0	0.0	34.5	35.4	37.2
	WP-st	1132k	5.8	0.5	35.1	36.0	37.7
	PP-st	1238k	7.8	0.7	35.5	36.4	38.2
G_3	**V-union**	967k	1.2	0.0	34.9	35.3	37.0
	WP-st	2681k	8.3	1.1	35.1	36.2	37.9
	PP-st	5002k	10.7	2.6	35.5	36.4	38.5

Table 4.4: Chinese to English translation results with alternative grammars and extraction methods (lower-cased BLEU shown). The number of rules and time (secs/word) and prune (times/word) measurements are done on *tune-nw* set.

Condition	*tune-nw*	*test-nw1*
G_H/**V-union**	34.60/34.56/0.02/0.41	34.89/34.73/0.03/0.52
G_2/**PP-st**	34.74/34.56/0.02/0.44	35.04/34.91/0.06/0.62

Table 4.5: Repeated MERT runs with different random parameter initialisations. Original first-pass BLEU score, BLEU score mean, variance and spread (in original/mean/variance/spread format) are reported for two conditions. The results show that the optimisation is relatively stable.

MERT stability For the results reported in Table 4.4, the MERT parameter optimisation was carried out once. Following Clark et al. (2011), we run the MERT parameter optimisation step 10 times from different initial random parameters for the following conditions: G_H/**V-union** and G_2/**PP-st**. We report the original first-pass decoding BLEU score and the first-pass decoding BLEU score mean, variance and spread (i.e. difference between minimum and maximum) for the additional 10 runs on the *tune-nw* and *test-nw1* sets in Table 4.5. The results show that the optimisation is relatively stable. In addition, we can see that the gain in BLEU for our method on the test set is consistent for the original run and for the average of the additional 10 runs.

Grammar complexity As expected, for the standard extraction method (see rows entitled **V-union**), grammar G_1 is shown to underperform all other grammars due to its structural limitations: in parsing the source, G_1 does not allow inversions after concatenating two phrases. Grammar G_2 obtains much better scores, nearly generating the same translation quality as the baseline grammar G_H. Finally, G_3 does not prove able to outperform G_2 consistently, which suggests that the phrase-disjoint rules with one nonterminal are redundant for the translation grammar.

Rule extraction method For all grammars, we find that the proposed extraction methods based on alignment posteriors outperform standard Viterbi-based extraction, with improvements ranging from 0.5 to 1.1 BLEU points for *test-nw1* (depending on the grammar) and from 1.0 to 1.5 for *test-nw2*. In all cases, the use of phrase posteriors **PP** is the best option. Interestingly, we find that G_2 extracted with **WP** and **PP** methods outperforms the more complex G_H grammar as obtained from Viterbi alignments.

Rule set statistics For grammar G_2 and for the *tune-nw* set, Viterbi-based extraction produces 0.7M rules, while the WP and PP extraction methods yield 1.1M and 1.2M rules, respectively. We further analyse the sets of rules $X \to \langle\gamma, \alpha\rangle$ in terms of the number of distinct source and target sequences γ and α which are extracted. Viterbi extraction yields 82k distinct source sequences whereas the WP and PP methods yield 116k and 146k sequences, respectively. In terms of the average number of target sequences for each source sequence, Viterbi extraction yields an average of 8.7 while WP and PP yield 9.7 and 8.4 rules on average. This shows that method **PP** yields wider coverage but with sharper source-to-target rule translation probability distributions than method **WP**, as the average number of translations per rule is determined by a threshold of 0.01 for the minimum source-to-target probability.

Decoding time and pruning in search In connection to the previous comments, we find an increased need for search pruning, and subsequently slower decoding speed, as the search space grows larger with methods **WP** and **PP**. A larger search space is created by the larger rule sets, which allows the system to generate new hypotheses of better quality.

4.4.5 Comparison between WP and PP

Section 4.4.4 has shown that the **PP** extraction method gives the best translation results in our experiments. Reasons for this may be that more rules were extracted and the translation models were better estimated. These results were obtained with a fixed value of the parameter n_{obs} that was found to give good results for each method. Now, we would like to observe the effect of varying n_{obs}. Given an extraction method, we want to observe the effect of decreasing n_{obs}, that is augmenting the size of the rule set. Additionally, we want to obtain two comparable rule sets in terms of statistics such as number of rules, number of different rule source sides for both phrasal and hierarchical rules, etc., for two different extraction methods in order to observe the effect of estimating the translation model with one method versus the other.

Table 4.6 summarises the results. The table shows 4 configurations: the **WP** extraction method with two different values of n_{obs} and the **PP** extraction method with two different values of n_{obs}. Configurations (**WP** n_{obs}=2) and (**PP** n_{obs}=1) give comparable rule sets. Configurations (**WP** n_{obs}=1) and (**PP** n_{obs}=0.2) also give comparable rule sets in terms of size. We first study the effect of decreasing n_{obs} for each extraction method. For the **WP** method, decreasing n_{obs} from 2 to 1 leads to a average decrease of 0.1 BLEU computed on the test sets for different grammars. We believe that increasing the size of the rule set can lead to more pruning and therefore to a degradation in translation performance. For the **PP** method, decreasing the n_{obs} from 1 to 0.2 leads to an average gain of 0.2 BLEU. We conclude that the **PP** method is more robust than the **WP** method with larger sets of rules.

We then study the effect of using phrase pair posteriors (in **PP**) versus using integer counts (in **WP**) to estimate translation models for comparable rule sets. The configurations (**PP** n_{obs}=1) and (**PP** n_{obs}=0.2) with respect to the configurations (**WP** n_{obs}=2) and (**WP** n_{obs}=1) present an average gain of 0.2 BLEU. This shows that the translation model estimation using phrase pair posteriors is beneficial in translation.

4.4.6 Symmetrising Alignments of Parallel Text

In this section, we investigate extraction from alignments (and posterior distributions) over parallel text which are generated using alignment models trained in the source-to-target (**st**) and target-to-source (**ts**) directions. Our motivation is that symmetrisation strategies have been reported to be bene-

	G_1			G_2			G_4		
Configuration	*tune-nw*	*test-nw1*	*test-nw3*	*tune-nw*	*test-nw1*	*test-nw3*	*tune-nw*	*test-nw1*	*test-nw3*
WP n_{obs}=2	34.3	34.8	22.7	35.1	36.0	23.0	34.9	35.5	23.0
WP n_{obs}=1	34.4	35.1	22.4	35.2	36.0	23.2	34.7	35.4	22.4
PP n_{obs}=1	34.5	34.7	22.4	35.3	36.2	23.2	34.8	35.6	23.1
PP n_{obs}=0.2	34.4	35.1	22.8	35.5	36.4	23.3	34.9	35.7	22.9

Table 4.6: Performance comparison measured by lowercase BLEU across different grammars for different values of n_{obs}

ficial for Viterbi extraction methods (Koehn et al., 2003).

Results are shown in Table 4.7 for grammar G_2. We find that rules extracted under the source-to-target alignment models (**V-st**, **WP-st** and **PP-st**) consistently perform better than the **V-ts**, **WP-ts** and **PP-ts** cases. Also, for Viterbi extraction we find that the source-to-target **V-st** case performs better than any of the symmetrisation strategies, which is not consistent with previous findings for non-hierarchical phrase-based systems (Koehn et al., 2003).

We use the **PP** rule extraction method to extract two sets of rules, under the **st** and **ts** alignment models respectively. We now investigate two ways of merging these sets into a single grammar for translation. The first strategy is **PP-merge** and merges both rule sets by assigning to each rule the maximum count assigned by either alignment model. We then extend the previous strategy by adding three binary feature functions to the system, indicating whether the rule was extracted under the **st** model, the **ts** model or both. The motivation is that minimum error rate training can weight rules differently according to the alignment model they were extracted from. However, we do not find any improvement with either strategy.

Finally, we use linearised lattice minimum Bayes-risk decoding to combine translation lattices (see Section 2.10.3) as produced by rules extracted under each alignment direction (see rows named LMBR(**V-st**,**V-ts**) and LMBR(**PP-st**,**PP-ts**)). Gains are consistent when comparing this to applying lattice minimum Bayes' risk to each of the best individual systems (rows named LMBR(**V-st**) and LMBR(**PP-st**)). Overall, the best-performing strategy is to extract two sets of translation rules under the phrase pair posteriors in each translation direction, and then to perform translation twice and combine the output translations.

4.4.7 Additional Language Pair: Russian-English

We also investigated our proposed method in the context of the Russian-English language pair, which presents less reordering challenges than Chinese-English. We repeated experiments from Table 4.4 with the Viterbi method as a baseline and the phrase-pair posterior method, which showed the best results for Chinese-English. We use the same word alignment models and development sets as for the system described in Section 5.2. Results are presented in Table 4.8. For this language pair, the translation quality obtained by the Viterbi method and the phrase-pair extraction method is comparable,

Rule Extraction	*tune-nw*	*test-nw1*	*test-nw2*
V-st	34.7	35.6	37.5
V-ts	34.0	34.8	36.6
V-union	34.5	35.4	37.2
V-gdf	34.4	35.3	37.1
WP-st	35.1	36.0	37.7
WP-ts	34.5	35.0	37.0
PP-st	35.5	36.4	38.2
PP-ts	34.8	35.3	37.2
PP-merge	35.5	36.4	38.4
PP-merge-MERT	35.5	36.4	38.3
LMBR(**V-st**)	35.0	35.8	38.4
LMBR(**V-st,V-ts**)	35.5	36.3	38.9
LMBR(**PP-st**)	36.1	36.8	38.8
LMBR(**PP-st,PP-ts**)	36.4	36.9	39.3

Table 4.7: Translation results under grammar G_2 with individual rule sets, merged rule sets, and rescoring and system combination with lattice-based minimum Bayes' risk (lower-cased BLEU shown)

Grammar	Extraction	# Rules	*tune*	*test1*	*test2*
G_H	**V-union**	886k	33.7	32.7	25.6
	V-union	386k	33.5	32.3	25.6
G_1	**PP-st**	458k	33.4	32.1	25.5
	V-union	500k	33.3	32.2	25.5
G_2	**PP-st**	712k	33.7	32.2	25.6
	V-union	901k	33.7	32.4	25.5
G_3	**PP-st**	2787k	33.7	32.3	25.8

Table 4.8: Russian to English translation results with alternative grammars and extraction methods (lower-cased BLEU shown). Experiments from Table 4.4 were repeated for the baseline and the phrase-pair posterior extraction method.

with differences in the order of 0.1 BLEU. We hypothesise that because there is less reordering between Russian and English, the quality of the Viterbi alignments, therefore the quality of the rules extracted, is high enough to already produce relatively high quality translations.

4.5 Conclusion

We have presented new rule extraction methods that lead to larger rule sets as well as better estimated translation models. In Chinese-English translation experiments, a simple grammar estimated with these methods was shown to outperform a baseline hierarchical grammar extracted from Viterbi alignments. A fine-grained analysis was provided to explain the advantages of the phrase-pair posterior extraction method. Finally, it was shown that the best way to exploit both alignment directions is to train separate models for each direction, translate separately and combine the lattice outputs. In addition, the same techniques were explored in the context of Russian-English translation and provided the same translation quality as the baseline technique.

Chapter 5

Domain Adaptation for Hierarchical Phrase-Based Translation Systems

In Chapter 3, we have demonstrated improvements in scalability for generation of and retrieval from hierarchical phrase-based grammars. In Chapter 4, we have introduced an original model for hierarchical phrase-based grammar extraction and estimation. In this chapter, we will investigate further refinements for hierarchical phrase-based grammar modelling in the context of domain adaptation. We will also investigate possible improvements for language modelling both in the context of domain adaptation and in terms of training data size.

5.1 Introduction

As described in Section 1.2.1, SMT state-of-the-art systems consist of a pipeline of various modules. These modules interact in complex ways, which makes experimentation challenging, but also provides an opportunity to make improvements in separate modules with the hope that the end-to-end system will be improved overall. In this chapter, we describe investigations into language model and grammar refinement techniques for SMT. We compare various strategies for grammar and language modelling in the context of domain adaptation. We also explore the interaction between a first pass language model used in decoding and a second pass language model used in rescoring. Finally, we compare two smoothing strategies for large language model rescoring. The techniques presented in this chapter may be used in order to build high quality systems for a translation evaluation or to customise a translation system for a client in the industry setting. Figure 5.1 indicates which modules we attempt to improve.

Experiments are based on a system submitted to the WMT13 Russian-English translation shared task (Pino et al., 2013). Because we developed a very competitive system for this evaluation, refinements on that system give a good indication of which techniques are worthwhile applying to SMT systems. We briefly summarise the system building in Section 5.2. In Section 5.3, we review domain adaptation techniques for machine translation. In Section 5.4, we show how to adapt a language model to obtain better performance on a specific domain such as newswire text. In Section 5.5, we show how additional grammar rule features related to specific domains may help in translation. In Section 5.6, we investigate the various strategies for combining first pass translation and language model rescoring in terms of language model training data size and n-gram language model order. Finally, in Section 5.7, we compare two smoothing strategies for large language model rescoring.

5.2 Description of the System to be Developed

The experiments reported in this chapter are based on the system submitted to the WMT13 Russian-English translation shared task (Pino et al., 2013). 19 systems were submitted to this particular task. The CUED system obtained the best case-sensitive BLEU score and the second best human judgement score amongst constrained-track systems (Bojar et al., 2013).[1] In this section, we summarise the system building.

We use all the Russian-English parallel data available in the constrained track. We filter out non Russian-English sentence pairs with the *language-detection* library.[2] A sentence pair is filtered out if the language detector detects a different language with probability more than 0.999995 in either the source or the target. This discards 78,543 sentence pairs from an initial 2,543,462. For example, the sentence in Figure 5.2 was detected as French with a probability of 0.999997. In addition, sentence pairs where the source sentence has no Russian character, defined by the Perl regular expression

[1] http://matrix.statmt.org/matrix/systems_list/1738
[2] http://code.google.com/p/language-detection/

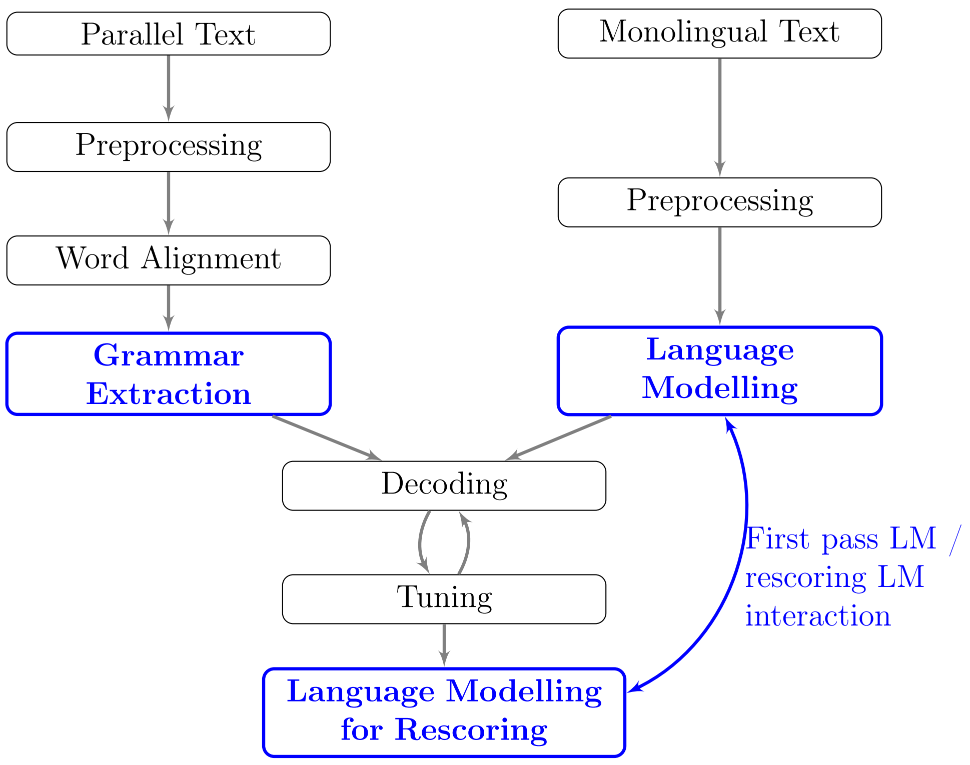

Figure 5.1: Machine Translation System Development Pipeline. Modules that we attempt to refine are in bold blue. Grammar extraction and first-pass language modelling are refined in the context of domain adaptation. We also investigate smoothing for language modelling in rescoring. Finally, we study the interaction between the first-pass language model and the rescoring language model.

Sur base des précédents avis, il semble que l'hôtel a fait un effort.

Figure 5.2: Example of target side sentence of a Russian-English sentence pair. The sentence was detected as actually being French with 0.999998 probability.

[\x0400-\x04ff], are discarded. This regular expression corresponds to the Unicode block for Cyrillic characters (in hexadecimal notation, code 0400 to code 04ff). This further discards 19000 sentence pairs. We take the view that discarding a small portion of training data in order to obtain cleaner data is beneficial in translation, when such cues are easily and reliably available.

The Russian side of the parallel corpus is tokenised with the Stanford CoreNLP toolkit.[3] The English side of the parallel corpus is tokenised with a standard English tokeniser, which splits sentences into tokens using white space and punctuation, and retains abbreviations as single tokens. Both sides of the parallel corpus are then lowercased. Corpus statistics after filtering and tokenisation are summarised in Table 5.1.

Language	# Tokens	# Types
RU	47.4M	1.2M
EN	50.4M	0.7M

Table 5.1: Russian-English parallel corpus statistics after filtering and tokenisation. Parallel text contains approximately 50M tokens on each side. This translation task can be characterised as a *medium size* translation task.

Parallel data is aligned using the MTTK toolkit (Deng and Byrne, 2008): we train a word-to-phrase HMM model with a maximum phrase length of 4 in both source-to-target and target-to-source directions (see Section 2.3.1). The final alignments are obtained by taking the union of alignments obtained in both directions. A hierarchical phrase-based grammar is extracted from the alignments as described in Section 3.4. The constraints for extraction described in Section 3.5.2 are set as follows:

- $s_{\text{max}} = 9$ (maximum number of source terminals for phrase-based rules)
- $s_{\text{max elements}} = 5$ (maximum number of source terminals and nonterminals)
- $s_{\text{max terminals}} = 5$ (maximum number of source consecutive terminals)
- $s_{\text{max NT}} = 10$ (maximum source nonterminal span)

[3] http://nlp.stanford.edu/software/corenlp.shtml

We are using a shallow-1 hierarchical grammar (see Section 2.6.2.4) in our experiments. This model is constrained enough that the decoder can build exact search spaces, i.e. there is no pruning in search that may lead to search errors under the model.

We use the KenLM toolkit (Heafield et al., 2013) to estimate a single modified Kneser-Ney smoothed 4-gram language model (see Section 2.7.3) on all available monolingual data available in the constrained track. Statistics for the monolingual data are presented in Table 5.2.

Corpus	# Tokens
EU + NC + UN + CzEng + Yx	652.5M
Giga + CC + Wiki	654.1M
News Crawl	1594.3M
afp	874.1M
apw	1429.3M
cna + wpb	66.4M
ltw	326.5M
nyt	1744.3M
xin	425.3M
Total	7766.9M

Table 5.2: Statistics for English monolingual corpora used to train language models. Abbreviations are used for the following corpora: Europarl (EU), News Commentary (NC), United Nations (UN), Czech-English corpus (CzEng), Yandex (Yx), 10^9 French-English corpus (Giga), Common Crawl (CC), Wikipedia Headlines (Wiki). Corpora "afp" and below are the various news agency from the GigaWord corpus.

For first-pass decoding, we use HiFST as described in Section 2.9. First-pass decoding is followed by a large 5-gram language model rescoring step as described in Section 2.10. The 5-gram language model also uses all available monolingual data described in Table 5.2.

Two development sets are available: *newstest2012* with 3003 sentences and *newstest2013* with 3000 sentences. We extract odd numbered sentences from *newstest2012* in order to obtain a tuning set *newstest2012.tune*. Even numbered sentences from *newstest2012* form a test set *newstest2012.test*. The *newstest2013* set is used as an additional test set. *newstest2012.tune*,

Configuration	*tune*	*test1*	*test2*
baseline 1st pass	33.22	32.01	25.35
baseline +5g	33.33	32.26	25.53

Table 5.3: WMT13 Russian-English baseline system. Performance is measured by case-insensitive BLEU. The 1st pass configuration indicates results obtained after decoding. The +5g configuration indicates results obtained after 5-gram language model rescoring.

newstest2012.test and *newstest2013* are respectively shortened to *tune*, *test1* and *test2*.

We report our baseline experiment in Table 5.3. We indicate case-insensitive BLEU score after first-pass decoding and 5-gram language model rescoring. We will now investigate various strategies in order to improve performance over the baseline. We will investigate domain adaptation techniques in order to estimate better language models and grammars for translating newswire data. We will also study the interaction between the first-pass language model and the rescoring language model. Finally, we will study an alternative smoothing technique to Stupid Backoff for the rescoring language model.

5.3 Domain Adaptation for Machine Translation

In this section, we first introduce in Section 5.3.1 the general problem of domain adaptation and how certain machine translation settings may be instances of this problem. We then review previous work on addressing the problem of domain adaptation in SMT in Section 5.3.2.

5.3.1 Domain Adaptation

Let us first formalise the problem of domain adaptation. In a standard multi-class classification problem, we are given a training set $\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i \in [1, N]\}$ where $\mathcal{X}$ is a set of instances to be labelled and $\mathcal{Y}$ is a finite set of labels. The multi-class classification learning problem is to find a function $f : \mathcal{X} \to \mathcal{Y}$ that minimises the number of prediction errors. Machine translation can be seen as a multi-class classification problem where $\mathcal{X}$ is the set of

> In addition, a new seven-year EU budget needs to be passed, which is very complicated due to the current crisis.

Figure 5.3: Example of English sentence in the newswire domain.

all possible source sentences and $\mathcal{Y}$ is the set of all possible target sentences. The multi-class classification framework is not very intuitive for machine translation. First, at least in theory, there are an infinite number of possible target sentences to choose from in translation. In addition, many possible translations are correct or acceptable in general. However, the multi-class classification setting is assumed by the original objective function (see Section 2.2), which we recall in Equation 5.1. This objective function minimises the number of misclassifications.

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} p(\boldsymbol{e} \mid \boldsymbol{f}) \tag{5.1}$$

A *domain* is a particular distribution $\mathcal{D}$ over $\mathcal{X}$. For example, $\mathcal{D}$ can be such that source sentences in $\mathcal{X}$ drawn from $\mathcal{D}$ are in the newswire domain. For example, if source sentences are in English, then source sentences drawn sampled from $\mathcal{X}$ with distribution $\mathcal{D}$ may look like the sentence in Figure 5.3.

For domain adaptation, we assume an out-of-domain distribution $\mathcal{D}_O$ and a in-domain distribution $\mathcal{D}_I$. A model is trained on a sample drawn from $\mathcal{D}_O$ but the model performance is evaluated on a sample drawn from $\mathcal{D}_I$. *Domain adaptation* consists in altering the training procedure by using information from domain $\mathcal{D}_I$ in order to achieve better performance on $\mathcal{D}_I$. The out-of-domain and in-domain distributions are also called source domain and target domain respectively. A simple example of domain adaptation setting for SMT may be that the parallel text is a parallel corpus of tweets[4] and sentences to be translated come from newspaper articles.

5.3.2 Previous Work on Domain Adaptation for SMT

In machine translation, we can distinguish two types of domain adaptation problem: *cross-domain* adaptation and *dynamic* adaptation (Foster and

[4] `https://twitter.com/`

Kuhn, 2007). In cross-domain adaptation, a small amount of in-domain training data is available while in dynamic adaptation, no in-domain training data is available. Dynamic adaptation may be important for online translation systems for example. Our concern is cross-domain adaptation, where at least an in-domain development set is available for parameter tuning.

Koehn and Schroeder (2007) explore various techniques for machine translation domain adaptation. They use Europarl (Koehn, 2005) as out-of-domain training data and a news commentary parallel corpus[5] for in-domain training data. Training the language model on in-domain training data gives an improvement of 0.8 BLEU with respect to the baseline. Training two separated language models on the in-domain and out-of-domain training data and interpolating them with weights set to minimise perplexity on the tuning set gives an improvement of 0.4 BLEU. Using these two language models as separate features to be optimised by MERT (Och, 2003) gives an improvement of 0.6 BLEU. Finally, using two separate translation models trained on the in-domain and out-of-domain data and tuning the weights with MERT gives an improvement of 0.9 BLEU. These results are reported on the tuning set only. In this chapter, we explore very similar techniques.

5.4 Language Model Adaptation for Machine Translation

In this section, we contrast two simple language model adaptation techniques for machine translation. Experiments are carried out on the Russian-English system described in Section 5.2.

We use the KenLM toolkit to estimate separate modified Kneser-Ney smoothed 4-gram language models for each of the corpora listed in Table 5.2. The component models are then interpolated with the SRILM toolkit (Stolcke, 2002) to form a single LM. The interpolation weights are optimised for perplexity on the concatenation of the English side of the *news-test2008*, *newstest2009* and *newssyscomb2009* development sets which were provided for previous translation evaluations, using the *compute-best-mix* tool, which is part of the SRILM toolkit. The weights reflect both the size of the component models and the genre of the corpus the component models are trained on, e.g. weights are larger for larger corpora in the news genre (Foster et al.,

[5] http://statmt.org/wmt07/shared-task.html

Row	Configuration	*tune*	*test1*	*test2*
1	baseline 1st pass	33.22	32.01	25.35
2	baseline +5g	33.33	32.26	25.53
3	linear 1st pass	33.71	32.26	25.47
4	linear +5g	33.74	32.51	25.58
5	loglinear 1st pass	33.23	31.75	25.37
6	loglinear +5g	33.28	31.85	25.48

Table 5.4: Performance comparison, measured by case-insensitive BLEU, between an uninterpolated language model (baseline), a linearly interpolated language model (linear) and a log-linearly interpolated language model (loglinear). Off-line linear interpolation (rows 3 and 4) is the best strategy. The loglinear configuration performs worst, however this may be due to the MERT optimisation algorithm not being able to find a high quality set of parameters with 20 features.

2013). Thus we obtain a linear mixture of language models. We call this configuration "linear". As a contrast, we also keep the individual language models as separate features in optimisation. We call this configuration "loglinear".

We compare the baseline, the linear and the loglinear configurations in Table 5.4. We observe that the best strategy for building a language model is to do offline linear interpolation to minimise perplexity on a development set that has the same genre as the translation test sets (row 3). The second best strategy is to use an uninterpolated language model (row 1). The worst strategy is to do a log-linear interpolation of the various language model components by tuning these language language models as separate features with lattice MERT (row 5). Note that these observations are confirmed even after rescoring with a large Stupid Backoff 5-gram language model (rows 2, 4 and 6). It is possible that the loglinear configuration, which has 20 features as opposed to 12 features for the baseline and the linear configuration, performs worst because the MERT algorithm becomes unstable with more features (Foster and Kuhn, 2009). These results are in contrast to those obtained previously (Koehn and Schroeder, 2007), where the loglinear interpolation was found to provide more benefits over the linear interpolation. In addition, more language models are considered (9 vs. 2). Finally, both tune and test results are reported here.

5.5 Domain Adaptation with Provenance Features

In Section 5.4, we have presented domain adaptation strategies for the language model. In this section, we will focus on domain adaptation for the translation model. Chiang et al. (2011) introduce *provenance* lexical features. The concept of provenance was introduced previously (Matsoukas et al., 2009) and provides metadata information such as genre or collection for each sentence pair in parallel text. Word translation tables and lexical features are computed for each provenance. Finally, for each provenance, the ratio of the provenance specific lexical feature with the global lexical feature is added as a feature to the translation system.

We use a very similar approach with the following differences and extensions:

- The lexical features are computed with Model 1 rather than Viterbi alignments, as described in Section 2.6.5.
- The features added to the system are not the ratios between a provenance specific lexical feature and the global lexical feature but simply the provenance specific lexical features.
- We extend this technique to compute provenance specific source-to-target and target-to-source translation models as described in Section 3.4.

For our experiments, we use 4 provenances that correspond to the 4 subcorpora provided for parallel text: the Common Crawl corpus, the News Commentary corpus, the Yandex corpus and the Wiki Headlines corpus. This gives an additional 16 features to our system: source-to-target and target-to-source translation and lexical features for each provenance. This configuration is called “provenance”.

When retrieving relevant rules for a particular test set, various thresholds are applied, such as number of targets per source or translation probability cutoffs. Thresholds involving source-to-target translation scores are applied separately for each provenance and the union of all surviving rules for each provenance is kept. This configuration is called “provenance union”. This specific configuration was used for submission to the translation task competition at WMT13. To our knowledge, this original technique has not been

Row	Configuration	*tune*	*test1*	*test2*
1	no provenance 1st pass	33.71	32.26	25.47
2	no provenance +5g	33.74	32.51	25.58
3	provenance 1st pass	33.22	32.14	25.40
4	provenance +5g	33.22	32.14	25.41
5	provenance union 1st pass	33.65	32.36	25.55
6	provenance union +5g	33.67	32.58	25.63
7	no provenance union 1st pass	33.22	31.78	25.59
8	no provenance union +5g	33.22	31.78	25.59

Table 5.5: Effect of using provenance features. Case-insensitive BLEU scores are reported. The most effective strategy is provenance union. Because the "no provenance union" does not improve over the baseline, we conclude that the gains obtained by the provenance union strategy are not only due to additional rules; instead, they are due to the conjunction of additional rules with provenance features for better discrimination.

employed previously. Results show that it provides additional gains over the simple use of provenance features.

The first-pass language model is the linear mixture described in Section 5.4. We also compare the various configurations after 5-gram rescoring. Results are presented in Table 5.5. We can see that adding provenance features slightly degrades performance (row 3 vs. row 1, row 4 vs. row 2). This again may be due to the inability of the MERT optimisation algorithm to tune more than the 12 basic features. However, if we use the provenance union strategy, we can see that having additional rules together with provenance features is the best strategy (row 5 vs. row 1, row 6 vs. row 2). In order to verify whether the gains are only due to additional rules, we rerun the experiment with the provenance union configuration but remove the provenance features (row 7 and row 8). The performance of the "no provenance union" configuration is worse or equal to the "no provenance" configuration, which demonstrates that the provenance union strategy is optimal in this set of experiments and that the gains provided are not only due to having additional translation rules. The effectiveness of the provenance union strategy is due to additional rules in conjunction with additional features for better discrimination between rules.

5.6 Interaction between First Pass Language Model and Rescoring Language Model

In Section 5.4 and Section 5.5, we have investigated various strategies to adapt the language model and translation model to the genre of the test set to be translated. In each case, we verified that the conclusions held even after rescoring with a large 5-gram language model. In this section, we study the interaction between the first pass language model and the rescoring language model in terms of n-gram order and training data size. We give recommendations on how much data should be used to train a first pass language model and what order should be chosen for the first pass n-gram language model.

Our translation systems are typically run in two steps. The first step usually consists in decoding with a 4-gram language model. The second step consists in 5-gram lattice rescoring. Brants et al. (2007) argue that single pass decoding is conceptually simpler and may lose less information. In this section, we also attempt to verify this claim by incorporating 5-gram language models directly in first-pass decoding and evaluate performance with respect to a two-pass strategy.

We describe the various configurations used for experiments and reported in Table 5.6. For monolingual data size, we use a “large” configuration which uses all available data described in Table 5.2, and a “medium” configuration which uses the following corpora: EU, NC, UN, CzEng, Yx, Giga, CC, Wiki, afp and xin. (The keys for abbreviations are described in Table 5.2). We also train 4-gram and 5-gram language models with large and medium data for the first pass language model. The “large 4g” configuration corresponds to the linear configuration from Table 5.4.

The initial parameter for the MERT optimisation step was the same for all configurations. This parameter was obtained from a separate Arabic-to-English experiment carried out earlier. Reusing already optimised parameters between experiments allows us to do faster experimentation since less iterations are needed until convergence.

Results are presented in Table 5.6. Comparing row 1 and row 3, we can see that for a 4-gram language model, more training data is beneficial in first-pass decoding. This conclusion is confirmed after 5-gram rescoring (row 2 vs. row 4). However, when comparing row 5 vs. row 7 and row 6 vs. row 8, we conclude that in the case of a first-pass 5-gram language model, more

Row	Configuration	*tune*	*test1*	*test2*
1	medium 4g 1st pass	32.96	31.53	24.60
2	medium 4g +5g	33.43	32.12	25.30
3	large 4g 1st pass	33.71	32.26	25.47
4	large 4g +5g	33.74	32.51	25.58
5	medium 5g 1st pass	32.62	31.62	24.77
6	medium 5g +5g	33.20	31.96	25.27
7	large 5g 1st pass	32.99	31.79	25.18
8	large 5g +5g	32.99	31.79	25.18

Table 5.6: Comparing various data size conditions and n-gram orders for language model training and the effect on large 5-gram language model rescoring. Case-insensitive BLEU scores are reported. In conclusion, more language model training data is helpful but the use of higher order n-gram language models is only beneficial in rescoring.

training data is only helpful in first-pass decoding. It is possible that even more monolingual data is required for a 5-gram language model to generate a lattice of sufficient quality in first-pass decoding so as to obtain improvements in rescoring.

Comparing row 2 vs. row 6 and row 4 vs. row 8, we see that for equal amounts of training data, the best strategy is to use a first-pass 4-gram language model followed by large 5-gram language model rescoring. One possible reason is that a first pass 5-gram language model may not have enough training data to be reliably estimated and therefore may not produce a first-pass lattice of high quality translations to be rescored, whereas a first pass 4-gram language model is able to discard poor translations in first pass decoding. One caveat with this conclusion of course is that the maximum amount of data experimented with is 7.8B words as opposed to 2 trillion words reported by Brants et al. (2007). Future work may be dedicated to fill this gap in experimentation. For example, we plan to exploit a large 5-gram language model (Buck et al., 2014) trained on the Common Crawl corpus[6] in rescoring experiments. We are not aware of any studies in the machine translation literature on interactions between first pass and second pass language models. The main conclusion, i.e. that a two-pass decoding approach

[6] https://commoncrawl.org/

is beneficial over a single-pass approach given the amount of training data, has not been demonstrated experimentally before.

5.7 Language Model Smoothing in Language Model Lattice Rescoring

In Section 5.6, we have given recommendations about what strategy to adopt in first-pass decoding in order to obtain optimal results in rescoring. The conclusion was to use a 4-gram language model trained on large amounts of data. In this section, we attempt to improve on the 5-gram rescoring procedure by exploring two alternative smoothing strategies for the 5-gram language model used for rescoring.

We train a Stupid Backoff 5-gram language model (see Section 2.7.4 and Section 3.2.1) and a modified Kneser-Ney 5-gram language model on all available English data for WMT13, described in Table 5.2. We compare the two smoothing methods in rescoring over various configurations used throughout this chapter. We did not include the “provenance union no provenance” and the “large 5g” configurations. On average, over the seven experiments from Table 5.7, we obtain a gain of 0.11 BLEU on the tuning set *tune*, a gain of 0.14 BLEU on the first test set *test1* and a gain of 0.13 BLEU on the second test set *test2*. We conclude that the use of modified Kneser-Ney smoothing is slightly beneficial in 5-gram lattice rescoring when the amount of training data is in the order of several billion tokens. This confirms conclusions from Figure 5 in a previous publication (Brants et al., 2007) that Kneser-Ney smoothing is superior to Stupid Backoff smoothing with amounts of training data in a similar order to the work here. These conclusions were made for first pass decoding while here. This is the first time these two smoothing strategies are compared in the context of 5-gram language model lattice rescoring. We also note that we obtain an improvement over the system submitted to the WMT13 translation task using the “provenance union + KN 5g” configuration (row 15 in bold vs. row 14).

5.8 Conclusion

In this chapter, we have investigated various refinements on translation system building. We have explored domain adaptation techniques in order to

Row	Configuration	*tune*	*test1*	*test2*
1	baseline 1st pass	33.22	32.01	25.35
2	baseline + SB 5g	33.33	32.26	25.53
3	baseline + KN 5g	33.31	32.28	25.65
4	linear 1st pass	33.71	32.26	25.47
5	linear + SB 5g	33.74	32.51	25.58
6	linear + KN 5g	33.73	32.26	25.48
7	loglinear 1st pass	33.23	31.75	25.37
8	loglinear + SB 5g	33.28	31.85	25.48
9	loglinear + KN 5g	33.30	31.92	25.35
10	provenance 1st pass	33.22	32.14	25.40
11	provenance + SB 5g	33.22	32.14	25.41
12	provenance + KN 5g	33.54	32.37	25.75
13	provenance union 1st pass	33.65	32.36	25.55
14	provenance union + SB 5g	33.67	32.58	25.63
15	provenance union + KN 5g	**33.89**	**32.82**	**25.84**
16	medium 5g 1st pass	32.62	31.62	24.77
17	medium 5g + SB 5g	33.20	31.96	25.27
18	medium 5g + KN 5g	33.24	32.29	25.47
19	medium 4g 1st pass	32.96	31.53	24.60
20	medium 4g + SB 5g	33.43	32.12	25.30
21	medium 4g + KN 5g	33.65	32.46	25.55

Table 5.7: Comparison between the gains obtained by Stupid Backoff 5-gram rescoring and Kneser-Ney 5-gram rescoring. Case-insensitive BLEU scores are reported. In conclusion, at the level of several billion tokens for language model training, Kneser-Ney smoothing is beneficial over Stupid Backoff smoothing in the context of 5-gram lattice rescoring. Row 15 (in bold) achieves a better performance than row 14, which was our system submitted to the WMT13 translation shared task.

refine the language model and the translation model. For language model building, we find that the best strategy for cross-domain adaptation is to build an interpolated language model and tune the interpolation weights in order to minimise the perplexity on a development set in the target domain, as opposed to tuning the language model weights with MERT. We also have examined the effect of provenance features in the translation model. We found that the use provenance features is beneficial only when these features are used to discriminate rules from a larger rule set.

After having attempted to refine models used in first-pass decoding, we have studied the interaction between the first pass language model and the 5-gram language model used in lattice rescoring. We have come to the conclusion that in our setting, a two-pass strategy is preferable and that using large amounts of data to train a 4-gram language model for first-pass decoding yields better results in rescoring.

Finally, we have demonstrated that Kneser-Ney smoothing produces better results than Stupid Backoff smoothing when training a 5-gram language model on 7.8 billion words for rescoring purposes. This confirms previous research in the context of rescoring as opposed to first-pass decoding.

Chapter 6

String Regeneration as Phrase-Based Translation

In the previous chapters, we have proposed various improvements to hierarchical phrase-based translation systems, in terms of infrastructure (Chapter 3), grammar modelling (Chapter 4 and Chapter 5) and language modelling for first pass decoding as well as rescoring (Chapter 5). In the chapter to follow (Chapter 7), we will also introduce techniques for potential improvements in fluency in the output of hierarchical phrase-based systems. In this chapter, we lay the ground work to be applied in translation in Chapter 7.

Machine translation output was originally evaluated by human judges in terms of *adequacy*, or how well the meaning of the source text is preserved in the translated text, and *fluency*, or how well-formed the translation is (White et al., 1993), i.e. to what extent it is syntactically, morphologically, and orthographically correct (Reiter and Dale, 1997). Adequacy and fluency motivate the separation of SMT models into a translation model and a language model (see Section 2.2). There is increasing concern that translation output often lacks fluency (Knight, 2007). In this chapter, we study the related problem of string regeneration: given a sentence where the word order has been scrambled, how difficult is it to recover the original sentence ? Our approach is inspired by phrase-based translation techniques and achieves state-of-the-art performance on the string regeneration task. Software written for this chapter is available online.[1]

[1] `https://github.com/jmp84/NgramGen`

6.1 Introduction

Before the widespread use of automatic metrics such as BLEU (see Section 2.8.1) or METEOR (Banerjee and Lavie, 2005), machine translation systems were originally evaluated in terms of adequacy and fluency. Current popular automatic metrics for translation can be "gamed" in the sense that it is possible to obtain a high score according to these metrics with an output that lacks fluency. For example, a translation may obtain a high BLEU score simply because it contains many n-grams that are also in the reference translation but this translation may not be well-formed syntactically. Figure 6.1 shows how an improvement of 20 BLEU points at the sentence level between an MT hypothesis and an oracle MT hypothesis, i.e. the best performing hypothesis among a set, may not translate into an improvement in fluency. Of course, BLEU is designed to measure quality at the set level, so hopefully other oracle MT hypotheses will probably improve fluency. The nature of the translation metrics may have led research on translation to neglect fluency.

Correct word ordering is an essential part of fluency. Producing a translation with words correctly ordered is a very difficult task. Different word ordering between distant language pairs such as Chinese-English is one of the main reasons for incorrect word ordering in translation. In general, obtaining a correct word order is a fundamental problem in many natural language processing applications. In machine translation, because source language and target language have a different word ordering, a machine translation system needs to model word reordering. In some language pairs such as French-English, the source language gives good cues to the target word order; however, in other language pairs such as Chinese-English or Japanese-English, the word order can be very different in the source and the target language. In natural language generation tasks, such as paraphrasing or summarisation, word ordering is also critical at the surface realisation step (Reiter and Dale, 1997). Word ordering is also one possible type of grammatical error, especially for foreign language learners (Yu and Chen, 2012).

In this chapter, we study the word ordering problem in isolation through the task of string regeneration (Wan et al., 2009). Given an input sentence where the word order has been scrambled, the string regeneration task consists in recovering the original sentence. This task can be seen as one of the steps in surface realisation for language generation where the input is a list of content words and function words and the output a grammatical sentence. It also allows us to study the realisation task independently of the transla-

SMT System (19.76 Sentence-Level BLEU): enda according to jerzy made a televised speech, the president of burundi civil war of resistance army soldiers in the past ten years, and had to settle on january 5 before the completion of the new military leadership will be in place before january 7, four of the former rebel army leader.

SMT System Oracle (39.91 Sentence-Level BLEU): en, a us $according to al made a televised speech, said that the president of the burundi's rebel army soldiers in the past 10-year civil war must be before january 5 and the completion of the new military leadership will be in place by january 7, and four of the former leaders of the rebel forces.

Reference: According to President Ndayizeye's televised speech, former rebel fighters in Burundi's 10-year civil war must be settled in by January 5. The new army leadership is to be made up of 40 percent former rebels and should be in place by January 7.

Figure 6.1: Example showing how a large improvement in BLEU score does not imply an improvement in fluency. The first quote is one of the output sentences produced by an SMT system presented in Chapter 7. The second quote is the oracle translation for that system, i.e. the best possible translation to be found in the lattice of translations generated by the system. The third quote is one of the human reference translations. Note that the oracle hypothesis was picked to maximise sentence-level BLEU, which shows that, in principle, it is possible to obtain a translation with a high BLEU score but that lacks fluency.

tion task. Finally, it can have alternative applications such as automatically producing alternative references for machine translation.

We use a simple but extensible approach, inspired from stack-based decoding for phrase-based translation (see Section 2.5.4). Given a bag of words as input, we use n-gram rules extracted from a large monolingual corpus and concatenate these n-grams in order to form hypotheses. The hypotheses are scored with a linear model in the same manner as translation hypotheses in phrase-based translation. One of the critical features in this system is the language model. In experiments to follow, we only study this main language model feature, however, our decoder has other features implemented and arbitrary features can be added easily to the system, demonstrating the system flexibility.

Our strategy achieves state-of-the-art performance as measured by BLEU (Section 2.8.1) in the string regeneration task. We also study various capabilities of our decoder: in Section 2.5.4, we show how to split the input into chunks for more rapid experimentation; in Section 6.6.4, we analyse the effect of stack-based pruning; in Section 6.6.5, we study the effect of including n-gram rules with a larger n-gram order; in Section 6.6.6, we relax decoding constraints by allowing n-gram rules to overlap; in Section 6.6.7, we introduce future cost estimation using a unigram language model; finally, in Section 6.6.8, we demonstrate how SMT rescoring techniques are also applicable to the output of our decoder.

6.2 Background

Brown et al. (1990) anecdotally mention the string regeneration task, or *bag of words translation*, in order to demonstrate the effectiveness of n-gram language models. With a 3-gram language model, they recover 63% of a sentences with less than 11 words with brute force search over all permutations. This is of course impractical for longer sentences.

String regeneration has recently gained interest as a standalone task.Wan et al. (2009) introduced the string regeneration task as a proxy to measure the grammaticality of a natural language generation system output. String regeneration can also be considered as a simplified version of the *linguistic realisation* task in a natural language generation system (Reiter and Dale, 1997), which consists in generating a surface form that is syntactically, morphologically, and orthographically correct.

Wan et al. (2009) use a modified version of the minimum spanning tree algorithm in order to find an optimal dependency tree that covers an input bag of words. Once a dependency tree is built, an n-gram language model is used so as to order siblings in the tree. Performance is measured by the BLEU metric. Wan et al. (2009) demonstrate BLEU improvements when regenerating Section 23 of the Penn Treebank using their original algorithm vs. baseline algorithms based on n-gram language models only. Base nouns indicated by the Penn Treebank are kept together for experimentation.

He et al. (2009) also use the dependency tree formalism for string realisation. They describe a sentence realiser that takes as input an unordered dependency tree, i.e. a dependency tree that only encode head-modifier relations but not the surface string ordering. Because an exhaustive search is intractable (due to the number of possible children permutations at each node), He et al. (2009) adopt a divide-and-conquer approach that recursively linearises subtrees in a bottom-up fashion. A log-linear model is used to select the best possible linearisation for each subtree. Features include dependency relations, n-gram language models for the surface string and for the sequence of heads.

CCG grammars (Zhang and Clark, 2011) have also been used for the string regeneration task as an alternative to the dependency grammar formalism. Given an input bag of words, the best CCG parse that covers the input is searched for. Similarly to previous work, input base nouns are kept as single units. In follow-up work (Zhang et al., 2012), CCG grammars are combined with large language models and provide performance improvements as measured by the BLEU metric.

Some of the works described so far are instances of *tree linearisation*: given a set of unordered dependencies, reconstruct a dependency tree which gives the word ordering (Belz et al., 2012). Zhang (2013) extends this task to partial tree linearisation: the input is a set of possibly missing POS tags and possibly missing unordered dependencies. This last work achieves state-of-the art performance on string regeneration on the Penn Treebank, when using dependency information on the input. Zhang (2013) also reports results on string regeneration without using dependency information on the input but keeping base noun phrases as single units.

In this chapter, we restrict the input to be a bag of words without any additional syntactic or semantic information. Our work is directly inspired by recent work that exploits phrase-based grammars together with a hierarchical phrase-based decoder based on FSTs (see Section 2.9) (de Gispert et al., 2014). We also use phrase-based grammars, but our decoding procedure is analogous to stack-based decoding for phrase-based translation. To our knowledge, our method achieves state-of-the-art performance on this task. We will now present our model, which is analogous to the phrase-based translation model, but in a monolingual setting.

6.3 Phrase-Based Translation Model for String Regeneration

In the previous section, we have reviewed various strategies for string regeneration. We observe that, in general, the use of syntax was beneficial over the exclusive use of an n-gram language model. We do not take advantage of syntactic information, instead, we complement an n-gram language with n-gram rules extracted from monolingual data. We now describe our method, which is inspired from phrase-based translation techniques (see Section 2.5). We employ a feature based linear model (see Section 2.4) as our objective function:

$$\hat{\boldsymbol{e}} = \operatorname*{argmax}_{\boldsymbol{e}} \boldsymbol{\lambda} \cdot \boldsymbol{h}(\boldsymbol{e}) \tag{6.1}$$

where:

- $\hat{\boldsymbol{e}}$ is the best possible hypothesis according to the model.
- $\boldsymbol{e}$ is an English sentence.
- $\boldsymbol{\lambda}$ is a feature weight vector.
- $\boldsymbol{h}(\boldsymbol{e})$ is a feature vector for the hypothesis $\boldsymbol{e}$.

$\boldsymbol{\Phi}(\boldsymbol{e})$ contains a language model $g(\boldsymbol{e})$ as well as local features $\boldsymbol{\varphi}(\boldsymbol{e})$ that are additive over phrases that segment $\boldsymbol{e}$. Similarly, $\boldsymbol{\lambda}$ has a weight λ_g for the language model and weights $\boldsymbol{\lambda}_{\boldsymbol{\varphi}}$ for local features. As in phrase-based translation, a hypothesis is decomposed into phrases $\boldsymbol{e}_1^I$. We can then rewrite Equation 6.1 into Equation 6.2:

$$\begin{aligned} \hat{\boldsymbol{e}} &= \operatorname*{argmax}_{\boldsymbol{e}=\boldsymbol{e}_1^I} \lambda_g g(\boldsymbol{e}) + \boldsymbol{\lambda}_{\boldsymbol{\varphi}} \cdot \boldsymbol{\varphi}(\boldsymbol{e}) \\ &= \operatorname*{argmax}_{\boldsymbol{e}=\boldsymbol{e}_1^I} \lambda_g g(\boldsymbol{e}) + \boldsymbol{\lambda}_{\boldsymbol{\varphi}} \cdot \sum_{i=1}^{I} \boldsymbol{\varphi}(\boldsymbol{e}_i) \end{aligned} \tag{6.2}$$

Note that this model makes a max approximation: various segmentations of the hypotheses are considered but the score only depends on one single segmentation. Experiments to follow only make use of the language model feature but our implementation has other features integrated, such as word count or rule count, and arbitrary features can be added easily. Our decoder algorithm described subsequently in Section 6.5 generates hypotheses left to right by concatenating phrases $\boldsymbol{e_1}$, ..., $\boldsymbol{e_I}$. For a phrase index $i \in [1, I]$, we define the partial model score $\mathrm{sc}(\boldsymbol{e_1^i})$ of partial hypothesis $\boldsymbol{e_1^i}$ as in Equation 6.3:

$$\mathrm{sc}(\boldsymbol{e_1^i}) = \lambda_g g(\boldsymbol{e_1^i}) + \boldsymbol{\lambda_\varphi} \cdot \sum_{k=1}^{i} \boldsymbol{\varphi(e_k)} \tag{6.3}$$

6.4 Phrase-Based Translation Rules for String Regeneration

In the previous section, we have introduced our model, inspired from phrase-based translation, for string regeneration. This model assumes that a hypothesis is segmented into phrases. In this section, we describe how to obtain these phrases from monolingual data. In Section 6.4.1, we review the n-gram rule extraction process, which is based on the Hadoop implementation of MapReduce. In Section 6.4.2, we describe how to obtain relevant rules for a given test set.

6.4.1 Rule Extraction

n-gram rules are extracted from large collections of monolingual text. The text is tokenised and lowercased. We then employ the MapReduce framework as implemented by Hadoop in order to extract n-grams together with their occurrence counts from the corpus.

The input key to the *map* function is irrelevant. The input value to the *map* function is a line of text. The *map* function simply extracts all n-grams from the input line of text and outputs these n-grams with a count of one. The input to the *reduce* function is an n-gram and a series of constant values equal to one. The *reduce* function simply sums the counts and produces the n-gram with its total count. This MapReduce job is analogous to the canonical "word count" example except that here n-grams of higher order than unigrams are considered. For our experiments, we extract n-grams from order 1 to 5.

6.4.2 Rule Filtering

Once we have extracted all n-grams from a monolingual corpus, we would like to retrieve the n-grams relevant to a test set that we wish to experiment on. An n-gram is relevant if the vocabulary of the n-gram is included in any of the vocabularies of each sentence in the test set.

Contrary to the process described in Section 3.5, we do not generate queries and then seek those queries in an HFile. This is because the number of possible queries is too large to fit in memory. For a sentence of length N with words distinct from each other, the number of possible relevant 5-grams is $N \times (N-1) \times (N-2) \times (N-3) \times (N-4)$. For sentence of length 100, this would correspond to approximately 9B queries. Even with an alternative implementation where keys in the HFile are lexicographically sorted n-grams and values are the permutations and counts found in the monolingual data, the number of 5-gram queries would be $\binom{N}{5}$, which is approximately 75M queries for a 100 word sentence.

We therefore take the alternative approach of scanning the set of n-grams and retaining the relevant ones. For this, we again use a MapReduce job. The input key to the *map* function is an n-gram and the input value is a count. For each test sentence, the *map* function checks if the n-gram vocabulary is included in the vocabulary of that sentence and if so, it generates all possible coverages of this n-gram for that sentence, where a coverage indicates the positions of the words covered by the n-gram in the input sentence. The *map* output key is the n-gram together with a sentence id. The *map* output value is a coverage and a count. The *reduce* function simply removes duplicates in the list of coverages.

6.5 String Regeneration Search

In Section 6.3, we have described our string regeneration model and in Section 6.4, we have described how to extract n-gram rules. We now present our search algorithm and introduce our string regeneration decoder, *NgramGen*, which reorders an input bag of word using the model introduced and stack-based decoding techniques reviewed in Section 2.5.4.

6.5.1 Input

The input to the decoder consists of:

- A bag of words, or multiset, $\boldsymbol{w}$, represented as a vector. For example, we have $\boldsymbol{w} =$ [do, you, do] for the input sentence "do you do" to be regenerated.

- A set of n-grams, where the vocabulary of each n-gram is a subset of the vocabulary of the input bag of words. Each n-gram is also associated with one or more coverages as computed in Section 6.4.2. For a given n-gram, its coverage is a bit vector that indicates what positions the n-gram covers in the input bag of words. In case of repeated words, an n-gram can have several possible coverages. For example, with the input bag of words [do, you, do], the n-gram [do, you] has two possible coverages: 110 and 011. The input n-grams and coverages are represented as an associative array datastructure. We denote this input by $\mathcal{N}$. Note that it would be possible for the decoder to compute possible coverages on the fly, however we choose to precompute those at the filtering step described in Section 6.4.2.

- A n-gram language model $\mathcal{G}$. The n-gram language model is used as a the main feature in the model to encourage fluent output.

We will now describe how hypotheses, i.e. permutations of the input, are generated by the decoder from a bag of words.

6.5.2 Algorithm

Our algorithm is inspired by stack based decoding for phrase-based translation, reviewed in Section 2.5.4. We first give an overview before outlining the algorithm details.

6.5.2.1 Overview

In order to represent partial hypotheses, we use a vector of *columns* (more commonly called stacks). Each column contains a set of *states*. Each state represents a set of partial hypotheses. All states in a column represent hypotheses that cover the same number of words in the input.

We start to build hypotheses from an initial state that represents the empty hypothesis. This initial state is located in the first column, which represents the hypotheses that have not covered any word in the input. The empty hypothesis is then extended with the input n-gram rules and their coverage. New states are created to take into account these extensions. Then, partial hypotheses represented by these states are repeatedly extended until hypotheses that cover the entire input are produced.

6.5.2.2 Algorithm Details

As mentioned in Section 6.5.2.1, we represent the set of partial hypotheses with a vector of columns that we denote M. For an input $\boldsymbol{w}$ of size $|\boldsymbol{w}|$, M has a size of $|\boldsymbol{w}| + 1$. Elements of M are denoted M_0, ..., $M_{|\boldsymbol{w}|}$. The i-th column M_i represents hypotheses that have covered i words in the input. The columns M_i are the analog of stacks in stack based decoding for phrase-based translation.

Each column contains a set of states that represents a set of partial hypothesis. A state contains the information necessary to extend a partial hypothesis and represents a set of partial hypotheses that can be *recombined* (see Section 2.5.4). This means that if two partial hypotheses h_1 and h_2 are represented by a state s and h_1 has a higher model score than h_2, then if h_1 and h_2 are extended with the same rules, then the extension of h_1 will have a higher score than the extension h_2. In our case, two hypotheses can be recombined if they share the same last $n - 1$ words where n is the order of the n-gram language model.

The states are sorted by their *score*. For a given state, its score is the best model score of any of the partial hypotheses that the state represents. The decoding algorithm is presented in Figure 6.2. We first initialise the matrix M to be a vector of empty columns of size $|\boldsymbol{w}| + 1$ (line 2). The first column is filled with an initial state representing an empty hypothesis. Then, we loop over the column indices (line 3), the states in each column (line 4) and the n-gram rules (line 5). In each case, we attempt to extend a state with an n-gram rule. We first test if the n-gram rule r is applicable to state s (line 6). If this is the case, we extend the state s with the rule r (line 7). We will now describe what kind of information is contained in a state. This will allow use to also describe the last two operations, CanApply and Extend.

```
 1: function DECODE($\boldsymbol{w}$, $\mathcal{N}$, $\mathcal{G}$)
 2:     INITIALIZE($M$)
 3:     for $0 \leq i < |\boldsymbol{w}|$ do
 4:         for state $s \in M_i$ do
 5:             for ngram $r \in \mathcal{N}$ do
 6:                 if CANAPPLY($s, r$) then
 7:                     EXTEND($s, r$)
 8:                 end if
 9:             end for
10:         end for
11:     end for
12: end function
```

Figure 6.2: NgramGen decoding algorithm. The input is a bag of words $\boldsymbol{w}$, a set $\mathcal{N}$ of n-gram rules with their coverage and a language model $\mathcal{G}$. The column vector M is first initialised with a state representing the empty hypothesis. Then, the initial state is iteratively extended.

State Definition The states must contain enough information in order to be able to extend a hypothesis, and states also represent all hypotheses that can be recombined. Thus, a state contains the following information:

- Coverage: the coverage is a bit vector that indicates which words in the input have been covered. This implies a sorting of the input bag of words. The sorting is arbitrary and we simply choose to represent the bag of words by the sequence of words to be recovered.

- History: in order to compute the n-gram language model score correctly when extending a partial hypothesis, we store the last $n-1$ words of the partial hypothesis we want to extend. The definition of history is therefore the same as the definition of history for an n-gram language model. In our implementation, the history is simply encoded as an `lm::ngram::State` as defined in the KenLM toolkit (Heafield, 2011).

CANAPPLY Operation Given a state s with coverage $c(s)$ and an n-gram rule r with coverage $c(r)$, rule r can be applied to state s if $c(s)$ and $c(r)$ are disjoint. We will see in Section 6.6.6 that this constrain can be relaxed if we allow for overlapping n-grams.

Extend Operation Given a state s with coverage $c(s)$ and and n-gram rule r with coverage $c(r)$ that can be applied to s, we extend s with r into a new state s' as follows:

- The coverage $c(s')$ of s' is the bitwise OR of $c(s)$ and $c(r)$:

$$c(s') = c(s) \mid c(r) \tag{6.4}$$

- The new partial score $\text{sc}(s')$ for s' is defined in terms of the partial score $\text{sc}(s)$ for s, the history $h(s)$ and the rule r:

$$\text{sc}(s') = \text{sc}(s) + \lambda_g g(r|h(s)) + \boldsymbol{\lambda_\varphi} \cdot \boldsymbol{\varphi(r)} \tag{6.5}$$

- If s' already exists in the column corresponding to its coverage, then the score of the existing state is updated if $\text{sc}(s')$ is better than the existing score. Otherwise, s' is simply added as a new state to the column corresponding to its coverage.

Pruning The search space for an input bag of size N is the number of permutations of the input, which is $N!$ if all input words are distinct. We therefore need to apply pruning during search to make the decoding process tractable. We support histogram pruning and threshold pruning (see Section 2.5.4). After each extension, if we add a new state, we enforce that the column where a state was added satisfies the pruning constraints. This means that for histogram pruning, with m the maximum number of states per column, if we add a new state to a column that has m states, then the state with the lowest score is removed from that column.

For threshold pruning with a threshold t, the situation is slightly more complex. When we want to add a state s with score $\text{sc}(s)$ to a column where the best score is b, we consider three cases:

- $\text{sc}(s) \leq b$ and $\text{sc}(s) \geq b - t$: the state s is added to the column.
- $\text{sc}(s) \leq b$ and $\text{sc}(s) < b - t$: the state s is not added to the column.
- $\text{sc}(s) > b$: the state s is added to the column and all states in that column with a score less than $\text{sc}(s) - t$ are removed from the column.

We have presented the decoding algorithm and various data structures to support the algorithm. We will now describe how the set of hypotheses is encoded with FSTs.

6.5.3 FST Building

We mentioned in Section 6.5.2.2 that a state represents partial hypotheses that can be recombined as well as the necessary information to be able to extend a partial hypothesis. However, since we do not keep backpointers between states, we cannot recover hypotheses directly from M, the vector of columns. Instead, we build an FST that contains all hypotheses. For each extension, an arc is added to the output FST. We make use of the OpenFst toolkit (Allauzen et al., 2007). This allows us to rescore the output using tools already integrated with OpenFst (Blackwood, 2010). Another advantage is that hypotheses that are recombined are not deleted and all completed hypotheses are stored in the output FST. This allows hypotheses that would be discarded in recombination to remain in the output and get an opportunity to become the best hypothesis after rescoring.

6.5.4 Example

We now demonstrate how our algorithm works with an example. Our input consists of:

- a bag of words: [do, you, do]. Note that the word *do* is repeated.
- a bigram language model. Thus the history for each state will consist of the last word of the partial hypothesis.
- monolingual n-gram rules with their coverage of the input bag of words. We use the rules listed in Table 6.1.

After running our algorithm without pruning, we obtain the data structure represented in Figure 6.3. We can see that the same state $\{111, \text{do}\}$ is reused for the two hypotheses *do you do*, *you do do*, because these hypotheses have the same coverage and the same history. We also note that the hypothesis *do you do* is repeated. This is not an issue as we use an operation of determinisation on the resulting FST.

We will now present various experiments that demonstrate the effectiveness of our decoder for the string regeneration task. We will also analyse various capabilities of the decoder in order to understand which configurations lead to better results.

N-gram	Coverage
do	100
do	001
you do	011
you do	110
do you	110
do you	011

Table 6.1: Example of input monolingual n-gram rules with their coverage, corresponding to the example input [do, you, do]. The FST representing the set of hypotheses obtained with these rules is represented in Figure 6.3.

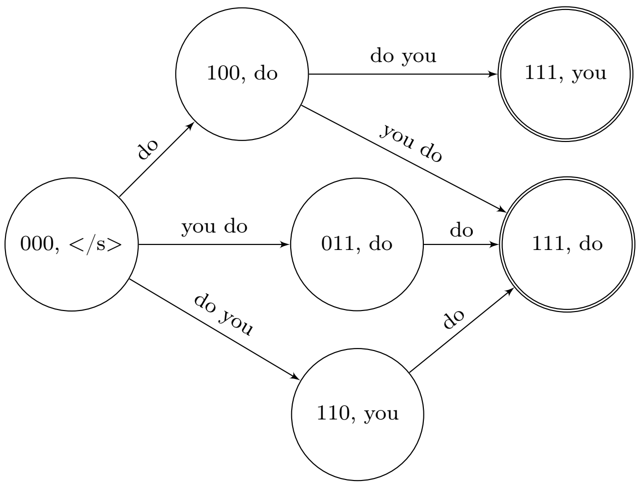

Figure 6.3: Example of output obtained from running the NgramGen decoder on the input [do, you, do] with the rules listed in Table 6.1. Double circled accepting states represent completed hypotheses.

6.6 Experiments

In this section, we will present various string regeneration experiments. We first present a baseline without restriction on the input length. Then, we show how the input can be split into chunks of maximum length for more rapid experimentation. We then study the effect of pruning, n-gram rule length, the use of overlap between rules, the benefits of future cost estimation and finally the benefits of applying rescoring to our first pass decoder.

6.6.1 Experimental Setup

We run string regeneration experiments on various data sets. We use the following data sets:

- MT08-nw: the first English reference for the newswire portion of the Arabic-English translation task for the NIST Open Machine Translation 2008 Evaluation.[2]
- MT09-nw: the first English reference for the newswire portion of the Arabic-English translation task for the NIST Open Machine Translation 2009 Evaluation.[3].
- WSJ22: Section 22 from the Penn Treebank.
- WSJ23: Section 23 from the Penn Treebank.

The test sets are first tokenised and lowercased. Then, n-gram rules up to length 5 for each of the processed test sets are extracted from a large monolingual text collection. For MT08-nw and MT09-nw, we use all available monolingual data for the NIST Open Machine Translation 2012 Evaluation.[4] This consists of approximately 10.6 billion words. For WSJ22 and WSJ23, we use all available monolingual data for the WMT13 evaluation, described in Table 5.2. This consists of approximately 7.8 billion words. For MT08-nw and MT09-nw, we estimate a modified Kneser-Ney 4-gram language model on approximately 1.3 billion words of English text, including the AFP and Xinhua portions of the Gigaword corpus version 4 and the English side of

[2] `http://www.itl.nist.gov/iad/mig/tests/mt/2008`
[3] `http://www.itl.nist.gov/iad/mig/tests/mt/2009`
[4] `http://www.nist.gov/itl/iad/mig/openmt12.cfm`

Test Set	BLEU
MT08-nw	42.32
MT09-nw	37.29
WSJ22	44.01
WSJ23	45.48

Table 6.2: Baseline for the NgramGen Decoder on the various test sets used throughout experimentation. No limits are imposed on the input length, thus pruning setting are set so that decoding is feasible on the given hardware. Performance is measured with case insensitive BLEU.

various Arabic-English parallel corpora used in MT evaluations. For WSJ22 and WSJ23, we estimate a modified Kneser-Ney 4-gram language model on all available monolingual data for the WMT13 evaluation.

6.6.2 Baseline

We first run a baseline on our various test sets with the NgramGen decoder. Because there are no constraints on the input length, some long sentences need more pruning for the decoder to complete the procedure. Therefore, we use a length dependent histogram pruning of $\frac{11000}{\text{length(input)}}$. For example, for an input sentence of length 100, this means that we use a histogram pruning of 110 for that particular sentence. Results are reported in Table 6.2.

We also compare the performance obtained by the NgramGen decoder with previous work on the data set WSJ23 in Table 6.3. We can see that our method, without using any additional information apart from the input bag-of-word, achieves state-of-the-art performance. All other methods keep base noun phrases as a single unit and also use additional syntactic information such as part-of-speech tags.

In the section to follow, we impose length restrictions on the input bag of words in order to allow more rapid experimentation.

6.6.3 Sentence Splitting

Running the NgramGen decoder without limiting the length of the input as in Section 6.6.2 can be time and memory consuming. For more rapid experimentation, we only reorder chunks of a limited size in the input. The input

Work	BLEU	Additional Input Information
(Wan et al., 2009)	33.7	POS tag + base noun phrases
(Zhang and Clark, 2011)	40.1	POS tag + base noun phrases
(Zhang et al., 2012)	43.8	POS tag + base noun phrases
(Zhang, 2013)	44.7	base noun phrases
This work	**45.5**	**none**

Table 6.3: Comparing the NgramGen decoder performance with previous work on the WSJ23 data set. Case-insensitive BLEU scores are reported. The NgramGen decoder achieves state-of-the-art performance, without using additional information to the input bag of words.

is divided each time a punctuation sign such as a comma or a semi-colon is observed or after having seen a maximum number of tokens. The implementation simply considers only n-grams whose coverage falls completely into the chunk being reordered; thus the language model is computed correctly at the boundaries between chunks. We also use a histogram pruning of 1000 states regardless of the input length.

With this restriction, we obtain the results reported in Table 6.4. Logically, performance increases as the maximum size of chunks decreases. For a maximum chunk length of 1, the BLEU score would be 100. We will use this baseline for more rapid experimentation and further analysis. Specifically, in experiments to follow, we will conduct experiments with a relatively short maximum chunk length of 11 and a more realistic maximum chunk length of 20.

It should be emphasised that the purpose of splitting the input bag of words by using information from the sentence to be regenerated is only to be able to investigate more quickly how to better use the regeneration decoder. This method should not be used for evaluating an end-to-end system on the string regeneration task. However, we will see in Chapter 7 that using this method on the output of a translation system is justified since no information from the reference translation is used.

6.6.4 Pruning

In Section 6.6.3, we have shown how to split the input into chunks to allow more rapid experimentation. Using maximum chunk lengths of 11 and 20,

Maximum Chunk Length	MT08-nw
7	82.83
9	78.53
11	74.86
13	71.75
15	68.95
20	64.36

Table 6.4: NgramGen decoder baseline with restrictions on the input length. Case-insensitive BLEU scores are reported. For further experiments, maximum chunk lengths of 11 and 20 will be used to contrast results on shorter or longer inputs. The BLEU score for a maximum chunk length of 1 would evidently be 100.

we will study the effect of various histogram pruning thresholds.

Results are reported in Table 6.5. In both maximum input length settings, performance increases as the maximum number of states per column increases. For a maximum input length of 11, performance improves by 0.39 BLEU (0.5% relative, row 1 vs. row 7). For a maximum input length of 20, performance improves by 1.18 BLEU (1.8% relative, row 8 vs. row 14). This is due to a larger search space being explored in the case of a maximum input length of 20.

6.6.5 n-gram Order for n-gram Rules

As mentioned in Section 6.2, previous work complements the use of n-gram language models with syntactic information for the string regeneration task or for the tree linearisation task. In this work, n-gram language models are complemented by the use of n-gram rules extracted from large amounts of monolingual text. In this section, we observe the impact of using all rules from unigrams up to 5-grams or only using rules with specific lengths. Specifically, we compare the following configurations:

- All n-gram orders: this is the default used in experiments so far.
- Unigrams, 4-grams and 5-grams only: our motivation for this configuration is to only use rules that are likely to produce a fluent output,

Row	Histogram Pruning	Max Input Length	MT08-nw
1	500	11	74.67
2	1000	11	74.86
3	1500	11	74.98
4	2000	11	75.02
5	3000	11	75.09
6	4000	11	75.08
7	5000	11	75.06
8	500	20	64.21
9	1000	20	64.36
10	1500	20	64.85
11	2000	20	64.94
12	3000	20	65.17
13	4000	20	65.33
14	5000	20	65.39

Table 6.5: Effect of histogram pruning on performance with a maximum input length of 11 and 20. Case-insensitive BLEU scores are reported. In both maximum input length settings, increasing the maximum column size increases performance, but more so with maximum input length of 20 since the search space is much larger in that case.

namely 4-grams and 5-grams. We always use unigrams in order to obtain complete hypotheses. For example, if we only used 4-grams and 5-grams, an input bag of words of size 6 could never be covered, unless overlapping rules were allowed (see Section 6.6.6).

- Unigrams and 5-grams only: our motivation is identical to using only unigrams, 4-grams and 5-grams.

- Unigrams only: in this setting, only the language model informs the decoder. This setting is similar to the one mentioned in Section 6.2 where all permutations are considered and ranked by the language model. Our motivation for this setting is to investigate whether n-gram rules are at all useful for string regeneration.

Results are reported in Table 6.6. We can see that for a relatively short maximum of 11 on the input length, using more n-gram orders is always beneficial. Because the maximum input length is short, the search space can be explored more thoroughly and this benefits configurations that generate larger search spaces. For a more realistic maximum input length of 20, results are more in line with our initial motivation for each configuration. Comparing row 6 vs. row 5, we can observe that removing lower order n-grams is slightly beneficial. The same trend is observed when comparing row 7 vs. row 5. Finally, for both maximum input length configurations of 11 and 20, it is always beneficial to use n-gram rules with $n > 1$.

6.6.6 Overlapping n-gram Rules

In Section 6.5.2.2, we defined the CanApply operation by requiring that the state coverage and the coverage of the rule used to extend the state be disjoint. In this section, we relax this constraint to allow for a coverage intersection of a certain maximum overlapping length. The idea of allowing overlapping rules has been used previously in phrase-based translation for creating phrase tables with longer rules (Tribble et al., 2003) but not in decoding.

Results are presented in Table 6.7. Configurations for overlapping rules include no overlap, a maximum overlap of 1 or a maximum overlap of 2. As in previous experiments, we contrast the effect of overlapping rules with a maximum input length of either 11 or 20. Decoding is run in parallel using Sun Grid Engine (Gentzsch, 2001). The total decoding time is measured as

Row	Rule Configuration	Max Input Length	MT08-nw
1	all rules	11	74.86
2	1g/4g/5g	11	74.85
3	1g/5g	11	74.82
4	1g only	11	74.13
5	all rules	20	64.36
6	1g/4g/5g	20	64.42
7	1g/5g	20	64.74
8	1g only	20	63.36

Table 6.6: Effect of selecting n-gram rules based on their n-gram order. Case-insensitive BLEU scores are reported. For a short maximum input length, using more orders is always beneficial because this increases a tractable search space. For a maximum input length of 20, retaining only n-gram rules with higher order n-gram produces better results.

the sum of time spent on each job. We also study the effect of overlapping rules when all n-gram orders are included (rule configuration "all") or when only unigrams and 5-grams are allowed (rule configuration "1g/5g"). We hypothesise that overlapping rules may be more beneficial when lower n-gram orders are missing. However, we observe that that allowing for overlapping rules does not improve performance, apart from increasing the search space and decoding time in general. We do not use this decoder capability in future experiments.

6.6.7 Future Cost

The justification for adding a future cost estimate to the partial hypothesis score is analogous to the one provided for phrase-based translation in Section 2.5.4. Hypotheses that cover the same number of words in the input bag of words are grouped together in a stack and hypotheses with costs that are too high are pruned out. However, these hypotheses may not be directly comparable, because some hypotheses cover very frequent words that are favoured by the language model while other hypotheses may cover infrequent words. We therefore estimate a future cost for each hypothesis. This future cost is added to the partial hypothesis cost for pruning purposes.

We investigate the effect of using a unigram language model to estimate

Row	Overlap	Max Input Length	Rule Configuration	MT08-nw	Total Time (s)
1	no overlap	11	all	74.86	20075
2	1	11	all	74.87	19685
3	2	11	all	74.85	18200
4	no overlap	11	1g/5g	74.82	6774
5	1	11	1g/5g	74.81	8626
6	2	11	1g/5g	74.80	8805
7	no overlap	20	all	64.36	57285
8	1	20	all	64.39	89989
9	2	20	all	64.28	95775
10	no overlap	20	1g/5g	64.74	13420
11	1	20	1g/5g	64.72	20314
12	2	20	1g/5g	64.63	22522

Table 6.7: Effect of allowing for overlapping rules. Case-insensitive BLEU scores are reported. In all configurations, allowing for overlapping rules does not improve performance. In addition, overlapping rules increase decoding time in general.

Configuration	Max Input Length	MT08-nw
no future cost estimate	11	74.86
future cost estimate	11	75.10
no future cost estimate	20	64.36
future cost estimate	20	65.05

Table 6.8: Effect of using a future cost estimate for pruning. Case-insensitive BLEU scores are reported. Future cost is computed with a unigram language model. Future cost is beneficial for both a relatively short maximum input length of 11 and for a more realistic maximum input length of 20, with respective improvements of 0.24 BLEU and 0.69 BLEU.

the future cost. Given a partial hypothesis, its future cost is defined by the unigram language model score of the words from the input bag of words that are not covered by the partial hypothesis. Because there is no notion of word ordering in the input, we cannot easily use a higher order n-gram language model language model to estimate future cost.

We compare the use of future cost estimates with maximum input lengths of 11 and 20, as in previous experimentation. Results are reported in Table 6.8. In both settings, adding a unigram language model as future cost estimate to the cost of a partial hypothesis for pruning is beneficial. For a short maximum length of 11 on the input length, we obtain a gain of 0.24 BLEU while for a maximum input length of 20, we obtain a gain of 0.69 BLEU.

6.6.8 Rescoring

In Section 6.5.3, we described how the NgramGen decoder generates an FST that encodes a set of hypotheses. We choose this output format in order to be able to apply various lattice rescoring techniques that also utilise this format as input (see Section 2.10.1, Section 2.10.2 and Section 2.10.3).

In this section, we show that these rescoring techniques are not only effective on the output of a first pass translation decoder but also in our string regeneration setting. Specifically, we run 5-gram language model lattice rescoring as well as LMBR lattice rescoring experiments. LMBR applied to hypothesis combination, described in Section 2.10.2, will be applied in Chapter 7.

Configuration	MT08-nw	MT09-nw	WSJ22	WSJ23
first pass	64.36	60.91	62.76	64.39
+5g	66.11	63.26	63.18	65.04
+LMBR	67.37	64.53	64.28	65.85

Table 6.9: Effect of lattice rescoring on the first pass lattices obtained by the NgramGen decoder. Case-insensitive BLEU scores are reported. Rows "+5g" and "+LMBR" show large BLEU gains with respect to first pass decoding, demonstrating the effectiveness of these lattice rescoring techniques across various tasks, including translation and string regeneration.

Results are reported in Table 6.9, on all our test sets. The MT08-nw set is used to tune LMBR parameters, and these parameters are tested on the MT09-nw set. Similarly, WSJ22 is used as a tuning set for WSJ23. Because we only use a single language model as the only feature and because in all decoder hypotheses, the length is identical, there is no tuning or testing set for first-pass decoding or 5-gram language mode rescoring.

We can see that large BLEU gains are obtained in 5-gram rescoring: +1.29 BLEU on average across all test sets. LMBR rescoring also provides an average gain of 2.54 BLEU with respect to first pass decoding over the test sets MT09-nw and WSJ23.

6.7 Conclusion

In this chapter, we have presented a novel approach to the string regeneration task, inspired from phrase-based models for SMT. We have implemented a decoder, NgramGen, that operates in a similar fashion to phrase-based stack-based decoders and achieves state-of-the-art performance on the string regeneration task from a simple bag of words on the Penn Treebank Section 23 data set.

We have also analysed various capabilities of our decoder. From this analysis, we draw the following conclusions:

- Removing lower order n-gram rules other than unigrams, which are needed to ensure that the input can be covered entirely, improves performance.

- Allowing for overlapping rules slows down the decoding procedure and does not provide gains in our experiments.
- Future cost estimates improve the pruning procedure and overall performance.
- Lattice rescoring techniques are applicable and provide gains in the string regeneration setting.

In the following chapter, we will explore potential application of our decoder to word reordering in machine translation. We will consider the output of a machine translation decoder as the input bag of words for the regeneration decoder.

Chapter 7

String Regeneration Applied to Machine Translation

In Chapter 6, we have introduced techniques for string regeneration inspired from the phrase-based translation paradigm. We have demonstrated their effectiveness and achieved state-of-the-art performance in terms of BLEU on the string regeneration task on the Penn Treebank. In this chapter, we will attempt to apply these techniques to the output of a machine translation system. One key difference is that the quality of the input bag of words is substantially lower when regenerating translation output instead of regenerating an original English sentence.

Our motivation is the general motivation for the rescoring paradigm. Instead of allowing any kind of reordering in first pass decoding, which would make the search space intractable, we first create a search space of relatively high quality, then we relax reordering constraints. We will show that using our string regeneration decoder, we are able to match the quality of our translation system. In some instances, we show very slight gains in performance. This work is meant to lay the ground work for future incorporation of natural language generation into SMT.

In Section 7.1, we describe our machine translation baseline system. In Section 7.2, we describe our string regeneration baseline, both with and without future cost estimates. Because the regeneration baseline degrades performance with respect to the translation baseline, we carry out an error analysis in Section 7.3. In Section 7.4 and in Section 7.5, we show two ways of incorporating information from the translation system in order to match the translation system quality; in some instances, we obtain very slight improvements over the MT baseline.

7.1 SMT Baseline System

In this section, we briefly summarise the machine translation system used as a baseline for our string regeneration rescoring experiments. The Cambridge University Engineering Department participated in the NIST 2012 translation competition.[1] We describe here the Chinese-English system. We pick this language pair rather than for example Arabic-English simply because Chinese and English are quite distant in terms of word ordering (see Section 2.6.1).

We use all the Chinese-English parallel data available for the constrained track. Parallel data consists of 9.2M sentence pairs, 203M tokens on the Chinese side and 218M tokens on the English side, after preprocessing. Parallel data is aligned using a word-to-phrase HMM model with the MTTK toolkit (see Section 2.3.1). In the source-to-target direction, the maximum size of a Chinese phrase is 2 while in the target-to-source direction, the maximum size of an English phrase is 4. The final alignments are obtained by taking the union of Viterbi alignments obtained in both source-to-target and target-to-source directions. A hierarchical phrase-based grammar is then extracted from the alignments using the infrastructure described in Section 3.4.

We build a first-pass language model using the target side of the parallel text and the AFP and Xinhua agencies of the Gigaword corpus (Parker et al., 2009). We first build separate modified Kneser-Ney smoothed 4-gram language models (see Section 2.7.3) on each of these three corpora. The language models built on AFP and Xinhua are interpolated with equal weights. The resulting language model is interpolated with the language model built on the target side of the parallel text with equal weights. We also build a second-pass Stupid Backoff 5-gram language model for rescoring as described in Section 2.10. The output of 5-gram rescoring is rescored with LMBR (see Section 2.10.2).

We use the features described in Section 2.6.5 as well as 36 provenance related features (see Section 5.5). The union strategy is employed for test set grammar filtering, as described in Section 5.5. In this chapter, all results are reported on an internal tuning set (Tune) in the newswire domain com-

[1] `http://www.nist.gov/itl/iad/mig/openmt12.cfm`

Configuration	Tune	MT08
MT baseline 1st pass	34.96	35.71
MT baseline +5g	36.04	36.60
MT baseline +lmbr	36.80	37.59

Table 7.1: CUED NIST 2012 system performance as measured by case insensitive BLEU on MT08. The output of the first pass decoder will serve as input to the regeneration decoder.

prising 1755 sentences and on the newswire portion of the NIST 2008[2] test set comprising 691 sentences (MT08).

The results are reported in Table 7.1. For information, the system submitted for the NIST12 evaluation was based on hypothesis combination over various preprocessing of the source and obtained a score of 33.90 BLEU on the newswire portion of the MT12 test set.

In the next section, we will describe our string regeneration setting where the output of the translation system is taken as input by our regeneration decoder.

7.2 Regeneration Decoder Baseline on MT Output

In this section, we describe how the first-pass output of the SMT system described in Section 7.1 is processed as input by the NgramGen regeneration decoder. We consider a test set with N source sentences $\boldsymbol{f_1}$, ..., $\boldsymbol{f_N}$ to be translated. The first-pass translation decoder generates N sets of hypotheses encoded as N lattices $\mathcal{H}_1$, ..., $\mathcal{H}_N$. We extract 10-best lists from these lattices. The i-th-best translations are denoted $\boldsymbol{e}_1^i$, ..., $\boldsymbol{e}_N^i$. Thus we obtain 10 sets of bags-of-words. The first set of bags-of-words consists of the best first-pass translations for each source sentence, that is $\boldsymbol{e}_1^1$, ..., $\boldsymbol{e}_N^1$, and so on for the second set up to the tenth set of bags, which contains $\boldsymbol{e}_1^{10}$, ..., $\boldsymbol{e}_N^{10}$. Clearly, the quality of bags-of-words decreases as we go from the first set of bags to the tenth set.

We then run our regeneration decoder NgramGen on each set of bags-of-

[2] `http://www.itl.nist.gov/iad/mig/tests/mt/2008/`

words separately. The configuration for running the decoder on the i-th set of bags is called ith-best. We obtain N lattices $\mathcal{L}_1^1$, ..., $\mathcal{L}_N^1$ for the first set of bags, and so on for the second set of bags to the tenth set of bags, for which we obtain N lattices $\mathcal{L}_1^{10}$, ..., $\mathcal{L}_N^{10}$. Performance is measured on each separate set of NgramGen output lattices. Intuitively, we can predict that the performance will decrease from the NgramGen output for the first set of bags to the NgramGen output for the tenth set of bags.

We also form the union of lattices obtained from each set: for the i-th source sentence $\boldsymbol{f_i}$, we form the union of the lattices $\mathcal{L}_i^1$, ..., $\mathcal{L}_i^{10}$. This configuration is simply called *union*. We use the following settings for the regeneration decoder:

- We use a histogram pruning of 1000, that is a maximum number of 1000 states per column (see Section 6.5.2.2).

- We split the input according to punctuation (comma and semi-colon), with a maximum chunk length of 11, as described in Section 6.6.3. Note that contrary to the string regeneration settings, the splitting information comes from an MT hypothesis and not a reference. Therefore, it is legitimate to use input splitting for experimentation, decoding times become more reasonable and the NgramGen hypotheses are constrained to be close to the MT hypotheses, which we know have a relatively high quality already.

- As in Section 6.6.7, future cost is estimated with a unigram language model.

Results are summarised in Table 7.2. The main observation is that for all regeneration configurations, there is a substantial drop in performance with respect to the machine translation baseline. This is not unexpected since we are only using the language model feature for regeneration: except for the bag-of-words obtained from the translation hypothesis and the splitting points for more efficient decoding, all information coming from the translation system is discarded.

We measured the performance of the oracle hypothesis for the translation system and the regeneration systems on the Tune set. Again, regeneration oracle BLEU scores are substantially lower than the translation oracle BLEU. However, it is interesting to observe that the oracle hypothesis obtained from the union configuration is around 9 BLEU points above the MT baseline.

This indicates that there is a possible reordering of the 10-best hypotheses obtained by the translation decoder that improves translation quality.

Experiments were run with and without future cost estimates. We obtain small gains when future cost estimates are used (see column 3 vs. column1, column vs. column 2 and column 6 vs. column 5) but gains are very small compared to what was obtained when regenerating a well-formed English sentence as in Section 6.6.7.

Finally, as predicted, because the quality of the bags decreases from the 1st-best configuration to the 10th-best configuration, performance after regeneration from these sets of bags also decreases. In the next section, we will analyse the drop in performance for the regeneration configuration with respect to the MT baseline.

7.3 Analysis of the Loss of MT Hypotheses

We have observed in the previous section that simply reordering the n-best output of the translation decoder with our regeneration decoder gives a drop in performance with respect to the performance obtained by the baseline translation system. Although the translation system output lacks fluency due to word reordering problems, we know that the quality of this output is relatively high, thus we would like the output of the regeneration step to stay relatively close to the MT hypothesis. In this section, we will examine how often the MT hypothesis is not regenerated at all in the regeneration lattices, how often it is chosen as a regeneration hypothesis and at what point in the decoding procedure the MT hypothesis disappears from the regeneration lattice. We carry out this analysis on the 1st-best configuration from Table 7.2 and on the Tune set.

The MT hypothesis is present in the regeneration lattice 92.3% of the time when no future cost is estimated and 95.3% of the time with future cost estimates. This is somewhat reassuring because these numbers imply that the language model score of the MT hypothesis is high enough so that the MT hypothesis is not very frequently discarded by pruning in regeneration. In contrast, the MT hypothesis is chosen as the best hypothesis by the regeneration decoder only 43.6% of the time, with or without future costs. This means that, most of the time, the MT hypothesis is present in the regeneration lattice, but not chosen as the best regeneration hypothesis, simply because its model score, computed by the language model only, is lower than

Column	1	2	3	4	5	6
Set	Tune	Tune	Tune	Tune	MT08	MT08
Oracle		✓		✓		
MT baseline 1st pass	34.96	56.12	34.96	56.12	35.71	35.71
Future Cost			✓	✓		✓
1st-best	33.37	39.14	33.40	39.18	32.78	32.79
2nd-best	33.16	38.93	33.22	39.02	32.77	32.77
3rd-best	33.14	38.84	33.17	38.91	32.54	32.55
4th-best	33.13	38.80	33.16	38.87	32.35	32.39
5th-best	33.20	38.95	33.28	39.00	32.12	32.17
6th-best	33.08	38.74	33.12	38.81	32.33	32.40
7th-best	33.19	39.00	33.24	39.05	32.16	32.18
8th-best	32.98	38.79	33.04	38.84	32.11	32.14
9th-best	32.71	38.59	32.75	38.65	32.11	32.11
10th-best	33.05	38.81	33.11	38.88	32.50	32.55
union	32.85	43.96	32.90	44.00	32.11	32.18

Table 7.2: Regeneration experiments on input bags obtained from the 10-best hypotheses of a translation system. Case-insensitive BLEU scores are reported. 10-best lists are generated by an MT system. Each set of nth-best hypotheses is used as a set of input bags-of-words for the NgramGen decoder. Oracle performance on the Tune set is also measured. Regeneration decoding experiments are repeated with and without future cost estimates. Substantial drops in performance are observed in regeneration with respect to the translation baseline. Oracle scores for the union configuration show that by simply improving word reordering on the 10-best output of the MT system, gains may be achieved. Future cost estimates provide very small gains.

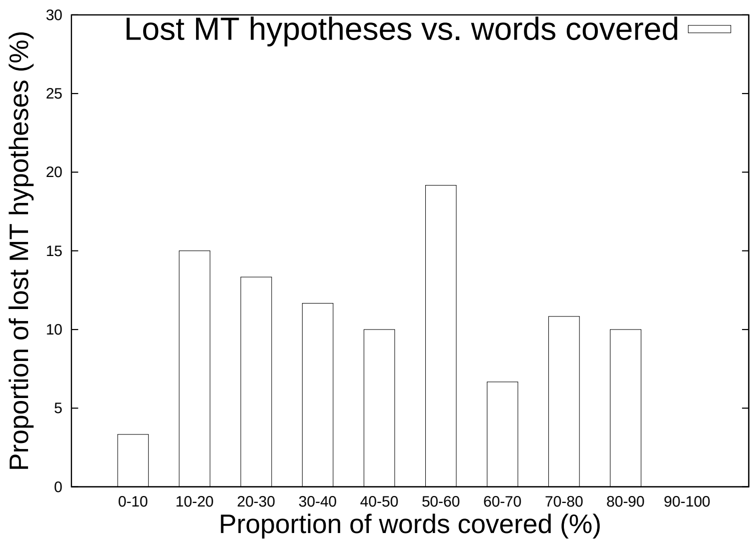

Figure 7.1: Analysis of when the MT hypothesis is lost. The x-axis represents bins of 10% for the relative size of the input bag-of-word. The y-axis is simply the count of instances.

the score of another regeneration hypothesis.

For the case when the MT hypothesis is not present in the regeneration lattice, we further analyse at which point the MT hypothesis is lost during the decoding procedure. As described in Section 6.5.2.2, a state is defined by coverage and history. For this analysis, we add a Boolean variable to the state definition that indicates whether the partial input is one of the partial hypotheses represented by this state. We can then easily compute after how many words covered the MT hypothesis has been lost.

Figure 7.1 illustrates the proportion of MT hypotheses lost vs. the proportion of input words covered. The plot shows that MT hypotheses can be lost quite early in the process, as early as when 20% of the size of the input bag-of-word has been covered. Using future cost estimates produces a very similar graph and does not alleviate the problem of losing MT hypotheses early in the decoding process.

From this analysis, we conclude that using the language model score as a single feature is not adequate for the task of regeneration from translation output. Designing and implementing additional features is a worthwhile opportunity for future work. In the sections to follow, while keeping a single feature for regeneration, we will study alternative ways to bring the regeneration output closer to the translation output.

7.4 Biased Language Model for Regeneration

We have shown in Section 7.3 that the MT hypothesis, which is known to be of relatively high quality, is not chosen as the best regeneration hypothesis more than half of the time and is sometimes not even present in the regeneration lattice. In this section, we describe a first solution to remedy this issue and bring the regeneration output closer to the translation output.

Our solution is simply to bias the language model towards the MT hypothesis. In order to do this, we first compute n-gram posterior probabilities from the 1st-pass translation lattices (see Section 2.10.2). These posteriors are then converted to integer counts in order to estimate a Good-Turing (Good, 1953; Chen and Goodman, 1998) smoothed 4-gram language model from these counts. We then interpolate this language model with the original language model with various interpolation parameters.

We report on experiments that are carried out on the set of bags obtained from the best MT hypotheses; experiments on lower quality bags gave analogous results. Interpolation parameters range from 0.4 (weight assigned to the original language model) to 0.99. We obtain the results in Table 7.3. We can see that we are able to obtain hypotheses that have the same quality as the MT hypotheses, and for an interpolation weight of 0.7, we obtain a very slight improvement on the tuning set. We also obtain very similar results when future cost estimates are included.

We use the best configuration from Table 7.3 (interpolation weight of 0.7) to run our regeneration decoder on the test set MT08 and obtain the results in Table 7.4. We obtain slight gains in terms of BLEU score on the MT08 test using this technique.

7.5 Translation and Regeneration Hypothesis Combination

In Section 7.4, we have described a first solution to bringing the quality of regeneration to the level of translation: biasing the language model using MT posteriors. Another possibility is exploit the system combination

Configuration	Tune	
MT baseline 1st pass	34.96	
Future Cost		✓
1st-best	33.37	33.40
1st-best 0.4	34.96	34.95
1st-best 0.5	34.96	34.96
1st-best 0.6	34.96	34.96
1st-best 0.7	34.97	34.96
1st-best 0.8	34.94	34.94
1st-best 0.9	34.90	34.90
1st-best 0.95	34.81	34.81
1st-best 0.99	34.11	34.13

Table 7.3: Regeneration experiments on the set of bags obtained from the best MT hypotheses, with biased language models. Case-insensitive BLEU scores are reported. The original language model is interpolated with a language model built from MT posteriors. A range of interpolation parameters is experimented on. It is possible to recover the MT performance when the interpolation weight for the MT posterior based language model is large enough. Including future cost estimates gives very similar results.

Configuration	Tune	MT08
MT baseline 1st pass	34.96	35.71
1st-best 0.7	34.97	35.75
1st-best 0.7 future cost	34.96	35.75

Table 7.4: Regeneration with a language model biased towards MT posteriors. Case-insensitive BLEU scores are reported. The best configuration from Table 7.3 is used to carry out regeneration on the MT08 test set. Very slight gains are observed.

Configuration	Tune	MT08
MT baseline +lmbr	36.80	37.59
NgramGen 1-best baseline	36.81	37.29
NgramGen union baseline	36.81	37.50
NgramGen 1-best baseline future cost	36.82	37.45
NgramGen union baseline future cost	36.81	37.50
NgramGen 1-best interpolation 0.7	36.82	37.39
NgramGen union interpolation 0.7	36.78	37.59
NgramGen 1-best interpolation 0.7 Future Cost	36.81	37.39
NgramGen union interpolation 0.7 Future Cost	36.77	**37.60**

Table 7.5: Experiments in hypothesis combination between the 5-gram rescored output of the translation decoder and the 5-gram output of the regeneration decoder.

paradigm. Because the regeneration decoder produces FSTs, we can reuse an existing hypothesis combination system based on FSTs and described in Section 2.10.3. Hypothesis combination can also be used in conjunction with the biased language model presented in Section 7.4.

The output of the regeneration decoder is first rescored with the large 5-gram language model described in Section 7.1 and then combined with the 5-gram rescored MT hypotheses. We experiment on the 1-best configuration (the set of bags is obtained from the best first-pass MT hypotheses) and on the union configuration (see Section 7.2). Configurations for the regeneration decoder also include the baseline reported in Section 7.2 as well as the biased language model method with an interpolation weight of 0.7 presented in Section 7.4.

Results are reported in Table 7.5. We can observe that using the union of lattices produced by NgramGen is beneficial for hypothesis combination: more hypotheses are present in lattices obtained with the union configuration. Using the biased language model described in the previous section, we are able to match and obtain a very slight gain over the LMBR rescored translation hypotheses.

7.6 Conclusion

In this chapter, we have demonstrated how to apply our regeneration decoder to the output of a translation system in a 10-best hypothesis rescoring setting. Because the regeneration decoder only uses a language model as single feature and throws away word order information obtained from the translation system except for splitting the input bag of words, by simply applying the NgramGen decoder to the output of the translation system, translation performance is degraded.

We have shown two possible ways of integrating the information from the translation system into the regeneration system in order to match and slightly outperform the translation quality obtained by the translation system. The first solution is to bias the language model used in regeneration towards the MT hypotheses by building an interpolated model between the original language model and a language model built on the MT posteriors. The second solution simply takes advantage of the hypothesis combination paradigm. In addition, both solutions were used in conjunction by combining the output of the MT system with the output of the regeneration system where the language model was biased.

One possible avenue for future work is to design and implement additional features in the decoder. The use of a few simple features, such as word and rule count, was already investigated but these features were not found to be beneficial. Because the regeneration system was introduced into translation with no loss in performance, another interesting next goal will be to keep identical performance and use the NgramGen system in order to improve fluency.

Chapter 8

Summary and Future Work

Statistical Machine Translation was originally framed as a source-channel model with two main components, the translation model and the language model. This separation into two modules provided opportunities for improving each module individually, with the hope that improving the language model would improve fluency while improving the translation model would improve adequacy. Current state-of-the-art SMT systems include many more components that interact in complex ways but also provide further individual improvement opportunities.

In this thesis, we present various improvements and analyses in the SMT pipeline, with an emphasis on refining hierarchical phrase-based models. In Section 8.1, we review the contributions of this thesis and in Section 8.2, we propose possible future work for each our contributions.

8.1 Review of Work

This thesis demonstrated various refinements in hierarchical phrase-based translation systems. A scalable infrastructure was developed for generating and retrieving rules from hierarchical grammars. We also developed an original training method for hierarchical phrase-based models which leverages computationally efficient HMM models usually used for word alignment. Other SMT pipeline components were refined: we studied the effect of additional grammar features for domain adaptation and provided recommendations for language models to be used in first-pass decoding and rescoring. Finally, we developed a string regeneration system from the ground up, analysed its competitive performance in the string regeneration task and its potential application to rescoring machine translation output. We now review each of these contributions.

8.1.1 Hierarchical Phrase-Based Grammars: Infrastructure

In order to enable more rapid experimentation and to ensure scalability for hierarchical phrase-based grammar extraction and retrieval, we developed a flexible MapReduce implementation of the grammar extraction and estimation algorithm (Dyer et al., 2008). Our implementation described in Chapter 3 includes extensions for the computation of provenance features (Chiang et al., 2011) and arbitrary corpus-level features can be added very easily to our framework.

A framework relying on the HFile format for efficient retrieval of hierarchical rules to be used in test set decoding was also developed. Comprehensive time and memory measurements demonstrated that our system is competitive with respect to linearly scanning, possibly concurrently, the datastructure containing the grammar extracted from parallel text (Pino et al., 2012).

8.1.2 Hierarchical Phrase-Based Grammars: Modelling

In the SMT pipeline, the word alignment module and the grammar extraction and estimation module usually interact through a fixed set of alignment links produced by the word alignment models. Our contribution was to allow for more communication between these two modules (de Gispert et al., 2010b). In Chapter 4, we first reframed the rule extraction procedure into a general framework. Under this framework, we introduced two novel methods for grammar extraction and estimation based on word alignment posterior probabilities.

The link posterior extraction method was based on alignment link posterior probabilities and allowed to extract a greater number of rules more reliably. The phrase pair posterior method was based on phrase pair alignment posterior probabilities and was shown to provide a better estimated translation model. Both methods provided substantial gains in translation quality in a medium size Chinese-English translation task. Alternative strategies for combining source-to-target and target-to-source word alignment models were also investigated; hypothesis combination obtained the best performance in our experiments.

8.1.3 Hierarchical Phrase-Based System Development

The translation model, refined in Chapter 4, is one of the main features in the log-linear model for translation. In Chapter 5, we studied the effect of provenance features (Chiang et al., 2011) for domain adaptation. The concept of provenance feature was extended from provenance lexical translation features to provenance translation features. We demonstrated that, when rule filtering thresholds are applied for each provenance and all surviving rules are kept, translation gains can be obtained.

We also provided recommendations for language modelling both in first-pass translation and rescoring. In the context of domain adaptation, we found that the best strategy for language modelling was off-line linear interpolation of various language models with weights tuned for perplexity on a held-out development set as opposed to log-linear interpolation with weights tuned by MERT. We also observed that when a few billion words of monolingual data were available, a two-pass strategy of first-pass decoding with a 4-gram language model trained on all data followed by 5-gram rescoring provided the best performance in our experiments. Finally we confirmed in the rescoring setting previous observations (Brants et al., 2007) in the first-pass decoding setting that Kneser-Ney smoothing produced better translation quality than Stupid Backoff smoothing. Thus we obtained translation quality gains over the top ranking system in the WMT13 Russian-English task (Pino et al., 2013).

8.1.4 Fluency in Hierarchical Phrase-Based Systems

One main reason for the lack of fluency in machine translation output is a discrepancy in word order between distant language pairs such as Chinese and English. In Chapter 6, we studied the problem of word ordering in isolation through the string regeneration task. We described a regeneration model inspired from the phrase-based translation model. We also built a regeneration decoder analogous to a stack-based decoder for translation. We achieved state-of-the-art performance on the string regeneration task. Finally, we analysed various capabilities of the decoder.

In Chapter 7, we applied string regeneration to the output of a translation system in a 10-best rescoring setting. Performance was degraded, however, by combining information obtained from the translation system with the regeneration system, we were able to match the quality of the translation output.

8.2 Future Work

The retrieval framework presented in Chapter 3 can exploit parallelisation in two ways for further speed improvements. First, source pattern instances can be generated for each test set source sentence individually; then each source sentence can processed in parallel. Because the HFile format does not allow multithreaded reads, one way to implement this solution is to reuse the multiprocessing capability of the MapReduce framework. Another opportunity for parallelisation similar to the one just described is to process queries in parallel. Again, care has to be taken to use a multiprocessing approach rather than a multithreading approach. Finally, the source pattern instance creation step can be fine-tuned in order to approach the speed of a real time system at the sentence level.

In Chapter 4, the rule extraction algorithm was described in a framework involving source constraints, alignment constraints, target constraints, a ranking function and a counting function. This provides an opportunity for future work by investigating alternatives to the definitions of these constraints and ranking and counting functions. Specifically, other word alignment models may be explored.

In Chapter 5, we recommended to use a two-pass decoding strategy and we also concluded that Kneser-Ney smoothing was beneficial over Stupid Backoff smoothing in 5-gram language model rescoring. These recommendations are valid for language models built on a few billion words. In the future, we will revise them with language models trained on an order of magnitude more monolingual data.

We successfully applied phrase-based models and decoding techniques to the task of string regeneration in Chapter 6. However, interesting future work may be carried out to investigate the effect of using more features than simply a language model. In addition, we may study how overlapping rules can be used to obtain more chances of covering the entire input if no unigram rules are used.

The effect of additional features for the regeneration decoder may also be studied in the regeneration of translation output setting. If translation quality as measured BLEU does not increase, efforts will be dedicated to improve fluency while maintaining translation quality. Finally, more sophisticated future cost estimates may be studied, such as n-gram language model with $n > 1$: this is possible because when regenerating translation output, the word ordering information is legitimately available in the input.

Bibliography

Alfred V. Aho and Jeffrey D. Ullman. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1): 37–56, 1969. 28

Cyril Allauzen, Mehryar Mohri, and Brian Roark. Generalized Algorithms for Constructing Statistical Language Models. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 557–564, 2003. 43

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of CIAA*, pages 11–23, 2007. 137

Satanjeev Banerjee and Alon Lavie. 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, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W05/W05-0909`. 41, 126

Anja Belz, Bernd Bohnet, Simon Mille, Leo Wanner, and Michael White. The surface realisation task: Recent developments and future plans. In *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*, pages 136–140, Utica, IL, May 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W12-1525`. 129

Oliver Bender, Evgeny Matusov, Stefan Hahn, Sasa Hasan, Shahram Khadivi, and Hermann Ney. The RWTH Arabic-to-English Spoken Language Translation System. In *Proceedings of ASRU*, pages 396–401, 2007. 36

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational linguistics*, 22(1):39–71, 1996. 10, 19, 20

Alexandra Birch, Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Constraining the phrase-based, joint probability statistical translation model. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 154–157, New York City, June 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-3123`. 76

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738. 44

Graeme Blackwood. *Lattice Rescoring Methods for Statistical Machine Translation*. PhD thesis, University of Cambridge, 2010. 45, 46, 137

Graeme Blackwood, Adrià de Gispert, and William Byrne. Efficient Path Counting Transducers for Minimum Bayes-Risk Decoding of Statistical Machine Translation Lattices. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 27–32, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P10-2006`. 45

Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, July 1970. ISSN 0001-0782. doi: 10.1145/362686.362692. 53, 57

Phil Blunsom, Trevor Cohn, and Miles Osborne. A Discriminative Latent Variable Model for Statistical Machine Translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio, June 2008a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1024`. 34

Phil Blunsom, Trevor Cohn, and Miles Osborne. Bayesian Synchronous Grammar Induction. In *Proceedings of NIPS*, volume 21, 2008b. 77

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. A Gibbs Sampler for Phrasal Synchronous Grammar Induction. In *Proceedings of the ACL*, pages 782–790, 2009. 77

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2201`. 8, 48, 110

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large Language Models in Machine Translation. In *Proceedings of EMNLP-ACL*, pages 858–867, 2007. 40, 43, 47, 49, 50, 51, 120, 121, 122, 163

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990. 10, 128

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993. 9, 10, 12, 14, 16, 82, 83

Christian Buck, Kenneth Heafield, and Bas van Ooyen. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavík, Iceland, May 2014. 121

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 255–262, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219872. 54

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E.

Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4:1–4:26, June 2008. doi: 10.1145/1365815.1365816. 50, 55

Jean-Cédric Chappelier and Martin Rajman. A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Proceedings of TAPD*, pages 133–137, 1998. 42

Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. The bloomier filter: An efficient data structure for static support lookup tables. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 30–39, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics. ISBN 0-89871-558-X. URL `http://dl.acm.org/citation.cfm?id=982792.982797`. 53

Stanley F. Chen and Joshua Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Harvard University, 1998. 38, 39, 157

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219873. URL `http://www.aclweb.org/anthology/P05-1033`. 10, 27

David Chiang. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228, 2007. 10, 27, 28, 29, 30, 31, 33, 101

David Chiang, Steve DeNeefe, and Michael Pust. Two easy improvements to lexical weighting. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 455–460, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. 118, 162, 163

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages

176–181, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-2031`. 102

Martin Cmejrek, Bowen Zhou, and Bing Xiang. Enriching SCFG Rules Directly From Efficient Bilingual Chart Parsing. In *Proceedings of IWSLT*, pages 136–143, 2009. 77

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *Computational linguistics*, 36(3):505–533, 2010a. 31, 34, 99

Adrià de Gispert, Juan Pino, and William Byrne. Hierarchical phrase-based translation grammars extracted from alignment posterior probabilities. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 545–554, Cambridge, MA, October 2010b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D10-1053`. 72, 162

Adrià de Gispert, Marcus Tomalin, and William J. Byrne. Word ordering with phrase-based grammars. In *Proceedings of the EACL*, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. 129

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, January 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327492. 49, 59

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 14, 76

John DeNero and Dan Klein. The Complexity of Phrase Alignment Problems. In *Proceedings of ACL*, pages 25–28. Association for Computational Linguistics, 2008. 76, 77

Yonggang Deng. *Bitext Alignment for Statistical Machine Translation*. PhD thesis, Johns Hopkins University, 2005. 89

Yonggang Deng and William Byrne. HMM Word and Phrase Alignment for Statistical Machine Translation. In *Proceedings of HLT/EMNLP*, pages 169–176, 2005. 98

Yonggang Deng and William Byrne. HMM Word and Phrase Alignment for Statistical Machine Translation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):494–507, 2008. 14, 17, 75, 98, 112

P. Deutsch and J.-L. Gailly. Zlib compressed data format specification version 3.3, 1996. 56

George Doddington. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. 41

Chris Dyer, Aaron Cordova, Alex Mont, and Jimmy Lin. Fast, easy, and cheap: Construction of statistical machine translation models with MapReduce. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 199–207, Columbus, Ohio, June 2008. Association for Computational Linguistics. 49, 51, 52, 59, 162

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala, Sweden, July 2010. Association for Computational Linguistics. 64

Jonathan G. Fiscus. A Post-Processing System To Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER). In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 347–352, 1997. 46

George Foster and Roland Kuhn. Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-0717`. 115

George Foster and Roland Kuhn. Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 242–249, Athens, Greece, March 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W09-0439`. 117

George Foster, Roland Kuhn, and Howard Johnson. Phrasetable Smoothing for Statistical Machine Translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 53–61, Sydney, Australia, July 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-1607`. 77

George Foster, Boxing Chen, and Roland Kuhn. Simulating discriminative training for linear mixture adaptation in statistical machine translation. pages 183–190, 2013. 116

Alex Franz, Shankar Kumar, and Thorsten Brants. 1993-2007 United Nations Parallel Text, 2013. 5

Edward Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, September 1960. ISSN 0001-0782. doi: 10.1145/367390.367400. 52, 53

Michel Galley and Christopher D. Manning. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N10-1140`. 27

Juri Ganitkevitch, Yuan Cao, Jonathan Weese, Matt Post, and Chris Callison-Burch. Joshua 4.0: Packing, pro, and paraphrases. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 283–291, Montréal, Canada, June 2012. Association for Computational Linguistics. 53

Qin Gao and Stephan Vogel. Training phrase-based machine translation models on the cloud: Open source machine translation toolkit chaski. *The Prague Bulletin of Mathematical Linguistics*, 93:37–46, January 2010. doi: 10.2478/v10108-010-0004-8. 51

Wolfgang Gentzsch. Sun grid engine: Towards creating a compute power grid. In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, CCGRID '01, pages 35–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1010-8. 65, 144

Ulrich Germann. Aligned hansards of the 36th parliament of canada, Natural Language Group of the USC Information Sciences Institute, 2001. URL `http://www.isi.edu/natural-language/download/hansard/`. 5

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 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, July 2001. Association for Computational Linguistics. doi: 10.3115/1073012.1073042. URL `http://www.aclweb.org/anthology/P01-1030`. 6

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5. doi: 10.1145/945445.945450. 58

I.J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237, 1953. 157

Amit Goyal, Hal Daume III, and Suresh Venkatasubramanian. Streaming for large scale nlp: Language modeling. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 512–520, Boulder, Colorado, June 2009. Association for Computational Linguistics. 54

David Guthrie and Mark Hepple. Storing the web in memory: Space efficient language models with constant time retrieval. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 262–272, Cambridge, MA, October 2010. Association for Computational Linguistics. 53

Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings* *of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219911. URL http://www.aclweb.org/anthology/P05-1071. 5

Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. Dependency based chinese sentence realization. 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 809–816, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1091. 129

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. 53, 135

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-2121. 51, 113

Mark Hopkins, Greg Langmead, and Tai Vo. Extraction programs: A unified approach to translation rule extraction. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 523–532, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2166. 77

Matthias Huck, Joern Wuebker, Felix Rietig, and Hermann Ney. A phrase orientation model for hierarchical machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 452–463, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2258. 27

John Hutchins. From first conception to first demonstration: the nascent years of machine translation, 1947–1954. a chronology. *Machine Translation*, 12(3):195–252, 1997. ISSN 0922-6567. doi:

10.1023/A:1007969630568. URL http://dx.doi.org/10.1023/A%3A1007969630568. 10

John Hutchins and Evgenii Lovtskii. Petr petrovich troyanskii (1894–1950): A forgotten pioneer of mechanical translation. *Machine Translation*, 15 (3):187–221, 2000. ISSN 0922-6567. doi: 10.1023/A:1011653602669. URL http://dx.doi.org/10.1023/A%3A1011653602669. 10

Gonzalo Iglesias. *Hierarchical Phrase-based Translation with Weighted Finite-State Transducers*. PhD thesis, University of Vigo, Spain, 2010. 42

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. Rule Filtering by Pattern for Efficient Hierarchical Translation. In *Proceedings of EACL*, pages 380–388, 2009a. 31, 33, 101

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 433–441, Boulder, Colorado, June 2009b. Association for Computational Linguistics. 42, 49

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. The HiFST System for the EuroParl Spanish-to-English Task. In *Proceedings of SEPLN*, pages 207–214, 2009c. 33

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of ICASSP*, volume 1, pages 181–184, 1995. 39

Kevin Knight. Automatic language translation generation help needs badly. In *MT Summit XI Workshop on Using Corpora for NLG: Language Generation and Machine Translation, Keynote Address*, 2007. 125

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005. 5, 116

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010. 9, 18, 21, 26, 36

Philipp Koehn and Josh Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-0733`. 116, 117

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical Phrase-Based Translation. In *Proceedings of HLT/NAACL*, pages 48–54, 2003. 10, 16, 21, 106

Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of HLT-NAACL*, pages 169–176, 2004. 44, 45

Shankar Kumar, Franz J. Och, and Wolfgang Macherey. Improving Word Alignment with Bridge Languages. In *Proceedings of EMNLP-CoNLL*, pages 42–50, 2007. 75, 79

Abby Levenberg and Miles Osborne. Stream-based randomised language models for SMT. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 756–764, Singapore, August 2009. Association for Computational Linguistics. 54

Philip M Lewis II and Richard Edwin Stearns. Syntax-directed transduction. *Journal of the ACM (JACM)*, 15(3):465–488, 1968. 28

Chin-Yew Lin and Franz Josef Och. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of Coling 2004*, pages 501–507, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING. 44

Jimmy Lin and Chris Dyer. *Data-Intensive Text Processing with MapReduce*. Morgan & Claypool Publishers, 2010. 49, 52

Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. Weighted Alignment Matrices for Statistical Machine Translation. In *Proceedings of EMNLP*, pages 1017–1026, 2009. 74

Adam Lopez. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*

(EMNLP-CoNLL), pages 976–985, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 54, 64

Adam Lopez. Statistical machine translation. *ACM Computing Surveys*, 40 (3):8:1–8:49, August 2008. ISSN 0360-0300. doi: 10.1145/1380584.1380586. URL `http://doi.acm.org/10.1145/1380584.1380586`. 9

Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. In *Proceedings of the first ACM-SIAM symposium on Discrete algorithms*, pages 319–327. Society for Industrial and Applied Mathematics, 1990. 54

Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002. 54

Daniel Marcu and William Wong. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of EMNLP*, page 139. Association for Computational Linguistics, 2002. 76

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 708–717, Singapore, August 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D09/D09-1074`. 118

Evgeny Matusov, Richard Zens, and Hermann Ney. Symmetric word alignments for statistical machine translation. In *Proceedings of Coling 2004*, pages 219–225, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING. 75, 79

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013. 10

Franz J. Och. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*, pages 160–167, 2003. 20, 36, 42, 116

Franz Josef Och and Hermann Ney. 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, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073133. URL `http://www.aclweb.org/anthology/P02-1038`. 11, 20

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003. 16, 17

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational linguistics*, 30(4):417–449, 2004. 10, 11, 21, 23

Franz Josef Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, 1999. 10, 11, 16, 19, 21

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `http://www.aclweb.org/anthology/P02-1040`. 3, 36, 41

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fourth edition, 2009. 151

David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware/software Interface*. Morgan Kaufmann, 2009. 58

Adam Pauls and Dan Klein. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 258–267, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. 53

Adam Pauls, Dan Klein, David Chiang, and Kevin Knight. Unsupervised Syntactic Alignment with Inversion Transduction Grammars. In *Proceedings of the HLT-NAACL*, pages 118–126, 2010. 77

Juan Pino, Gonzalo Iglesias, Adrià de Gispert, Graeme Blackwood, Jamie Brunning, and William Byrne. The CUED HiFST System for the WMT10 Translation Shared Task. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 155–160, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-1722. 47

Juan Pino, Aurelien Waite, and William Byrne. Simple and efficient model filtering in statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, 98(1):5–24, 2012. 48, 55, 162

Juan Pino, Aurelien Waite, Tong Xiao, Adrià de Gispert, Federico Flego, and William Byrne. The University of Cambridge Russian-English system at WMT13. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 200–205, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2225. 110, 163

Matt Post, Juri Ganitkevitch, Luke Orland, Jonathan Weese, Yuan Cao, and Chris Callison-Burch. Joshua 5.0: Sparser, better, faster, server. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 206–212, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2226. 52

Česlav Przywara and Ondřej Bojar. eppex: Epochal phrase table extraction for statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, 96:89–98, 2011. 54

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 14, 83

Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3:57–87, 3 1997. ISSN 1469-8110. doi: 10.1017/S1351324997001502. URL http://journals.cambridge.org/article_S1351324997001502. 125, 126, 128

Markus Saers and Dekai Wu. Improving phrase-based translation via word alignments from stochastic inversion transduction grammars. In *Proceed-*

ings of the HLT-NAACL Workshop on Syntax and Structure in Statistical Translation, pages 28–36, 2009. 77

Hendra Setiawan and Philip Resnik. Generalizing Hierarchical Phrase-based Translation using Rules with Adjacent Nonterminals. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 349–352, Los Angeles, California, June 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N10-1052. 31

Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948. URL http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf. 10

Libin Shen, Jinxi Xu, and Ralph Weischedel. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proceedings of the ACL*, pages 577–585, 2008. 36

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1383, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-1135. 5

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Tate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231. Citeseer, 2006. 41

Andreas Stolcke. SRILM–An Extensible Language Modeling Toolkit. In *Proceedings of ICSLP*, volume 3, pages 901–904, 2002. 116

David Talbot and Thorsten Brants. Randomized language models via perfect hash functions. In *Proceedings of ACL-08: HLT*, pages 505–513, Columbus, Ohio, June 2008. Association for Computational Linguistics. 53

David Talbot and Miles Osborne. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 512–519, Prague, Czech Republic, June 2007a. Association for Computational Linguistics. 53, 58

David Talbot and Miles Osborne. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 468–476, Prague, Czech Republic, June 2007b. Association for Computational Linguistics. 53, 54, 58

Alicia Tribble, Stephan Vogel, and Alex Waibel. Overlapping phrase-level translation rules in an smt engine. In *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE'03), Beijing, China*, 2003. 144

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 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, October 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D08-1065`. 45

Nicola Ueffing, Franz Josef Och, and Hermann Ney. Generation of word graphs in statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 156–163. Association for Computational Linguistics, July 2002. doi: 10.3115/1118693.1118714. URL `http://www.aclweb.org/anthology/W02-1021`. 42

Ashish Venugopal and Andreas Zollmann. Grammar based statistical mt on hadoop: An end-to-end toolkit for large scale pscfg based mt. *The Prague Bulletin of Mathematical Linguistics*, 91:67–77, January 2009. doi: 10.2478/v10108-009-0017-3. 52, 53

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. Wider Pipelines: N-Best Alignments and Parses in MT Training. In *Proceedings of AMTA*, pages 192–201, 2008. 74

Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz J Och, and Juri Ganitkevitch. Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1363–1372. Association for Computational Linguistics, 2011. 5

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-Based Word Alignment in Statistical Translation. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 836–841. Association for Computational Linguistics, 1996. 14, 72, 83

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 852–860, Athens, Greece, March 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/E09-1097`. 126, 128, 129, 141

Warren Weaver. Translation. *Machine Translation of Languages*, 14:15–23, 1955. 10

Jonathan Weese, Juri Ganitkevitch, Chris Callison-Burch, Matt Post, and Adam Lopez. Joshua 3.0: Syntax-based machine translation with the thrax grammar extractor. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 478–484, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. 64

John S. White, Theresa A. O'Connell, and Lynn M. Carlson. Evaluation of machine translation. In *Proceedings of the Workshop on Human Language Technology*, pages 206–210. Association for Computational Linguistics, 1993. 125

Dekai Wu. Stochastic Inversion Transduction Grammars, with Application to Segmentation, Bracketing, and Alignment of Parallel Corpora. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1328–1337, 1995. 27

Dekai Wu. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–403, 1997. 27

Chi-Hsin Yu and Hsin-Hsi Chen. Detecting word ordering errors in Chinese sentences for learning Chinese as a foreign language. In *Proceedings of COLING 2012*, pages 3003–3018, Mumbai, India, December 2012. The COLING 2012 Organizing Committee. URL `http://www.aclweb.org/anthology/C12-1184`. 126

Xiaoyang Yu. Estimating language models using hadoop and hbase. Master's thesis, University of Edinburgh, Edinburgh, 2008. 50, 55

Richard Zens and Hermann Ney. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 492–499, Rochester, New York, April 2007. Association for Computational Linguistics. 53

Ying Zhang and Stephan Vogel. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT-05*, pages 294–301, 2005. 54

Ying Zhang and Stephan Vogel. Suffix array and its applications in empirical natural language processing. Technical Report CMU-LTI-06-010, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, December 2006. 54

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. Distributed language modeling for n-best list re-ranking. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 216–223, Sydney, Australia, July 2006. Association for Computational Linguistics. 54

Yue Zhang. Partial-tree linearization: generalized word ordering for text synthesis. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2232–2238. AAAI Press, 2013. 129, 141

Yue Zhang and Stephen Clark. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847, Prague, Czech

Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-1106`. 5

Yue Zhang and Stephen Clark. Syntax-Based Grammaticality Improvement using CCG and Guided Search. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D11-1106`. 129, 141

Yue Zhang, Graeme Blackwood, and Stephen Clark. Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 736–746, Avignon, France, April 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/E12-1075`. 129, 141

Andreas Zollmann and Ashish Venugopal. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City, June 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-3119`. 74