Factored Language Models for Statistical Machine Translation

Amittai E. Axelrod

Master of Science by Research

Institute for Communicating and Collaborative Systems

Division of Informatics

University of Edinburgh

2006

Abstract

Machine translation systems, as a whole, are currently not able to use the output of linguistic tools, such as part-of-speech taggers, to effectively improve translation performance. However, a new language modeling technique, Factored Language Models can incorporate the additional linguistic information that is produced by these tools.

In the field of automatic speech recognition, Factored Language Models smoothed with Generalized Parallel Backoff have been shown to significantly reduce language model perplexity. However, Factored Language Models have previously only been applied to statistical machine translation as part of a second-pass rescoring system.

In this thesis, we show that a state-of-the-art phrase-based system using factored language models with generalized parallel backoff can improve performance over an identical system using trigram language models. These improvements can be seen both with the use of additional word features and without. The relative gain from the Factored Language Models increases with smaller training corpora, making this approach especially useful for domains with limited data.

Acknowledgements

I greatly appreciate the many discussions I had with my supervisor Miles Osborne, both about the project and not. Philipp Koehn kindly provided most of the resources I used for this thesis; I could not have done it without him.

I am indebted to Chris Callison-Burch, Danielle Li and Alexandra Birch-Mayne for being quick to proofread and give helpful advice. Finally, I credit my consumption of 601 mission-critical cups of tea to the diligent efforts of Markus Becker, Andrew Smith, and my colleagues at 2 Buccleuch Place, 3rd Right.

Thank you.

```
ahtena% zwgc -ttymode
ahtena% znol
asedeno: darkmatter.MIT.EDU     VT-Owl  Sat Jun 17 15:54:07 2006
bighair: grumpy-fuzzball.mit.edu    pts/4  Sat Jun 17 15:04:52
chope: geskekelud.mit.edu       pts/2   Sat Jun 17 09:09:15 2006
davidmac: grobda.mit.edu        VT-Owl  Wed Jun  7 17:04:04 2006
goodell: minerva.eecs.harvard.edu       pts/1   Sun Jan 22 15:37:19
huberth: scyther.mit.edu  localhost:10.0  Sat Jun 17 17:02:39 2006
jayp: updog.mit.edu    pts/2  Tue May 16 20:41:01 2006
ladyada: mosfet.mit.edu  pts/6   Tue Jun 13 19:57:45 2006
kilroi: wandering-jew.mit.edu   pts/26  Sat Jun 17 15:44:18 2006
seph: all-night-tool.mit.edu pts/37  Sat Jun 17 17:53:14 2006
tibbetts: innocuous.mit.edu  pts/1   Tue Jun 13 09:09:16 2006
yangr: angry.mit.edu   pts/10  Fri May 26 19:27:31 2006
ahtena% zwrite -c amittai
Type your message now. End with control-D or a dot on a line by itself.
word
.
ahtena%
ahtena%
```

Declaration

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

(*Amittai E. Axelrod*)

Table of Contents

Chapter 1

Introduction

Statistical Machine Translation (SMT) is a probabilistic framework for translating text from one language to another, based on models induced automatically from a parallel corpus. Generally speaking, a statistical machine translation system is composed of three parts: a translation model, which captures the correspondence between words and phrases in different languages; a language model, which reflects the fluency of the target language; and a decoder, which incorporates the translation and language models to perform the actual translation.

Most work in statistical machine translation has focused on developing better translation models and software tools, rather than on improving language models. The dominant approach to language modeling is to use a simple n-gram language model, which probabilistically predicts the next word in a sequence based on the preceding few words. However, there is strong interest in using linguistic tools, such as parts-of-speech taggers or suffix strippers, as an aid to statistical machine translation. Machine translation systems, as a whole, are currently not able to use the linguistic output of these tools to effectively improve translation performance. However, it is possible to incorporate the additional linguistic information that is produced by these tools in a Factored Language Model (FLM).

A major advantage of factored language models over standard n-gram language models is the availability of a new smoothing option for estimating the likelihood of previously unseen word sequences. The new smoothing method for factored language models, Generalized Parallel Backoff (GPB), enables the model to combine multiple lower-order estimates instead of selecting which estimate to use in advance. In the field of automatic speech recognition, factored language models smoothed with generalized parallel backoff have been shown to significantly reduce language model perplexity.

However, factored language models have previously only been applied to statistical machine translation as part of a second-pass rescoring system.

In this thesis, we incorporate factored language models into a phrase-based statistical machine translation decoder and compare performace with that of a similar system using a standard trigram language model. We present results trained on the Europarl French-English corpus, as well as on a range of smaller corpora. We show that a system using factored language models smoothed with generalized parallel backoff shows better performance than a state-of-the-art system using trigram language models, both with the use of additional word features and without. This demonstrates both that factored language models can incorporate linguistic knowledge to improve translation, and that their richer smoothing abilities make them useful even when linguistic tools are not available for the target language. The relative gain from using the factored language models increases with smaller training corpora, making this approach especially useful for domains with limited data.

- Chapter 2 gives an overview of statistical machine translation. It describes the basic types of translation models, particularly the current state-of-the-art phrase-based, log-linear, models that we use as our experimental framework.

- Chapter 3 is an introduction to statistical language models and their smoothing techniques, emphasizing the the n-gram model that is the baseline for our experiments. It then describes factored language models with generalized parallel backoff.

- Chapter 4 motivates the use of factored language models within a statistical machine translation system, both by highlighting the merits of factored language models and by describing the shortcomings of the n-gram language models we seek to replace.

- Chapter 5 describes the experimental setup used to evaluate the effectiveness of factored language models with generalized parallel backoff for translation.

- Chapter 6 presents and explains experimental results. It shows that factored language models can improve translation performance in addition to being able to model the corpus significantly better. Furthermore, these benefits increase when factored language models are used for translation on smaller corpora.

- Chapter 7 discusses the implications of the experimental results, and suggests future research directions.

Chapter 2

Statistical Machine Translation

> "When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode' "(Weaver 1949/1955)

This chapter provides a general history of statistical machine translation models, up to the current state of the art. Section 2.1 explains the origins of the fundamental equation of machine translation. The canonical types of machine translation are introduced in section 2.2, with word-based systems detailed in section 2.3. Lastly, section 2.4 describes the phrase-based translation models which form the framework for this thesis.

2.1 History

Modern-day machine translation originated in 1949, when Warren Weaver suggested applying statistical and cryptographic techniques from the recently-propounded field of communication theory towards the problem of text translation from one natural language to another. However, early efforts in this direction faced philosophical doubts about whether fully automatic, high quality, machine translation was even possible in principle, even without the computational limitations of the time. Furthermore, in 1966 the ALPAC (Automatic Language Processing Advisory Committee) report concluded that machine translation could not overcome the low cost of and low demand for human translators, helpfully killing off most machine translation research to prevent "a possible excess of translation" (Pierce and Carroll 1966).

Automatic translation then became the domain of rule-based and knowledge-based systems. These systems were labor-intensive, as they performed translations based

on linguistic patterns, rules and exceptions that were input by hand. Some of these systems are still in use commercially.

The next research breakthrough came in the early 1990's when large bilingual corpora, such as the Hansards (Canadian parliamentary proceedings), were readily available in machine-readable formats. Furthermore, statistical methods had been used in speech recognition and natural language processing for over a decade, and computers were sufficiently powerful to use them effectively. Researchers at IBM proposed a statistical approach to machine translation based on a probabilistic dictionary (Brown *et al.* 1990). Shortly thereafter, methods for automatically aligning sentence pairs in a bilingual corpus were developed by Gale and Church (1991) and Brown *et al.* (1991).

Taking advantage of the newly available parallel corpora, Brown *et al.* (1993) described a series of five statistical models for machine translation, called the IBM Models. These models form the basis for word-based statistical machine translation. These models show how a sentence could generate its translation, word by word, using Shannon's (1948) noisy channel model of communication and with parameters estimated from a parallel corpus. Figure 2.1 shows how this stochastic process can be used to model the machine translation problem.

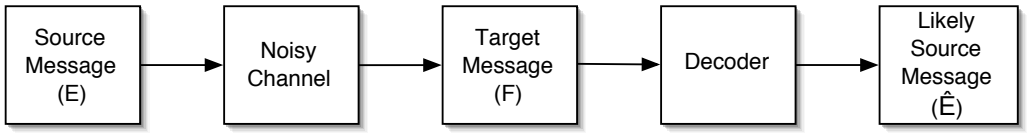

Figure 2.1: Translating with Shannon's noisy-channel model of communication

Brown *et al.* explained their perspective on the statistical translation process:

> A string of English words, e, can be translated into a string of French words in many different ways. Often, knowing the broader context in which e occurs may serve to winnow the field of acceptable French translations, but even so, many acceptable translations will remain; the choice among them is largely a matter of taste. In statistical translation, we take the view that every French string, f, is a possible translation of e. We assign to every pair of strings (e, f) a number $P(f|e)$, which we interpret as the probability that a translator, when presented with e will produce f as his translation. We further take the view that when a native speaker of French produces a string of French words, he has actually conceived of a string of English words, which he translated mentally. Given a French string f, the job of our translation system is to find the string e that the native speaker had in mind when he produced f. We minimize our chance of error by choosing that English string $\hat{e}$ for which $P(e|f)$ is greatest.

Treating a French sentence F as an encoding of an English sentence means that the probability of E being the intended translation of F can be expressed using Bayes' rule:

$$P(E|F) = \frac{P(E) \cdot P(F|E)}{P(F)} \tag{2.1}$$

Because the denominator is independent of the translation hypothesis E, finding the most likely translation $\hat{E}$ can be reduced to maximizing the numerator. The resulting "Fundamental Equation of Machine Translation" is:

$$\hat{E} = \underset{E}{\text{argmax}} \; P(E) \cdot P(F|E) \tag{2.2}$$

The term $P(E)$ represents the *language model probability*, and $P(F|E)$ is the *translation model probability*. As in speech recognition, the a priori probability of E can be thought of as the likelihood that E is a fluent sentence in the target language. The language model probability is high for well-formed English sentences, independent of their relationship to the French one. The translation model probability is the probability that the sentence F can be generated from E. The translation model probability is large for English sentences, regardless of their grammaticality, that have the necessary words in roughly the right places to explain the French sentence. Equation 2.2 therefore serves to assign a high probability to well-formed English sentences that account well for the French sentence.

Language model research has historically been independent of work on translation models. In this thesis, we apply a recently-developed language model to statistical machine translation in a novel way, namely using a factored language model within a phrase-based decoder. However, we postpone discussion of language models until Chapter 3 and focus on explaining the machine translation process for the rest of this chapter.

2.2 Contemporary Translation Modeling Overview

Machine translation is a hard problem in part because there is a trade-off between the methods with which we would like to translate and those that we can readily compute. There are several main strategies for attacking the translation problem, but most of them are still out of reach.

Warren Weaver viewed languages as:

> "[...] tall closed towers, all erected over a common foundation. Thus it may be true that the way to translate from Chinese to Arabic [...] is not

> to attempt the direct route [...]. Perhaps the way is to descend, from each language, down to the common base of human communication– the real but as yet undiscovered universal language [...].” (Weaver 1949/1955)

Weaver's notion of decoding linguistic utterances using the fundamental knowledge representation formalism of an interlingua, while ideal, is utterly impractical with the current state of the fields of Linguistics and Natural Language Processing. Other ambitious methods include semantic transfer approaches, where the meaning of the source sentence is derived via semantic parsing and the translation is then generated from it. There are cases where the translation is too literal to be clear, but the main obstacle is the lack of semantically annotated parallel corpora.

Next are syntactic transfer approaches, such as Yamada and Knight (2001), where the source sentence is initially parsed and then transformed into a syntactic tree in the target language. The translated sentence is then generated from this tree. Syntax-based approaches produce correctly ordered translations, but may miss the semantics of the sentence. However, the main obstacle is again the requirement of parsing tools, or more precisely the money to fund their research, a requirement that is currently not yet met for many languages.

Word-level translation models adopt a simple approach. They translate the source sentence word for word into the target language and then reorder the words until they make the most sense. This is the essence of the IBM translation models. It has the disadvantage of failing to capture semantic or syntactic information from the source sentence, thus degrading the translation quality. The great advantage of word-level translation models, however, is that they are readily trained from available data and do not require further linguistic tools that may not be available. Because of this, word-level translation models form the basis of statistical machine translation.

2.3 Word-Based Translation Modeling

Most research into improving statistical machine translation has focused on improving the translation model component of the Fundamental Equation (2.2). We now examine this translation model probability, $P(F|E)$, in more detail.

The translation model probability cannot be reliably calculated based on the sentences as a unit, due to sparseness. Instead, the sentences are decomposed into a sequence of words. In the IBM models, words in the target sentence are aligned with the word in the source sentence that generated them. The translation in Figure 2.2 con-

tains an example of a reordered alignment (*black cat* to *chat noir*) and of a many-to-one alignment (*fish* to *le poisson*).

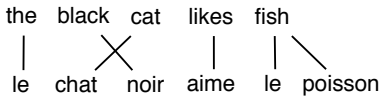

Figure 2.2: Example of a word-aligned translation from English to French.

2.3.1 Definitions

Suppose we have E, an English sentence with I words, and F, a French sentence with J words:

$$\begin{aligned} E &= e_1, e_2, \ldots e_I \\ F &= f_1, f_2, \ldots f_J \end{aligned}$$

According to the noisy channel model, a word alignment $a_j : j \rightarrow i$ aligns a French word f_j with the English word e_i that generated it (Brown *et al.* 1993). We call A the set of word alignments that account for all words in F:

$$A = a_1, a_2, \ldots a_J$$

Then the probability of the alignment is the product of individual word alignment probabilities:

$$P(F,A|E) = \prod_{j=1}^{J} P(f_j, a_j|E)$$

$P(F,A|E)$ is called the statistical alignment model. However, there are multiple possible sets A of word alignments for a sentence pair. Therefore, to obtain the probability of a particular translation, we sum over the set $\mathbf{A}$ of all such A:

$$P(F|E) = \sum_{A \in \mathbf{A}} P(F,A|E)$$

2.3.2 Calculating the Translation Model Probability

$P(F|E)$ would be simple to compute if word-aligned bilingual corpora were available. Because they are generally unavailable, word-level alignments are estimated from the sentence-aligned corpora that are accessible for many language pairs. Brown *et al.*

perform this alignment extraction by using an Expectation Maximization (EM) algorithm (Dempster *et al.* 1977). EM methods produce estimates for hidden parameter values by maximizing the likelihood probability of a training set. In this case, it estimates word translation probabilities (and other hidden variables of the translation probability) by selecting those that maximize the sentence alignment probability of the sentence-aligned parallel corpus. The more closely the training set represents the set the machine translation system will be used on, the more accurate the EM parameter estimates will be. However, EM is not guaranteed to produce the globally optimal translation probabilities.

For a training corpus of $\boldsymbol{S}$ aligned sentence pairs, the EM algorithm estimates the translation model parameters θ:

$$\theta = \underset{\theta}{\operatorname{argmax}} \prod_{S=1}^{\boldsymbol{S}} \sum_{A \in \boldsymbol{A}} P(F,A|E)$$

Given the word alignment probability estimates θ, the IBM translation models then collectively compute the translation model $P(F|E)$ for a word-based statistical machine translation system.

Word-based statistical machine translation models necessitated some elaborate model parameters (e.g. fertility, the likelihood of a word translating to more than one word) to account for how some real-world translations could be produced. A limitation of word-based models was their handling of reordering, null words, and non-compositional phrases (which cannot be translated as a function of their constituent words). The word-based system of translation models has been improved upon by recent phrase-based approaches to statistical machine translation, which use larger chunks of language as their basis.

2.4 Phrase-Based Translation Modeling

The term "phrase" denotes a multi-word segment, and not a syntactic unit such as a noun phrase. Using phrase alignments instead of word alignments to calculate the translation probability allows the inclusion of local context information in the translation model. Figure 2.3 shows an example of a phrase-aligned German to English translation, including a one-to-many alignment (*fliege* to *will fly*) and a many-to-many alignment (*nach Kanada* to *in Canadia*), which word-aligned translation models do not permit. This leads to better word choice in translation and more accurate word reordering when needed.

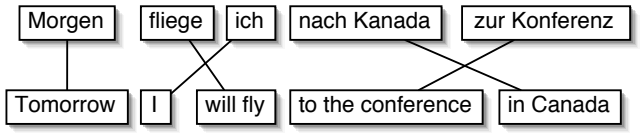

Figure 2.3: Example of a phrase-aligned translation from German to English.

Phrase-level alignments have been a more recent focus in statistical machine translation research. One approach has been to use phrases, corresponding to syntactic subtrees in a parsed sentence, as part of a syntax-based translation model (Yamada and Knight 2001).

An intermediate step between word-based and more linguistically-oriented translation models is that of using phrase-based translation models that use only the surface form of words– that is, just the words as they appear in the text. Phrase-based translation models, as proposed by Och (2003), use the automatically-generated word-level alignments from the IBM models to extract phrase-pair alignments. The phrase-pair alignment probabilities are then used as the fundamental units of the translation model instead of the word alignment probabilities. Koehn *et al.* (2003) show that phrase-based translation systems outperform the syntax-based translation model of Yamada and Knight (2001), so the approach is the current state-of-the-art.

2.4.1 A Simple Phrase-Based Model

We can use the noisy-channel approach to model phrase-based translation, just as in the word-based models of Section 2.3. Suppose E is an English sentence, and F is a French sentence. If we segment the source sentence F into a sequence of I phrases:

$$F = f_1, f_2, \ldots f_I$$

All possible segmentations are considered equiprobable. Each phrase f_i in F is translated into an English phrase, called e_i, using a probability distribution $\phi(f_i|e_i)$. The translation probability is now:

$$\begin{aligned} P(F|E) &= P(f_1^I|e_1^I) \\ &= \frac{P(e_1^I) \cdot P(e_1^I|f_1^I)}{P(f_1^I)} \end{aligned}$$

The most likely translation sentence $\hat{E}$ is:

$$\begin{aligned}\hat{E} &= \underset{E}{\operatorname{argmax}} \prod_{i=1}^{N} \alpha_i^{x_i(E|F)} \\ &= \underset{E}{\operatorname{argmax}} \; P(e_1^I) \cdot P(e_1^I | f_1^I) \\ &= \underset{E}{\operatorname{argmax}} \prod_{i=1}^{I} P(e_i) \cdot \phi(f_i|e_i)\end{aligned}$$

While this noisy-channel-based translation model is a reasonable extension of the word-based models, phrase-based translation models tend to use a *log-linear* framework to model the phrase translation probabilities.

2.4.2 The Log-linear Model

A log-linear model expresses the probability of a translation $P(E|F)$ as a combination of features $x_i(E|F)$ that characterize some aspect of the translation of sentence F into E. If we have N features, then the log-linear translation model is:

$$P(E|F) = \frac{1}{Z} \prod_{i=1}^{N} \alpha_i^{x_i(E|F)} \tag{2.3}$$

Here Z is a normalizing constant and α_n is the weight assigned to feature $x_n(E|F)$. In the context of statistical machine translation, we are interested in finding the sentence $\hat{E}$ which maximizes $P(E|F)$, so we can rewrite equation 2.3 as:

$$\hat{E} = \underset{E}{\operatorname{argmax}} \; \frac{1}{Z} \prod_{i=1}^{N} \alpha_i^{x_i(E|F)} \tag{2.4}$$

$$= \underset{E}{\operatorname{argmax}} \prod_{i=1}^{N} \alpha_i^{x_i(E|F)} \tag{2.5}$$

$$\tag{2.6}$$

If we choose only two weights and features, and set them relative to the language model $P(E)$ and the translation model $P(F|E)$:

$$\begin{aligned}x_1(E|F) &= \log_{\alpha_1} P(E) \\ x_2(E|F) &= \log_{\alpha_2} P(F|E)\end{aligned}$$

Then we see that the fundamental equation of machine translation (Equation 2.2)

is a special case of the log-linear model:

$$\begin{aligned}\hat{E} &= \underset{E}{\text{argmax}} \prod_{i=1}^{N} \alpha_i^{x_i(E|F)} \\ &= \underset{E}{\text{argmax}} \ \alpha_1^{x_1(E|F)} \cdot \alpha_2^{x_2(E|F)} \\ &= \underset{E}{\text{argmax}} \ P(E) \cdot P(F|E)\end{aligned}$$

Even in this simple case, the log-linear model is more flexible than the noisy-channel model because it allows the language model and the translation models to have varying weights so one can compensate for the other. In practice, a log-linear system is likely to express the language model and translation model probabilities as a composition of several features. The value of $P(E)$ might be the aggregate score of several different language models, and $P(F|E)$ could have different features for information that is expected to be important for translation, such as the likelihood of reordering elements of the translation.

2.4.3 Log-Linear, Phrase-Based Translation Models

As described in Equation 2.6, the most likely translation sentence $\hat{E}$ in the general log-linear framework is:

$$\hat{E} = \underset{E}{\text{argmax}} \prod_{i=1}^{N} \alpha_i^{x_i(E|F)} \tag{2.7}$$

An advantage of the log-linear model can be seen when taking logs of both sides of equation 2.3 and then simplifying the notation:

$$\begin{aligned}\log p(E|F) &= -\log Z + \sum_{i=1}^{N} x_i(E|F) \cdot \log \alpha_i \\ &\approx \sum_{i=1}^{N} x_i(E|F) \cdot \lambda_i\end{aligned}$$

The probability of a translation is now the sum of the feature probabilities, so a single feature of zero will not skew the translation probability. Furthermore, scaling the various features is done by setting the feature weights λ_i such that $\sum_i \lambda_i = 1$, which is what allows the use of the previously-mentioned EM algorithm to estimate the probabilities for the phrase-based model. We can now rewrite equation 2.7 as follows:

$$\begin{aligned}\hat{E} &= \underset{E}{\text{argmax}} \prod_{i=1}^{N} \alpha_i^{x_i(E|F)} \\ &= \underset{E}{\text{argmax}} \sum_{i} x_i(E|F) \cdot \lambda_i\end{aligned}$$

The phrase-based log-linear model described by Koehn *et al.* (2003) includes several other features to select the most likely translation. For example, the French phrases f_i are sequential within F, but the English phrases $e_1, e_2, \ldots e_I$ might need rearranging to form a grammatical sentence. The reordering probability is called *distortion*, and is treated as an explicit feature in the log-linear model instead of buried in the word alignment probability of the IBM models in a word-based system. The distortion d of English phrase i measures the distance between the French phrases that translate to the the current English phrase and the one before it. Another feature ω, is introduced to calibrate the length of the output sentence, as the models otherwise have a tendency to produce shorter sentences.

The values for these parameters are estimated with Minimum Error-Rate Training (MERT), a procedure which implements the EM training step (Och 2003). MERT operates by using a precalculated language model and set of probabilistic alignments, and then optimizing the weights for the features to maximize the overall system's translation score. The most common metric for statistical machine translation is BLEU (Papineni *et al.* 2002), which compares the words in a translated sentence with one or more reference translations. In this manner, the translation model's effectiveness for machine translation is often judged indirectly by its effect on the entire system's output.

Several other phrase-based translation models have been proposed, as by Venugopal *et al.* (2003) and Chiang (2005). These also use information from a word-aligned corpus to extract phrase translation parameters, but they vary in how the phrase translation lexicon is extracted. They have no widespread efficient implementations yet, so we used the currently accepted state-of-the-art system with the log-linear phrase-based translation model described in Koehn *et al.* (2003) as the framework for this thesis. The following chapters describe our work to improve the language model component.

Chapter 3

Statistical Language Modeling

This chapter is an introduction to statistical language models. A statistical language model is a probabilistic way to capture regularities of a particular language, in the form of word-order constraints. In other words, a statistical language model expresses the likelihood that a sequence of words is a fluent sequence in a particular language. Plausible sequences of words are given high probabilities whereas nonsensical ones are given low probabilities.

We briefly mentioned statistical language models in Chapter 2; here we explain them in more detail, with the goal of motivating the use of factored language models for statistical machine translation. Section 3.1 explains the n-gram model and associated smoothing techniques that are the basis for statistical language modeling. A linguistically-motivated approach to language modeling, probabilistic context-free grammars, is briefly described in section 3.2. Section 3.3 details the language model and smoothing scheme used in this thesis: Factored Language Models with Generalized Parallel Backoff. Section 3.4 mentions a useful method for generating factored language models. Finally, Section 3.5 describes the metrics for evaluating statistical language models.

3.1 The N-gram Language Model

The most widespread statistical language model, the n-gram model, was proposed by Jelinek and Mercer (Bahl *et al.* 1983) and has proved to be simple and robust. Much like phrase-based statistical machine translation, the n-gram language model has dominated the field since its introduction despite disregarding any inherent linguistic properties of the language being modeled. Language is reduced to a sequence of arbitrary

symbols with no deep structure or meaning– yet this simplification works.

The n-gram model makes the simplifying assumption that the n^{th} word w depends only on the history h, which consists of the $n-1$ preceeding words. By neglecting the leading terms, it models langage as a Markov chain of order $n-1$:

$$\Pr(w|h) \triangleq \Pr(w_i|w_1, w_2, \ldots, w_{i-1}) \approx \Pr(w_i|w_{i-N+1}, \ldots, w_{i-1})$$

The value of n trades off the stability of the estimate (i.e. its variance) against its appropriateness (i.e. bias) (Rosenfeld 2000). A high n provides a more accurate model, but a low n provides more reliable estimates. Despite the simplicity of the n-gram language model, obtaining accurate n-gram probabilities can be difficult because of data sparseness. Given infinite amounts of relevant data, the next word following a given history can be reasonably predicted with just the maximum likelihood estimate (MLE):

$$P_{MLE}(w|h) = \frac{Count(h,w)}{Count(h)} \tag{3.1}$$

The *Count* function simply measures the number of times something was observed in the training corpus. However, for any sized corpus there will be a value for n beyond which n-grams occur very infrequently and thus cannot be estimated reliably. Because of this, trigrams are a common choice for n-gram language models based on multi-million-word corpora. Rosenfeld (1996) mentions that a language model trained on 38 million words marked one third of test trigrams, from the same domain, as previously unseen. By comparison, this thesis uses the Europarl corpus, which is a standard parallel text, yet contains only 14 million words in each language. Rosenfeld (1996) furthermore showed that the majority of observed trigrams only appeared once in the training corpus.

Because the value of n tends to be chosen to improve model accuracy, direct MLE computation of n-gram probabilities from counts is likely to be highly inaccurate. Various smoothing techniques have been developed to compensate for the sparseness of n-gram counts, thereby improving the n-gram model's estimates for previously unseen word sequences.

3.1.1 Smoothing Techniques

The maximum likelihood estimate (Equation 3.1) predicts the next word based on the relative frequency of word sequences observed in the training corpus. However, as

we just mentioned, it is unsuitable for statistical inference because of data sparseness. Most words are uncommon, and thus many n-grams containing sequences of these words are unlikely to be seen in any corpus. The maximum likelihood estimate assigns a zero probability to these unseen events. Because the probability of a sentence is calculated as the product of the probabilities of component subsequences, these errors propagate and produce zero probability estimates for the sentence (Manning and Schütze 1999).

We therefore need better estimators that can allow for the possibility of sequences that did not appear in the training corpus. These procedures, called discounting or smoothing methods, work by decreasing the probability of observed events and allocating this probability mass to previously unseen events.

Many different smoothing techniques have been developed in the last twenty years of language modeling research, so we only describe a representative subset of them: Maximum Likelihood Estimate discounting via Good-Turing Estimation, Linear Interpolation, and Backoff Models.

3.1.1.1 Good-Turing Estimation

Good-Turing Estimation is is used in statistical language modeling to set aside some probability mass for estimating the likelihood of previously unseen words or n-grams. This Good-Turing estimation method for determining probability estimates for events is based on the assumption that their distribution is binomial. This seems a bit strange, as neither words nor n-grams have a binomial distribution.

However, if we group words by their frequency in the corpus, judged by the number of times they are seen, then we can assume a binomial distribution over the number of words in each group. This takes advantage of the informal Zipf's Law, which claims that the frequency of a word in a language is inversely related to its rank when words are sorted by frequency. Good-Turing estimation then uses the number of words with each frequency (that is, the count of counts) to adjust the maximum likelihood estimate for each of these groups of words that appear the same number of times. If there are N_r words in an N-word corpus that each appear r times, then the Good-Turing adjusted frequency r^* for these words is:

$$r^* = (r+1)\frac{E(N_{r+1})}{E(N_r)}$$

$E(N_r)$ is the expected value of N_r. In practice, Good-Turing estimation is generally done only for relatively rare events, where r is at most 5. In these cases, we can

substitute the empirically observed value N_r for the expected $E(N_r)$. The Good-Turing probability estimate for each of these words is thus the frequency over the corpus size:

$$P_{GT} = \frac{r^*}{N}$$

Good-Turing estimation reserves a probability mass of $E(N_1)/N$ for unseen events, but it does not provide any guidance for what to do next. One could give all previously unseen word sequences the same probability, but this is not accurate. Other smoothing methods examine the unseen n-grams themselves to more accurately estimate the likelihood of the entire sequence.

3.1.1.2 Simple Linear Interpolation

Linear Interpolation is one method for assigning probabilities to previously unseen events. It tackles the sparseness problem of an n-gram model by reserving some weight for lower-order n-gram models. The smaller n-grams, while less informative, are more likely to have been observed in the training corpus than the full-size word sequence. Based on the observation that an (n-1)-gram model is less sparse than an n-gram model, we can estimate the probability of an n-gram by combining the n-gram model with all the lower-order models into another probability model, as follows:

$$\begin{aligned} P_{LI}(w_n|w_1,\ldots,w_{n-1}) &= \lambda_1 P_1(w_n) + \\ &\quad \lambda_2 P_2(w_n|w_{n-1}) + \ldots + \\ &\quad \lambda_n P_n(w_n|w_1,\ldots,w_{n-1}) \end{aligned}$$

where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$. An arbitrary number of probability models could be combined into one language model, each with their own scaling weight, but our example is the most basic form. Setting the λ_i weights is not a straightforward manual task, and is generally done via the application of an expectation maximization (EM) algorithm, as described in Section 2.3.2.

3.1.1.3 Katz Backoff

Like linear interpolation, a Katz Backoff n-gram procedure uses the lower-order n-gram models to estimate the probability of word sequences. However, backoff only uses one of the models' probability estimates at a time. The notion of backing off to a more reliable though less accurate probability estimate is central to the factored language models we will be exploring in this thesis.

Proposed by Katz (1987), the backoff probability P_{BO} of a previously seen n-gram is the maximum likelihood estimate from equation 3.1, times a discounting factor d in order to reserve probability mass for unseen n-grams. The discount d depends on the history; frequently-observed sequences have more accurate likelihood estimates and so are discounted less. In practice, d is often the Good-Turing estimator described in section 3.1.1.1.

In the following model, the backoff strategy is triggered when the language model encounters a previously unseen trigram, as there is insufficient data for the MLE to be accurate:

$$P_{BO}(w_n|w_{n-2},w_{n-1}) = \begin{cases} (1-d_{w_{n-2},w_{n-1}}) \cdot P_{MLE}(w_n|w_{n-2},w_{n-1}) \\ \quad \text{if } \mathit{Count}(w_n,w_{n-1},w_{n-2}) > 0 \\ \alpha(w_{n-2},w_{n-1}) \cdot P_{BO}(w_n|w_{n-1}) \\ \quad \text{otherwise} \end{cases}$$

In general, the maximum-likelihood estimate of the n-gram is replaced with a probability derived from a lower-order (n-1)-gram obtained by dropping the oldest word from the n-gram. If this (n-1)-gram also has not been seen in the training corpus, then the second most distant word is dropped to produce an (n-2)-gram. The process continues until reaching some lower-order sequence of words that has been seen in the training data, or until reaching the uniform distribution over words. The probability of the n-gram is then estimated by multiplying the probability of the lower-order word sequence by a discounting factor α between 0 and 1. This is to ensure that only the probability mass set aside by the discounting step is distributed to the lower-order n-grams that are estimated by backing off.

3.1.1.4 General Linear Interpolation

Linear interpolation and backoff can be combined into a single smoothing strategy, called General Linear Interpolation. In simple linear interpolation, the λ weights for the lower-order models are fixed numbers. In general linear interpolation, the λ weights are a function of the history, allowing further differentiation between events that are unseen because they are rare and ones that are unseen because they are implausible. Combining k probability models into one language model with general linear interpolation would look like this:

$$P_{GLI}(w|h) = \sum_{i=1}^{k} \lambda_i(h) P_i(w|h)$$

Like simple linear interpolation, the weights are constrained by $0 \leq \lambda_i(h) \leq 1$ for all h and $\sum_i \lambda_i(h) = 1$. The weights are usually not set based on each specific history; finding $\lambda_{(w_{n-2},w_{n-1})}$ for each (w_{n-2}, w_{n-1}), per model, would exacerbate data sparseness. Rather, histories are bucketed, either grouped by their frequency– the count-of-counts used in Good-Turing estimation– or more elaborate methods. The λ weights for these buckets of histories are then set during language model training by the expectation maximization (EM) algorithm, just as in simple linear interpolation.

Ordinary Katz backoff can be seen as a special, though impractical, case of general linear interpolation:

$$\begin{aligned} P_{GLI}(w_n|w_{n-2},w_{n-1}) &= \lambda_{1(w_{n-2},w_{n-1})} \cdot P_1(w_n) + \\ &\quad \lambda_{2(w_{n-2},w_{n-1})} \cdot P_2(w_n|w_{n-1}) + \\ &\quad \lambda_{3(w_{n-2},w_{n-1})} \cdot P_3(w_n|w_{n-2},w_{n-1}) \end{aligned}$$

Because Katz backoff only utilizes one model score at a time, all but one of the lambda weights for a particular history are zero. The remaining weight, which gets a value of 1, is the λ corresponding to the lower-order model that would have been chosen by the backoff strategy for this history.

3.2 Syntactic Language Models

Research into smoothing methods, such as the ones in the previous section, has significantly improved the performance of n-gram language models, but the basic limitations remain. N-gram language models cannot handle long-range information such as might be encoded in syntax, for example, and the most useful size of the n-gram is limited by the corpus size.

A different approach to language modeling for statistical machine translation, based on Probabilistic Context-Free Grammars (PCFGs), integrates syntactic tools into the system. Syntax shows how words in a sentence group and relate to each other; a PCFG is a probabilistic model for the tree-like hierarchical structure of sentences in a language. The grammar is basically a set of rules for expanding nodes in a tree (in order to eventually derive the leaf nodes), and a corresponding set of probabilities for the rules. These rule probabilities are estimated from a training corpus of parsed sentences, called a treebank. After training, the PCFG can be used to generate likely parses for plain sentences.

A syntax-based statistical machine translation system might use a probabilistic context-free grammar for tree-to-string translation (Charniak *et al.* 2003). In this framework, translation is done by parsing the source sentence, performing some operations on the parse tree (reorder or insert nodes), and then translating the tree's leaf nodes in their new sequence. This method allows for consistent reordering of larger phrases, preserving grammaticality in the output.

The probabilistic context-free grammars are used first to generate parse hypotheses for the sentence to be translated, then to prune and rearrange the hypotheses down to a single tree, and lastly to translate the actual words at the tree leaf nodes. The PCFGs act as both translation model and language model, and the end result is similar to the noisy-channel model of Section 2.3.

Syntax-based models might be well-suited to statistical machine translation because they explicitly use an induced grammar to guide translation, which should improve fluency. This is a very promising approach, but still young and not as generally-applicable as an n-gram -based language model. This is because syntactic language models require a parser for the source language, which in turn requires having a treebank in order to estimate model parameters. As briefly mentioned in Section 2.2, treebanks are expensive and time-consuming resources to construct, and they only exist for a few languages.

Syntax-based language models have only been tested on very constrained machine translation tasks. Charniak *et al.* (2003), for example, reported results on only 350 sentences, with no punctuation, and fewer than 16 chinese characters in length. While the probabilistic context-free grammars improved translation quality relative to a trigram language model baseline, the overall performance was weak. Translation systems that use syntactic language models are not yet robust enough to handle more demanding translation tasks.

3.3 Factored Language Models

Factored Language Models (FLMs) are a newer language modeling technique: more sophisticated than n-grams, but more widely applicable than the syntax-based models and without the onerous requirement of a treebank. Factored language models do not reduce the complexity of an n-gram model, but they use a richer set of conditioning possibilities to improve performance. These models are capable of incorporating some amount of linguistic information, such as semantic classes or parts of speech, but un-

like in PCFG models, this information is not required and can be added to the model as available. The appeal of factored language models for machine translation is this option to layer linguistic information in the corpus, when available, on top of the power of statistical n-gram language models.

3.3.1 Notation

A Factored Language Model (FLM) considers a word as a collection of features or factors, one of which may be the actual surface form of the word. As described by Kirchhoff *et al.* (2003), a word w is a bundle or vector of K (parallel) factors such that

$$w \equiv \{f^1, f^2, \ldots f^K\} = f^{1:K}$$

The notation for the factored language model representation of a probability model over a sentence of T words, each with K factors, is:

$$P(w_1, w_2, \ldots, w_T) = P(f_1^{1:K}, f_2^{1:K}, \ldots, f_T^{1:K}) = P(f_{1:T}^{1:K})$$

Factors of a word can be anything, including word classes, morphological classes, stems, roots, or any other linguistic feature, such as may be found in highly inflected languages (Bilmes and Kirchhoff 2003). The surface form of a word can be a factor of itself, so the probabilistic language model can be over both words and their decompositional factors. For example, if we have part-of-speech information and we would like to use it as an additional factor, then the factored representation of words would look like:

$$\begin{aligned} \textit{the} &= (\textit{“the”}, \textit{article}) \\ \textit{black} &= (\textit{“black”}, \textit{adjective}) \\ \textit{cat} &= (\textit{“cat”}, \textit{noun}) \end{aligned}$$

3.3.2 Differences Between N-gram and Factored Language Models

Factored language models operate on an underlying context window of n words, much like an n-gram language model. However, there are several ways in which a factored language model is more general than an equivalently-sized n-gram model.

As seen in Section 3.1, an n-gram language model would represent the probability of the next word w_t in a sequence as being dependent only on the previous n-1 words:

$$P(w_1, w_2, \ldots, w_t) \equiv P(w_t | w_{t-(n-1)}, \ldots, w_{t-1})$$

Using the previous example, the trigram probability of the next word being "*cat*" depends only on its word history: ("*the*","*black*").

Because factored language models are an extension of an n-gram model, they use the same Markov independence assumption. Given the history of n-1 words, the likelihood of the next word w_t, composed of factors $f_t^{1:K}$, can similarly be written as:

$$P(w_1, w_2, \ldots, w_t) = P(f_{1:t}^{1:K}) \equiv P(f_t^{1:K} | f_{t-(n-1)}^{1:K}, \ldots, f_{t-1}^{1:K})$$

Therefore, the probability of the next factored feature bundle being "*cat*",*noun*") depends on its factored trigram history: $((\text{“}the\text{”}, article)(\text{“}black\text{”}, adjective))$.

In practice, we would calculate the n-gram language model probability of a sentence of T words using the chain rule:

$$P(w_1, w_2, \ldots, w_T) = \prod_t^T P(w_t | w_1, \ldots, w_{t-1}) \equiv \prod_t^T p(w_t | w_{t-(n-1)}, \ldots, w_{t-1})$$

A factored language model, however, does not impose a particular linear ordering on the factors in a word bundle. As such, there is not a fixed sequence of factors upon which to apply the chain rule. One possible factored language model probability for a sentence with T words could be calculated by taking the word bundles in sentence order, and the word features in the order in which they are arrayed within each bundle:

$$\begin{aligned} P(f_{1:T}^{1:K}) &= P(f_1^{1:K}, f_2^{1:K}, \ldots, f_T^{1:K}) \\ &\equiv \prod_t^T P(f_t^{1:K} | f_{t-(n-1)}^{1:K}, \ldots f_{t-1}^{1:K}) \\ &\equiv \prod_t^T \prod_k^K P(f_t^k | f_t^{1:k-1}, f_{t-(n-1)}^{1:K} \ldots f_{t-1}^{1:K}) \end{aligned}$$

Furthermore, not all available features have to be used in a factored language model. The relevant history for a word can be defined to be any subset of the available $n \cdot K$ factors. This might be useful if certain factors in the corpus are known to be unreliable, perhaps because the tools to generate them were unreliable. From these $n \cdot K$ factors that a factored language model can consider, there are 2^{nK} subsets of variables that can be used as the parent factors– the priors for the conditional probability. The factors within each subset can be permuted to form a distinct factor history. As this is the same as sampling without replacement, a set of factors can be used to represent a family of up to

$$\sum_{x=1}^{nk} \binom{nk}{x} x! = \sum_{x=1}^{nk} \frac{(nk)!}{(nk-x)!}$$

distinct factored language models. The factored language model framework thus allows the lexical context to be tailored to the most useful information in the factored corpus. Later in this chapter, we will discuss smoothing methods for estimating previously-unseen events. These are another significant way in which factored language models are more powerful than n-gram models.

3.3.2.1 Selecting Factors

The flexibility of factored language models comes from their being a generalization of earlier language model types. The simplest FLM has only one factor per word, the word itself, making it equivalent to an n-gram language model. A two-factor factored language model generalizes standard class-based language models, where one factor is the word class and the second is the word's surface form (Bilmes and Kirchhoff 2003). Furthermore, a factored language model can simulate the application of multiple n-gram based models, each of which can use various smoothing methods.

The challenge with factored language models is to find an appropriate set of factors with which to represent words, and further to find the best statistical model over these factors (Kirchhoff *et al.* 2003). Selecting factors can be done either manually, using linguistic knowledge about the language, or with a data-driven approach. Identifying the best probability model over a given set of factors can be viewed as a case of the structure learning problem for graphical models (Bilmes and Kirchhoff 2003), where the goal is to reduce the language model's perplexity.

In a directed graphical model, the graph represents a given probability model with each node in the graph corresponding to a random variable in the model (Kirchhoff *et al.* 2003). A conditional probability $P(A|B,C)$ is represented by arrows pointing from nodes B and C to node A. The parent nodes in the graph are the factors in the history h_t that are used to calculate the likelihood of the child node w_t at time t.

Figure 3.1 shows an example of a graphical model from (Kirchhoff *et al.* 2003). In this case, the factored language model estimates the likelihood of the next word in the text by using morphological factors and word stems in addition to the regular word surface forms. The underlying language model probability estimate of the likelihood of word w_t is calculated according to the model:

$$P(w_t) = P(w_t|s_t,m_t) \cdot P(s_t|m_t,w_{t-1},w_{t-2}) \cdot P(m_t,w_{t-1},w_{t-2})$$

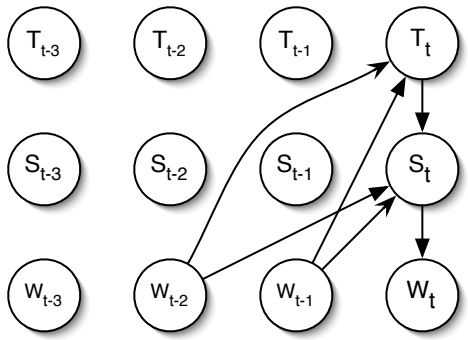

Figure 3.1: Graphical model of a factored language model over words W, stems S, and morphological factors M.

3.3.3 Generalized Parallel Backoff

As detailed in section 3.1.1.3, backoff procedures are used when the language model encounters a previously unseen word sequence. A series of language models are consulted in sequence, and the first model that can reliably estimate the likelihood of the unseen event is the one that is used to calculate the probability of the current word sequence. This relies on the backoff strategy deciding in advance the order in which the other models will be tried.

Generally, backoff procedures for an n-gram language model will go through the different n-gram models in sequence, from highest- to lowest-order, recursively dropping one word and checking the shorter n-gram when the current word sequence has not been observed enough. The process of dropping the oldest word at each step until a known word sequence is reached can be viewed in Figure 3.2 as a temporal or linear backoff path:

However, if as encouraged by a factored language model, the conditioning variables are not exclusively words, or do not clearly form a sequence, then the order in which to drop variables while backing off may not be immediately clear. For example, which contains more mutual information about the current word: the stem or the part of speech tags for the previous words? Which should be dropped first during backoff? Even for the simple case where all the parent variables are words, a more distant word may constrain the current word more than the most proximate word: in the trigram *"flying an airplane"*, the value of $P(airplane|flying)$ is probably greater than the maximum likelihood estimate of $P(airplane|an)$. By dropping one parent variable at a time, there are several other possible backoff paths in addition to the standard

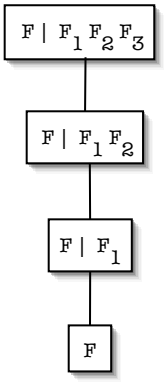

Figure 3.2

drop-oldest-first path.

For the case of a 4-gram model, the possible backoff paths over word features form the graph in Figure 3.3. The standard linear or Katz-style backoff method follows the leftmost path, shown in Figure 3.4.

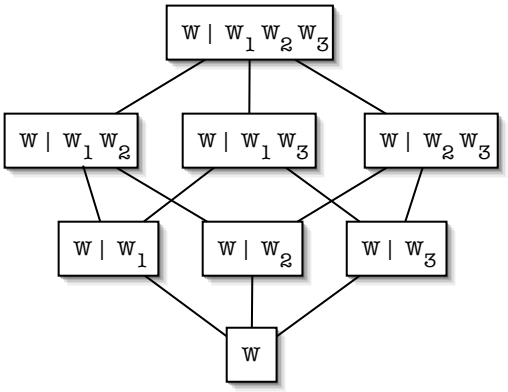

Figure 3.3

Backoff for factored language models differs in that it is not restricted to a single path through this backoff graph. A factored language model can fork when it encounters an unseen word sequence, and then estimate the likelihood based on the result of taking several different backoff paths. The factored language model can combine all the estimates derived from these backoff choices, or it can pick the most reliable of the backoff estimates to use. This procedure is called Generalized Parallel Backoff. An example of a parallel backoff scheme for a word-based factored language model is shown in Figure 3.5.

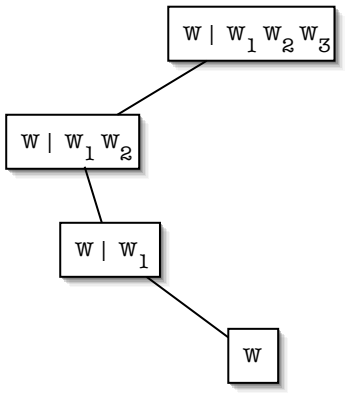

Figure 3.4

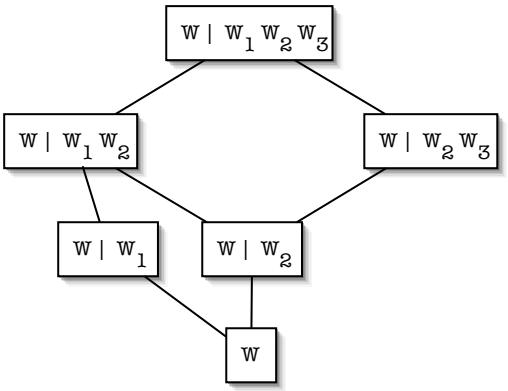

Figure 3.5

Generalized Parallel Backoff looks at the set of all possible ordering of variables contained in the backoff graph, and allows the selection of multiple paths and combining their probability estimates at runtime based on the current values of the variables (Kirchhoff and Yang 2005). Having the option during language model training to combine multiple backoff paths can only improve performance relative to an n-gram model, because the linear backoff path can always be defaulted to.

Backoff graphs are a generalization of Dupont and Rosenfeld's (1997) lattice-based language modelling, where factors are words and hierarchically-derived word classes. The probabilities for a three-factor model are calculated as follows:

$$p_{GBO}(f|f_1,f_2) = \begin{cases} (1-d_{f,f_1,f_2}) \cdot P_{MLE}(f|f_1,f_2) & \text{if } Count(f,f_1,f_2) > \tau \\ \alpha(f_1,f_2) \cdot g(f,f_1,f_2) & \text{otherwise} \end{cases}$$

This is another generalization of the standard backoff equation, using factors f_i for words w_i into Equation 3.2. The choice of the function g is where the backoff strategy is determined. For instance, if each factor f_i were the surface form of a word w_i, then $g(f,f_1,f_2) = P_{BO}(f|f_1,f_2)$ would produce regular drop-last-first backoff behavior.

There are several different options for generalized backoff backoff strategy functions g, detailed by Kirchhoff *et al.* (2003). These differ in how they select which node to back off to, and in how they combine the probabilities obtained from parallel backoff paths in order to select the final probability of the factored n-gram in question. The method recommended by Kirchhoff is the *Max Backoff-Graph Node Probability*, which chooses the backoff model to be the one that gives the factor and parent context the maximum probability. That is:

$$g(f,f_1,f_2) = P_{GBO}(f|f_{\ell_1},f_{\ell_2}) \tag{3.2}$$

where

$$(\ell_1,\ell_2) = \underset{(m_1,m_2)\in\{(1,2),(1,3),(2,3)\}}{\operatorname{argmax}} P_{GBO}(f|f_{m_1},f_{m_2})$$

This chooses the next backoff graph node to be the one that is the most likely, but this probability is itself obtained via backing off to all possible lower nodes and comparing the results. Throughout this thesis, we use Max Backoff-Graph Node Probability as the method for combining parallel backoff paths.

3.4 Training Factored Language Models

The model space for a factored language model is extremely large. Given an n-gram factored word representation with each word decomposed into k factors, there are $2^{n \cdot k}$

subsets of conditioning variables. For each set of m total conditioning factors, there are $m!$ distinct linear backoff paths. As generalized parallel backoff allows for the combination of linear backoff paths, there are actually $2^{m!}$ distinct backoff graphs that a factored language model with m parent factors can use. Each node of these backoff graphs then requires further smoothing parameters, such as Good-Turing count thresholds and the backoff procedure to use.

Exhaustive search of this model space is prohibitive, and manually specifying the edges to be used in the backoff graph and their parameters is tedious even with six conditioning factors. It would be useful to be able to automatically determine the structure of factored language models with a data-driven search procedure. However, non-linear interactions between model parameters make hill-climbing algorithms impractical.

One approach is to use a Genetic Algorithm to search a wide subset of the sample space, suggested by Duh and Kirchhoff (2004). This method works by encoding the language model's parameter combinations as a binary string and then evolving successive populations of model configurations. The better models are selected based on their perplexity on a development corpus, and used to seed the next population. Genetic algorithms do not guarantee ever finding the optimal solution, but their strength is quickly finding task-specific solutions that are good enough. We used this approach to automatically generate some of the factored language models tested with our machine translation system.

3.5 Evaluating Statistical Language Models

We are interested in improving statistical machine translation by applying more effective language models within the decoding process. Therefore, we kept the phrase-based translation model described in Chapter 2 static, in order to measure the effectiveness of different language models. To judge the language models, however, we need a metric. The standard method assesses language models by their *perplexity*, an information-theoretic measure of their predictive power. That is, how well can a language model predict the next word in a previously-unseen piece of text?

3.5.1 Information Theory Background

3.5.1.1 Entropy

Entropy is a measure of the uncertainty in a random event given the probability distribution of a random variable. Also known as *self-information*, entropy is the average uncertainty of the random variable. Given a probabilistic model of a data source that predicts the value of each event, entropy measures how much more certain the prediction would be if we added one more data point to the model. For a random variable W from which events w are selected with probability $P(w)$, the entropy of W is defined as:

$$H(W) \stackrel{\text{def}}{=} -\sum_{w \in W} P(w) \log P(w)$$

Within information theory, entropy is used to gauge the amount of information conveyed by an event. Entropy is measured in *bits*, because $H(W)$ is the lower bound on the number of bits needed to encode which value of W has been selected. For example, if there are 32 choices for the next word w, then the entropy is $\log_2 32 = 5$. Intuitively, entropy is a measure of our uncertainty about the next word: zero entropy means the next choice is purely deterministic according to our model.

We can think of a word w as being a particular value of a random variable W consisting of all words in a language. The entropy of a language L, measured over valid word sequences $W_1^n = w_1, w_2, \ldots, w_n$ in L, is:

$$H(w_1, w_2, \ldots, w_n) = -\sum_{W_1^n \in L} P(W_1^n) \log P(W_1^n) \tag{3.3}$$

3.5.1.2 Cross Entropy

However, we cannot produce a language model that is perfectly accurate– the corpus of an entire human language's valid constructions does not exist. We must therefore estimate the language's underlying probability distribution based on empirical observations. The entropy of the resulting language model is, on its own, not very interesting as we cannot tell what it means. There is a similar quantity, *cross entropy*, that relates the probability distribution of the constructed language model with the distribution of the underlying data.

The *cross entropy* of a random variable W using a probability model $M(w)$, instead

of the true distribution $P(w)$, is:

$$H(P,M) \stackrel{\text{def}}{=} -\sum_{w \in W} P(w) \log M(w) \tag{3.4}$$

$$= -\sum_{W_1^n \in L} P(W_1^n) \log M(W_1^n) \tag{3.5}$$

The second equation comes from taking the random variable w to be word sequences $W_1^n = w_1, w_2, \ldots, w_n$ in a language L. The cross entropy of a random variable according to the correct model is the same as the random variable's entropy. Also, the entropy of a random variable with distribution P is always less than or equal to its cross entropy estimated using an incorrect model M:

$$H(P) \leq H(P,M)$$

Cross entropy therefore measures how well the model fits the true distribution. When comparing language models, the one with the lowest cross entropy is generally the better one. To do so, we first must calculate the entropy of the language.

3.5.1.3 Calculating Entropy Measures

The entropy of a language model, as in Equation 3.3, will vary depending on how the length of the word sequences, as shorter ones are easier to predict. A more meaningful measure is the *per-word entropy* or *entropy rate*:

$$\frac{1}{n} H(w_1, w_2, \ldots, w_n) = -\frac{1}{n} \sum_{W_1^n \in L} P(W_1^n) \log P(W_1^n)$$

The true entropy of a language L is the entropy rate over infinitely long sequences:

$$H(L) = \lim_{n \to \infty} H(w_1, w_2, \ldots, w_n) \tag{3.6}$$

$$= \lim_{n \to \infty} -\frac{1}{n} \sum_{W_1^n \in L} P(W_1^n) \log P(W_1^n) \tag{3.7}$$

This equation can be simplified with minor abuse of the Shannon-McMillan-Breiman Theorem, also known as the Asymptotic Equipartiton Property. Under the assumption that language does not change over time, at least from the perspective of a static corpus, Equation 3.7 becomes:

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} \sum_{W_1^n \in L} P(W_1^n) \log P(W_1^n)$$

$$= \lim_{n \to \infty} -\frac{1}{n} \log P(W_1^n)$$

That is, if the word sequence is infinite *enough*, then the infinite sequence can be assumed to contain all possible sequences and thus we do not need to sum explicitly over them.

Applying this simplification to cross entropy in Equation 3.5, the cross-entropy rate of a language model with respect to the language is written as:

$$\begin{aligned} H(P,M) &= -\sum_{W_1^n \in L} P(W_1^n) \cdot \log M(W_1^n) \\ &= \lim_{n\to\infty} -\frac{1}{n} \log M(W_1^n) \end{aligned}$$

This is important because the cross entropy rate is now cast in terms of only the constructed language model probability distribution $M(W_1^n)$. This distribution is known, unlike the true language probability distribution, and thus $M(W_1^n)$ can easily be calculated from the word sequence W_1^n.

In practice, the sequence W_1^n is not infinite, but using a large enough test set the measured cross-entropy rate approximates the true cross-entropy rate. For example, if M is an n-gram language model, the cross-entropy rate is calculated by:

$$H(P,M) = -\frac{1}{n} \sum_{i=1}^{n} \log M(w_n | w_1, \ldots, w_{n-1})$$

3.5.2 Perplexity

The actual performance of language models is commonly reported in terms of *perplexity*, defined by Bahl *et al.* (1977) as:

$$PP = 2^{H(P,M)}$$

The advantage of perplexity over cross-entropy rate is that the scores ranges are larger and are therefore easier to compare. Perplexity can be thought of as the geometric average branching factor of the language, according to the language model, and is a function of both the language and the model. As a function of the model, it measures how closely the model fits the language. As a function of the language, it measures the complexity of the language.

In practice, perplexity score is used as the primary metric to evaluate language models, with a perplexity reduction of over 10% considered significant. We follow this convention, and use perplexity to evaluate the ability of factored language models to model the corpus more accurately. This score is independent of how the factored language models perform within the translation system, but it is useful for guessing

how the model should perform. Furthermore, perplexity scores are quick to calculate, unlike translation system scores. Because of this, we also use the perplexity score to select which factored language models to test in the thesis. The assumption is that lower perplexity models are better for language modeling, and thus should perform better in statistical machine translation.

In the previous chapter, we covered the basics of statistical machine translation. In this chapter, we presented the standard n-gram language model and the new factored language models, along with their respective smoothing methods. Next, we discuss the obstacles to researching language models for statistical machine translation, along with reasons why it is necessary due to the limitations of the n-gram models. We explain the advantages of factored language models with generalized parallel backoff, and outline our experiments to evaluate their effectiveness for translation.

Chapter 4

Factored Language Models in Statistical Machine Translation

In this chapter, we motivate the premise of this thesis: that factored language models are an effective approach to language modeling for statistical machine translation. We have seen already that statistical machine translation depends on language models as much as it does on translation models, as both are weighted features used for decoding. Nonetheless, research in SMT generally does not focus on language modeling, relying instead on language modeling techniques driven by the field of Automatic Speech Recognition (ASR). This is partly predicated on the assumption that what is good for ASR is also best for SMT, and partly due to the obstacles to language model research for SMT. Section 4.1 questions the utility of applying standard language models toward addressing the task-specific requirements associated with SMT. Section 4.2 describes the barriers to language model research for machine translation. Section 4.3 highlights the known problems with using n-gram language models for SMT, and Section 4.4 discusses why factored language models are a promising yet mostly untested approach for translation. Lastly, Section 4.5 outlines the experiments we conducted, as well as the motivation behind each of them.

4.1 Reasons for More Language Model Research in SMT

Statistical Language Models got their start within the field of Automatic Speech Recognition (ASR), and it is still speech recognition that drives their research today. To date, the dominant approach to language modeling for translation has been to take the best known model from speech recognition and apply it unmodified to a statistical machine

translation system. However, this is not an ideal long-term strategy, as the recognition and translation tasks are different: speech recognition is concerned with producing *the* text of a given utterance, whereas machine translation would like to produce *a* correct translation of a sentence. Both fields use a language model to gauge whether or not a hypothesis is a likely (or even plausible) chunk of language. In both cases, the language model can produce grammatically correct hypotheses that are incorrect. However, the search space for speech recognition is that of acoustic confusibility. There is a clear global maximum to work towards: there is one sentence, corresponding to the one that was uttered, whose score should be better than the others. Statistical machine translation has no such luck: there are myriad sentences translatable to the source sentence whose scores could be globally maximal.

Additionally, language model research tends to be geared towards working with English. While SMT research most often involves translating into English, interest in other languages is growing rapidly along with the amount of available data. For example, the European Union requires translation into the 20 official languages of its members. Data for these other languages is often sparser than equivalent-sized English corpora, because English is not particularly morphologically rich. That being said, morphologically complex English data, such as with biomedical corpora, is increasingly available. Off-the-shelf language model components are less effective in these sparse data conditions.

Machine translation systems should use language models that are specifically tailored to the translation task, given the differing datasets and criteria for the best output. In particular, richer smoothing is needed to combat data sparseness in translation. Linguistically-motivated language models, mentioned in Section 3.2, are not yet mature enough to replace statistical language models in widespread applications.

Factored language models with generalized parallel backoff are an intermediate step, capable of combining the power of n-gram language models with knowledge about the nature of language. Even when additional language tools are not available for factoring words in the target language, the richer smoothing of generalized parallel backoff could provide increased performance when integrated into a standard statistical machine translation decoder. This added power when dealing with scarce resources and/or highly inflected languages provides the possibility of creating language models specifically geared towards machine translation.

4.2 Barriers to Language Model Research for Statistical Machine Translation

Language model research in speech recognition is more established than in statistical machine translation. The present incarnation of statistical machine translation is significantly younger than the field of speech recognition, so it makes sense that ASR would have more mature language models and experienced researchers. However, there are other possible reasons why language models tailored for machine translation have not been heavily investigated.

4.2.1 Few Development Environments

In the first place, there is not an open development environment for researching language models for machine translation, largely because there are few publicly modifiable decoders. Language models are typically used to score hypothesis translations in two places during the translation process: during the production of translation hypothesis fragments, or as an external second-pass component that rescores the best output translations. Without being able to integrate language models into a machine translation decoder, new language modeling techniques are limited to being used for rescoring. Because of this, there is not much variation in first-pass language models used for translation.

Only a few statistical machine translation research groups have implemented their own decoder, as they can be expensive to produce. There are high-quality decoder binaries publicly available, but they are generally not modifiable due to licensing restrictions. The downloadable Pharaoh decoder is perhaps the best in its class, but it currently permits only trigram language models (Koehn 2004).

The factored language models and generalized parallel backoff methods that we use for translation were initially tested as only a rescoring technique by Kirchhoff and Yang (2005) for the reasons outlined above. While rescoring tools boost the translation output's score, the potential gains are ultimately constrained by the quality of the decoder's initial output. In this thesis, we evaluate the effectiveness of these methods when used within the decoder to generate translations, rather than to rerank output lists.

4.2.2 Hard to Correlate Results

Statistical language models are evaluated by their ability to predict the next word, measured via their perplexity (described in Section 3.5.2). While perplexity is a measurable characteristic of a language model, a language model's effect on a system is rated by its effect on the overall error rate. However, these are not directly related. As Rosenfeld (2000) asserts, the relationship is vague: "Error rates are typically non-linear and poorly understood fuctions of the language model. Lower perplexity usually results in lower error rates, but there are plenty of counterexamples in the literature."

Statistical machine translation proves no exception. The commonly accepted translation metric is BLEU, which judges a sentence on the number of n-grams that overlap with a set of reference translations (Papineni *et al.* 2002). In many instances, the BLEU score strongly correlates with human judgments. However, in some cases the correlation between BLEU score and human judgment is weak or non-existent, especially when comparing translations from very different systems (Callison-Burch *et al.* 2006).

The output sentences from a machine translation system are built by aggregating word sequences that have a high-scoring combination of translation model and language model probabilities. There is therefore not a clear breakdown of the impact of the language model score on the assembled translation. The language model's lack of direct influence at the word sequence production, aggregation and scoring steps makes it difficult to precisely correlate the perplexity of the language model used for translation with the task's error metric. Without a direct link between a language model's perplexity and the BLEU score of a translation system that uses it, guiding the construction of a language model for translation can be difficult.

4.2.3 System Tuning Can Mask Language Model Effect

The entire translation system's parameters can be tuned as a unit, adjusting the BLEU score without changing any individual component– i.e. independent of the perplexity of the language model. Within the log-linear framework of a statistical machine translation system, the language model and the translation model scaling factors are calculated relative to each other. This system parameter tuning occurs after the translation model and language models have been separately built and trained; it repeatedly tests model weight combinations with system output scores and adjusts them accordingly.

The purpose of this tuning step is to maximize the assembled system's output, and

it is able to compensate for the weakness of individual components by increasing the weight assigned to the other parameters. In particular, if the language model estimates are relatively inaccurate then the system will tend to bias the calculation of the output score towards the translation model's estimate, and vice versa.

In statistical machine translation, we compare the effectiveness of language models by the BLEU score of otherwise-identical decoder setups. However, the tuning step might attenuate the influence of a less-useful language model or overemphasize a slightly better language model, so the BLEU score of the output cannot be taken as an exact measure of the relative goodness of language models.

Given the difficulty of correlating the perplexity of a language model and the BLEU score of a translation system using that language model, there is a general question as to whether the time spent improving language models beyond the current n-gram approach would be better spent on other parts of the system.

4.3 Limitations of N-Gram-Based Language Models for Translation

N-gram language models have been a mainstay of speech recognition and other natural language processing applications for decades. Though they are simple, n-gram models are conceptually crude and the underlying Markov chains' independence assumptions are known to be insufficient to accurately represent language (Brill *et al.* 1998). Despite years of calling for improved language modeling techniques, linguistically-informed models have to date been unable to perform significantly better than n-gram models in a general setting (Rosenfeld 2000).

Rosenfeld considered n-gram models to be a "language-impoverished but data-optimal technique" whose success stymied work on knowledge-based sources. However, this is analogous to criticizing statistical machine translation for working better than, say, syntax-based translation. N-gram modeling techniques have improved steadily by the discovery of better smoothing methods for probability estimation. However, n-gram language models still have significant drawbacks, as we will explain. The next step is to combine the simplicity of n-gram language models with linguistically-motivated tools to create a hybrid language model that is more effective.

4.3.1 Model Saturation

A problem with all language models is that they saturate; there is a point at which adding new data to the training corpus produces diminishing returns. For n-gram language models, the limit is low: this figure is a couple hundred million words for bigram language models. For trigram models, it is estimated to be a few billion words(Rosenfeld 2000).

At saturation, nearly all common n-gram sequences already appear in the language model, but rare ones are still unlikely to be seen in the training corpus. However, these uncommon n-grams are the ones whose probability is the hardest to estimate correctly, so adding small quantities of new data does not correspondingly improve the language model. In fact, to achieve a constant improvement in BLEU score, the language model training corpus size needs to be doubled, as shown by the graph in Figure 4.1, based on Google Inc.'s entry in the 2005 NIST machine translation evaluation (Och 2005).

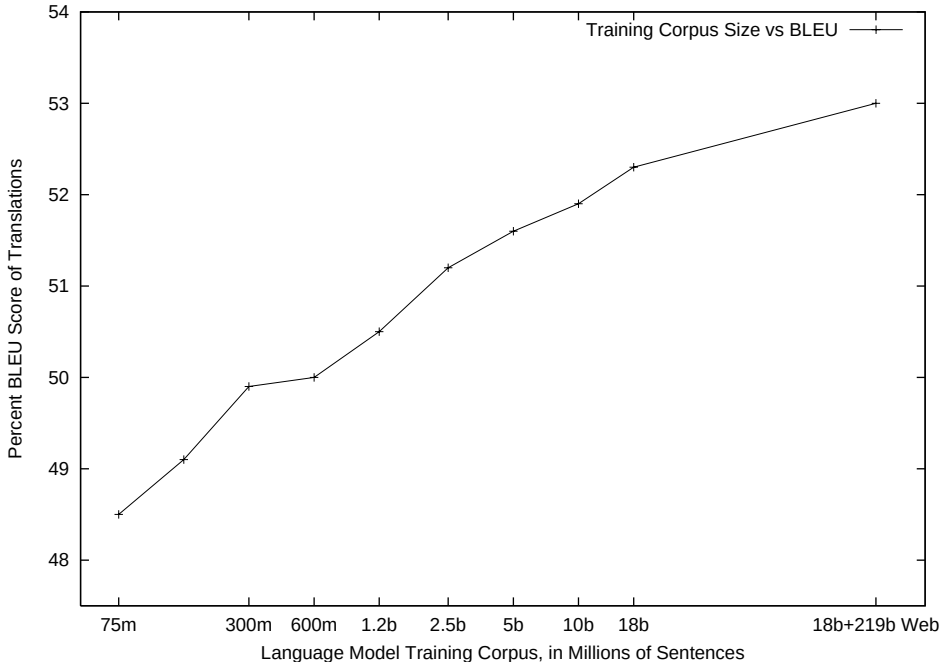

Figure 4.1: Effect of the size of the language model training corpus (in millions of sentences) on BLEU score. Doubling the corpus size is needed to maintain a constant score improvement.

Domain-relevant text is significantly more effective for language model training than unrelated text, at least when the evaluation task is on a limited domain. This can be seen in the right-most datapoint of Figure 4.1, where adding 12 times as much out-

of-domain data from the web to the corpus had the same effect on the overall score as each doubling the amount of in-domain newswire text.

4.3.2 Scaling Problems

After an n-gram model has reached saturation, improvements can only be found by either increasing the corpus size or using an (n+1)-gram model. For some languages, notably English, monolingual data is plentiful and easy to harvest. At the 2005 NIST MT evaluation, the highest-scoring translations were produced by Google, who used a 240 billion-word English corpus to train their 5-gram language model. However, as mentioned previously, only 10% of their language model's training corpus was newswire text, which was the domain of the evaluation task. With corpora that large, increasing the corpus size is neither efficient nor practical.

Larger n-gram language models are significantly sparser and have high-end space, time, and hardware computing requirements. For instance, it took Google 40 hours to translate each sentence of the evaluation dataset. Improving n-gram language models via sheer scaling is therefore an engineering challenge. While scaling up can be effective method, it is not accessible to many research groups. Computing hardware advances will bring better n-gram language model performance into the range of a larger portion of the machine translation community, but the necessary scale is still staggering.

Larger n-grams alone are not a good long-term strategy. This approach to language modeling mines corpora for improvements but, judging by the exponential amounts of data required, is an ineffective use of available resources. Even if 10-gram language models were common, probabilities for previously-unseen 10-word sequences would still be estimated based on the previous single word unless the training corpus was sufficiently large.

The languages selected for shared machine translation evaluation tasks typically represent ones that have the most bilingual digital text available for training. The Europarl corpus, popular for machine translation because it is available in 11 languages, contains only around 15 million words in each language. While monolingual data used for training language models is comparatively cheap and plentiful, there generally is still not enough to support much larger n-gram models.

Furthermore, the performance of language models trained on similarly-sized corpora in various languages will vary significantly, based on the morphological com-

plexity of the language. Inflected languages, such as Finnish or Arabic, have more word surface forms and thus fewer recurring n-grams than less-inflected ones such as English. As unknown word tokens adversely affect both language- and translation-model performance, systems trained on similarly-sized corpora in different languages can have significant variance in performance. For instance, the 2005 ACL SMT shared task average score for Finnish-English translation systems was 17.9 percent BLEU, compared to an average French-English tanslation score of 27.1 percent.

These limitations on available data and the efficiency of the n-gram model suggest that a language model that utilizes available corpora more efficiently would be of significant use to the statistical machine translation community. The requirements of such a language model would have to be more accessible and compact than an n-gram model. That is, in order to replace basic n-gram models, the new model should be flexible enough to more accurately model the target language based on the same corpus, and be able to do so for any language or domain it is applied to. Most linguistically-motivated language models, while out-performing n-gram models on specific trials, are not sufficiently reusable: parsers, part-of-speech-taggers, and other linguistic tools simply do not exist yet for many languages.

There have been prior efforts to extend n-gram language models in statistical machine translation. One such method generalizes the phrase table generated by the training algorithm for the model, to mimic n-grams that would be likely to be seen in a larger training corpus. This can be done by adding entries on a template or rule-based level. This compensates for data sparsity by automatically adding new likely entries to the probability table. For example, when a phrase such as such as “run under” is seen, add new entries corresponding to “runs under” and “running under” with a similar probability (Yang and Kirchhoff 2006). However, the model still can not estimate the probability of previously unseen n-grams that fall outside the realm of the templated phrases.

4.4 Advantages of Factored Language Models for SMT

While factored language models are also the product of speech recognition research, they are extensible enough to incorporate any obtainable data that is relevant to a machine translation task. In doing so, they bridge the gap between n-gram models and linguistically-informed language models, providing a framework for future language model experimentation. Factored language model tools are freely available, as they are

included in the popular SRI language modeling toolkit (Stolcke 2002).

By design, factored language models in their most limited form are exact duplicates of an n-gram language model. Adding generalized parallel backoff alone to the model is sufficient to provide a reduction in perplexity over the standard smoothed n-gram language model. This is because generalized parallel backoff maintains the n-gram model's smoothing techniques for linear backoff paths, while adding the ability to apply smoothing to combinations of paths.

The inclusion of additional factors for words depends on the availability of language-processing tools such as parsers, taggers, stemmers, and the like. However, the flexibility of factored language models does not require any particular factors to be used. While not available for all languages, there are many languages for which at least one such tool exists. There have been prior approaches, such as class-based and lattice-based language models which use additional linguistic information to provide perplexity reductions over an n-gram language model baseline. However, these have not been tested within the framework of statistical machine translation.

A factored language model with generalized parallel backoff can subsume the behavior of all these methods. Class-based and lattice-based language models are equivalent to a factored language model with two factors, the word and its class (Kirchhoff *et al.* 2003). Generalized phrase tables can be expressed as a factored language model with word stems as the second factor. Finally, the backoff translation models described by Yang and Kirchhoff (2006) are a factored language model that backs off linearly from words to stems.

4.4.1 Prior Work

Previous work with factored language models for statistical machine translation had been restricted to rescoring the output of a standard trigram-based translation system (Kirchhoff and Yang 2005). In their rescoring experiment, the trigram-based factored language model was shown to work as well as a 4-gram language model. While encouraging, it would seem to not be worth the extra training cost.

This may have been due to the selected trial: English is fairly morphologically simple, and French-English translation systems are one of the best-developed. In addition, the Europarl test corpus was used, which is reasonably large. A more flattering demonstration of the capabilities of factored language models would be to use a more distant language pair to translate into a morphologically complex language, such as

Finnish or Arabic, and to test the results with smaller corpora. This is because the larger the corpus, the better the underlying n-gram language model will perform, and thus the backoff and smoothing techniques of the factored language model will have less impact.

Nonetheless, in this thesis, we use Kirchhoff and Yang's (2005) setup and integrate factored language models into the standard Pharaoh decoder to show the effectiveness of these integrated factored language models on statistical machine translation. Even without rescoring, we find that the factored language models outperform trigram models. We hope this will encourage the adoption of factored language models and further investigation into tailoring them for statistical machine translation.

We have attempted to explain why language model research is not a primary focus of the statistical machine translation community. At present, improvements can still be mined from larger n-gram models, so the effort required to create new language modeling tools and tailor them specifically to machine translation is perhaps not worth it. Nonetheless, it is clear that n-gram models do not provide a viable long-term translation strategy. Factored language models present a closely-related yet more advanced approach to producing fluent translation output. Next, we present our experiments to judge these new language modeling technique's effectiveness for machine translation.

4.5 Experiments

So far, we have considered factored language models to be a cohesive set. However, their flexibility permits a great deal of diversity among the language models that can be generated with the same training corpus. With that in mind, we considered several use cases for factored language models and ran experiments for each, measuring the effect of integrating the factored language models into a statistical machine translation decoder. The language models under consideration, and the motivation behind their selection, are listed here by their labels.

trigram-lm : First, we set up the experimental baseline. This was a standard phrase-based statistical machine translation system which uses a basic trigram language model trained on the plain corpus.

match-lm : Before evaluating the performance of factored language models, we verified that our modifications to the Pharaoh decoder to support the SRI FLM tools

did not alter the decoder's functionality. The factored language model in this experiment replicates the trigram language model on the plain corpus, but using a factored language model over the plain corpus. The goal was for the two systems to have identical scores.

hand-lm : We would like to know whether the improved smoothing options alone are sufficient to improve performance in the absence of additional linguistic information. This factored language model differs from a trigram language model over the plain corpus only in that it uses generalized parallel backoff to estimate the probability of previously-unseen trigrams.

trigram-flm : We constructed a factored corpus in order to simulate the availability of additional linguistic information for the language model to use. However, adding the word tags and stems as string suffixes made the corpus sparser. This means that language models operating on the factored corpus start at a disadvantage relative to the baseline on the plain corpus. To see the extent of the handicap, we trained a second trigram language model on the factored corpus and then used it for translation.

hand-flm : The next experiment tested whether the additional linguistic information in the factored corpus alone was sufficient to improve translation performance relative to the baseline. This hand-selected factored language model backs off linearly over all the available factors, dropping the oldest one in sequence.

gaA-flm, gaB-flm, gaC-flm : Having examined the separate effects of the factored corpus, a factored model with additional linguistic information, and generalized parallel backoff with no additional factors, we put them all together. Because the parameter space for factored language models is large, we used a genetic search algorithm to select three models on the basis of their perplexity scores. In addition to evaluating the effectiveness of factored language models in machine translation it also evaluates the role of automated search methods in selecting useful models.

hand2 : This factored language model with generalized parallel backoff was constructed to mimic the general structure of the three automatically-selected models of the previous experiment, while fixing their shared failure to use all the factors available in the corpus.

We are additionally interested in how the benefits of factored language models scale with the size of the training corpus. We therefore repeated the language model experiments on each of the four smaller corpora, to simulate using factored language models on smaller domains. In the next chapter, we describe the framework we used to run the above-mentioned experiments.

Chapter 5

Experimental Framework

In this chapter, we describe the experimental setup used to test factored language models in machine translation. Section 5.1 explains the choice of data sets, as well as how they were processed. The tools used for training and using the language models are mentioned in Section 5.2, as well as the statistical machine translation system into which they were incorporated. Lastly, Section 5.3 describes the processes for system tuning and evaluation.

5.1 The Corpora

5.1.1 The Basic Corpus

Kirchhoff and Yang (2005) used the shared task from the *Association for Computational Linguistics 2005 Workshop on Building and Using Parallel Texts* as the basis for their work on using factored language models for sentence rescoring during translation. We used the same corpus to ensure our results are comparable with that of Kirchhoff and Yang. The shared task data was based on the French-English part of the Europarl corpus, and consisted of a 688,031-sentence training set, plus 2,000 development sentences and a 2,000-sentence evaluation set.

The shared task had already filtered out sentence pairs from the corpus with a length ratio greater than 9 to 1 between the translations. Also, sentences longer than 100 words (and their translations) were omitted. This is standard practice, as these types of sentence pairs tend to be difficult to word-align automatically. We cloned the corpus, with one set used as the baseline corpus for regular n-gram translation and the second one being further processed for use with the factored language model. Both the

plain and factored corpora were lowercased after construction.

We also created subsampled corpora consisting of approximately 5k, 25k, 50k, and 250k sentences from the main corpus. We used these datasets to observe how the impact of factored language models on translation scales with smaller amounts of training data.

5.1.2 The Factored Corpus

Kirchhoff and Yang selected the part of speech (POS) tag and the word stem as additional features for their factored translation model. We followed their example and used the same tools to generate the factored language model features for our factored corpus as well as the development and evaluation sets. The part of speech tags were generated by the Ratnaparkhi tagger (Ratnaparkhi 1996). We used the Porter suffixing algorithm (Porter 1980) to produce word stems from the surface forms. We also replaced the characters ":" and "-" with "**colon**" and "**dash**" in the English half of the corpus to avoid triggering the factored language model's feature separators.

Using the tagger and stemmer, we produced a corpus formatted for factored language model training, wherein each English word is expanded to a feature bundle *Tag-feature:Tag-feature:...* The three features in this corpus are the surface form of the word (W), the part-of-speech tag (T), and the word stem (S), as seen here:

```
W-are:T-vbp:S-ar W-there:T-rb:S-there W-any:T-dt:S-ani
W-comments:T-nns:S-comment W-?:T-.:S-?
```

This was a naïve way of adding factors to the corpus, but a necessary one as the tools used to align and train the corpus, such as GIZA++, do not support factoring words into features.

We also created 5k, 25k, 50k, and 250k -sentence versions of the factored corpus, to match the subsampled vanilla corpora.

5.2 Software Tools

We used a standard phrase-based statistical machine translation framework for our language model experiments, along with the following software tools:

Pharaoh : The translation models were compiled using Pharaoh, a phrase-based decoder (Koehn 2004). Pharaoh uses a multi-word phrase translation table along

with a language model to translate sentences. Output sentences are produced from left to right, using translation hypotheses selected by a beam search algorithm.

GIZA++ : The translation table was produced by GIZA++, which trains word-based translation models from aligned parallel corpora (Och and Ney 2000). GIZA++ is an implementation of the IBM models, so it induces word-level alignments between sentences.

SRILM Toolkit : The n-gram and factored language models were trained using the SRI Language Modeling Toolkit (Stolcke 2002). The improvements to support factored language models were written as part of the 2002 summer workshop at CLSP at Johns Hopkins University (Kirchhoff *et al.* 2003).

MERT : The translation system tuning was done using the Minimum-Error-Rate Training tool (Och 2003), which is an implementation of the Expectation-Maximization (EM) algorithm described in Section 2.3.2. MERT operates by using a precalculated language model and set of probabilistic alignments, and then optimizing the weights for the features to maximize the overall system's BLEU score on a reference set.

GA-FLM : We used GA-FLM, a genetic algorithm program, to guide the search for better factored language model structures as mentioned in Section 3.4. Written at the University of Washington by Duh and Kirchhoff (2004), it is an extension to the factored language model programs in the SRI Language Modeling toolkit.

5.2.1 Modifications

Prior to this thesis, factored language models had not been used for generating the output of a translation system. We used the Pharaoh machine translation decoder as the basis for this project. The decoder integrates a language model as a feature function in the log-linear model that is used to score the translation hypotheses. The normal version of the decoder judges the likelihood of a sequence of words with the standard n-gram-based language model functions within the SRILM toolkit. Therefore, in order to evaluate the use of factored language models, we modified the Pharaoh decoder to use the SRILM factored language model equivalents of the ngram tools (specifically, `fngram` and `fngram-count` instead of `ngram` and `ngram-count`).

Because we used a part-of-speech tagger and a word stemmer to mark the English side of the corpus, the translation model now proposes English translations that look like the factored corpus we previously described in Section 5.1.2, for instance: "`W-are:T-vbp:S-ar W-there:T-rb:S-there`". Just as in the factored corpus, each English word is replaced with a bundle of three features: the surface form of the word, the part of speech tag, and the word stem. This necessitated further alterations to the Pharaoh decoder's representation of a word. We also modified Pharaoh to use the SRI factored language model tools to assemble the feature bundles into a context matrix. A context matrix contains the features from the preceeding n-gram, arranged into an array that the factored language model can use to estimate the likelihood of the factored n-gram, as shown in Figure 5.1.

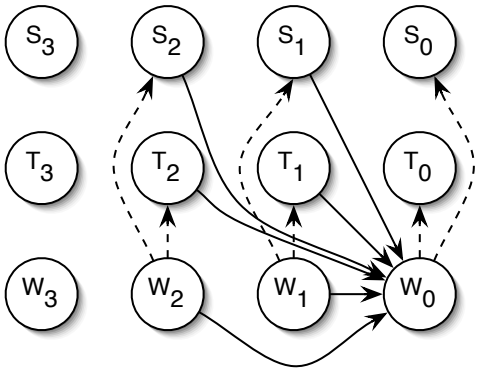

Figure 5.1: Graphical model of how the decoder represents factored n-grams. In this case, the surface form W_0 of the current word depends on the surface form W, part-of-speech tag T and stem S of the preceding two words only (solid arrows). This particular configuration of dependencies between factors is a model for $P(w_0) = P(w_0|w_2,w_1,t_2,t_1,s_2,s_1)$, which uses all the factors in the word's history. The dashed arrows show the nearly-deterministic dependency of the part-of-speech tag and stem of a word on the surface form.

5.3 Tuning and Evaluation

5.3.1 Translation Evaluation

Evaluation of machine translation systems can be done either by hand or with automatic metrics. Manual evaluation generally scores output sentences based on their

fluency as well as the accuracy of their content. While generally agreed to be the most desirable method for translation evaluation, manual translation is too time-consuming and expensive to be used to compare many different versions of a system during development and testing.

Automatic evaluation compares system translation output with reference translations from the parallel corpus. The standard automatic evaluation is BLEU, proposed by Papineni *et al.* (2002). BLEU is a precision-based metric that compares the system output and reference translations using n-gram matches between the sentences. While Callison-Burch *et al.* (2006) show that BLEU at times can have a weak correlation with human judgments on machine translation output from heterogeneous system types, it is still sufficient for tracking the results of similar systems, as is done here.

5.3.2 Tuning

The most likely translation produced by a log-linear statistical translation system is the one that maximizes the product of several weighted feature scores, as described in Section 2.4.2. The parameters of this model significantly affect translation quality, as they guide the decoder through the space of possible translations. These parameters are learned using Minimum Error Rate Training (MERT) (Och 2003). MERT iterates over a test set and considers rescoring translations with different parameter weights until the Bleu score of the test set has converged. We tuned the translation systems on a set of 500 sentences, specifically every 4^{th} sentence from the development set.

Because part of the development set is used for tuning the system, we judged the experiments on the provided evaluation dataset which is disjoint from both the training and development sets in order to prevent overfitting the system to the supplied data. As such, we used the system scores on the development set as rough estimates, and only ran the experiments on the evaluation dataset after all system tuning was complete. We report the scores on the development set as additional datapoints to verify any changes.

In this chapter, we described the tools and procedures used in setting up and evaluating the experiments for this thesis. We now present the results of the experiments listed in Section 4.5, in which we test the efficacy of different kinds of language models for statistical machine translation.

Chapter 6

Results

This chapter describes the results of experiments used to test the effectiveness of integrating factored language models with generalized parallel backoff into a statistical machine translation system. We compared the language models using two metrics: First, using the perplexity of the language model on the data sets, as described in Section 3.5.2. Second, we compared the quality of the translations produced by the decoder incorporating the language model, measured by the percent BLEU score which was introduced in Section 4.2.2.

Section 6.1 describes each of the language model experiments and their results. Section 6.2 collects all the experimental results together and summarizes them. The results of experiments on subsets of the main corpus are described in Section 6.3.

6.1 Experiments

6.1.1 Baseline Experiment: Trigram Language Model

First, we trained a standard trigram language model on the plain corpus and used it to translate the test set. This provided a reference point for measuring the performance of factored language models during decoding. We trained a trigram language model on the English half of the unmodified 688k sentence French-English Europarl corpus described in Section 5.1.1. This trigram language model uses modified Kneyser-Ney smoothing with interpolation for backoff, with Good-Turing thresholds of 3 for the tri- and bi-gram and 1 for the unigram. These are ordinary settings that ensure the histories used for estimating the trigram probability are not based on overly sparse data. This trigram language model was labeled `trigram-lm`.

To verify that the factored language model support was added to Pharaoh without changing anything else, we next trained a factored language model that mimics the behavior of a trigram language model. That is, it uses only the surface form of words and performs linear backoff by dropping the oldest word of the n-gram under consideration. This factored language model was labelled `match-lm`.

6.1.1.1 Perplexity Baseline

The perplexity score of the trigram language model `trigram-lm` was used as the baseline for evaluating language model quality in these experiments. The perplexity scores of the language model on the development and evaluation sets are shown in Table 6.1.

model	dev perplexity	eval perplexity
trigram-lm	73.01	71.85

Table 6.1: Perplexity score of standard trigram language model on the development and evaluation datasets. These are the experimental baseline perplexity scores.

The perplexity score of the basic factored language model `match-lm` appears in Table 6.2. As expected, the scores are exactly the same as the trigram language model's perplexity score on the dev and eval sets in Table 6.1.

model	dev perplexity	eval perplexity
match-lm	73.01	71.85

Table 6.2: Perplexity score on the dev and evaluation sets of a factored language model constructed to mimic the standard trigram language model.

6.1.1.2 BLEU Baseline

To evaluate translation quality, we first used the default Pharaoh parameters with the trigram language model to translate the development and training sets. This provides an untuned baseline against which to compare the performance of the translation system using other language models. In particular, the default parameters keep the language model's and translation model's weight constant. As such, changes in the BLEU score with the default parameters are due to differences in the language model and are neither assisted nor masked by the translation model.

model	dev BLEU	eval BLEU
trigram-lm	29.42	29.59

Table 6.3: Percent BLEU score of the translations of the develpoment and evaluation set produced by the unmodified Pharaoh decoder using a standard trigram language model.

Table 6.4 shows the percent BLEU scores of a translation system using the `match-lm` factored language model. The scores are nearly identical to the scores of the baseline system in Table 6.3, differing by 0.01 percent only on the evaluation set. Together with the identical perplexities of the `trigram-lm` and `match-lm` language models, this supports the claim that the modified Pharaoh decoder with factored language model support behaves just like the standard version of Pharaoh, which uses a trigram language model.

model	dev BLEU	eval BLEU
match-lm	29.42	29.58

Table 6.4: BLEU score of the translations of the development and evaluation sets produced by the Pharaoh decoder, using a factored language model to mimic the standard trigram language model. The results are nearly identical to the trigram language model baseline results.

Next, we performed minimum error-rate training (MERT) to tune the translation system as described in section 5.3.2. We used the resulting model weights to decode the development and training sets again with Pharaoh. By comparing other tuned scores against this, we test how well the factored language models perform when integrated into a state of the art statistical machine translation system. Table 6.5 shows that minimum error-rate training improved performance by 1 percent BLEU.

6.1.2 Smoothing via Generalized Parallel Backoff

With some target languages there might not be additional linguistic tools available to produce extra factors for translation. We already showed that the factored translation models can, with no extra information nor special smoothing, exactly replicate the performance of an n-gram language model. However, we are interested in seeing if

model	dev BLEU	eval BLEU
trigram-lm	29.42	29.59
trigram-lm +MERT	30.12	30.59

Table 6.5: Percent BLEU score of the translations of the development and evaluation set produced by the unmodified Pharaoh decoder using a standard trigram language model and tuned feature weights. These are the experimental baseline BLEU scores, and are 1 BLEU point better than the untuned system score on the evaluation set.

there are gains to be had from just using the parallel backoff capabilities. In order to do this, we constructed another factored language model on the plain corpus, using only the surface forms of words as factors. This language model, labeled `hand-lm` and diagrammed in Figure 6.1, differs from a regular trigram model only in its smoothing method. That is, the backoff path is not fixed; the model can drop either the last or the penultimate word to estimate the probability of the current (previously unseen) word. The most likely of the paths in the backoff graph is selected, using the Maximum Backoff-Graph Node Probability setting as described in Equation 3.2.

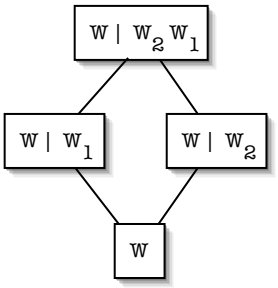

Figure 6.1: Backoff graph for the `hand-lm` language model which, like a regular trigram model, only uses the previous two words as parent features.

The perplexity scores of the `hand-lm` model on the development and evaluation sets appear in Table 6.6, with the trigram language model baseline scores shown for comparison. The use of generalized parallel backoff for smoothing reduces the perplexity of the language model, but only slightly.

We translated the development and evaluation sets with the `hand-lm` model, both before and after performing Minimum Error-Rate Training (MERT) to tune the translation system. Table 6.7 contains the percent BLEU scores, as well as those of the baseline for comparison. Using generalized parallel backoff for smoothing improves

model	dev perplexity	eval perplexity
trigram-lm	73.01	71.85
hand-lm	72.17	71.09

Table 6.6: Perplexity scores of language model with generalized parallel backoff, differing only in smoothing technique from the standard trigram language model.

the BLEU score by an absolute value of 0.26 percent. While this is not statistically significant as a percentage of the total BLEU score, in practice any change that produces an improvement above 0.1 percent BLEU is generally deemed to be worth keeping.

model	dev BLEU	eval BLEU
trigram-lm	29.42	29.59
hand-lm	29.67	29.86
trigram-lm +MERT	30.12	30.59
hand-lm +MERT	30.71	30.85

Table 6.7: Percent BLEU scores of the `hand-lm` factored language model, but with generalized parallel backoff as a different smoothing method from the standard trigram language model. Results from translating the development and evaluation sets, before and after system tuning, are shown.

The results in Tables 6.6 and 6.7 are striking, considering that no additional information has been added to the corpus– these results are due exclusively to the improved smoothing capabilities available to factored language models. This shows that factored language model tools provide an improvement even in translation tasks where a trigram language model would normally be used.

6.1.3 Trigram Language Model on the Factored Corpus

For the rest of the experiments, we used the factored corpus constructed by adding part of speech tags and word stems to the normal corpus. In the factored corpus, a phrase such as *"are there any"* expands to *"W-are:T-vbp:S-ar W-there:T-rb:S-there W-any:T-dt:S-ani"*, as described in Section 5.1.2.

Our first experiment with the factored corpus was to train a standard trigram language model on it with the same parameters as the baseline trigram language model

trained on the normal corpus. This tests whether adding the additional factor information to the corpus improves language model quality and translation scores. The perplexity scores of the new trigram model, labeled `trigram-flm`, on the factored development and evaluation sets appear in Table 6.8, along with the trigram language model baseline scores on the vanilla corpus as a comparison.

model	dev perplexity	eval perplexity
trigram-lm	73.01	71.85
trigram-flm	78.03	76.25

Table 6.8: Perplexity scores of a standard trigram language model trained on the factored corpus, compared to the baseline trigram language model trained on the plain corpus.

Adding the additional factors to the corpus, with our simplistic method, significantly worsens the perplexity of the language model.

We used the `trigram-flm` model in the unmodified Pharaoh decoder to translate the development and evaluation sets. Again, this experiment differs from the baseline setup only in that the trigram language model was trained on the factored corpus. The percent BLEU scores before and after system tuning are shown in Table 6.9, along with the baseline trigram language model score.

model	dev BLEU	eval BLEU
trigram-lm	29.42	29.59
trigram-flm	29.15	29.48
trigram-lm +MERT	30.12	30.59
trigram-flm +MERT	30.51	30.53

Table 6.9: Percent BLEU scores, before and after tuning, of the `trigram-flm` trigram language model, trained on the factored corpus, compared against the trigram language model trained on the plain corpus.

Again, adding the factors to the corpus decreased the translation scores slightly, implying that translation systems with factored language models are at a disadvantage relative to the baseline system with trigram language models of Section 6.1.1. Unfortunately, the handicap stems from the very linguistic information that is added to the corpus in order to improve performance.

This decreased performance is probably because the additional factors are written as string suffixes. Words that can be several parts of speech depending on context now appear as distinct strings, and thus the overall counts decrease for a given history. Lower counts imply less certainty in the selection of one option over another for the next word, and thus greater perplexity and greater ambiguity in translation.

6.1.4 Factored Language Model With Linear Backoff

Next, we tested a simple FLM that did not use generalized parallel backoff on the factored corpus. Having seen the effect of generalized parallel backoff on a trigram language model in Section 6.1.2, we wanted to test the performance of a simple factored language model by itself before using the new language model framework with the new backoff method for machine translation.

The language model, labeled `hand-flm` and diagrammed in Figure 6.2, treats the additional factors in the corpus as if they were extra words. It backs off linearly from word to stem to part-of-speech tag within each word bundle, and drops the oldest word bundle first. This model is similar to a 7-gram language model that decomposes word bundles such as *"W-are:T-vbp:S-ar W-there:T-rb:S-there W-any:T-dt:S-ani"* into *"are vbp ar there rb there any dt ani"*.

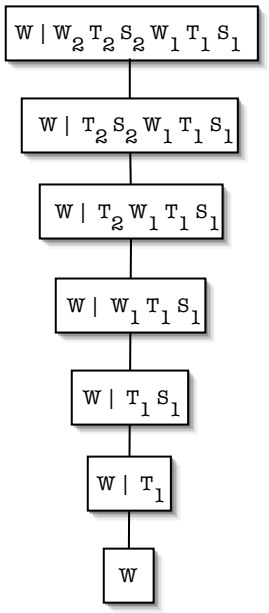

Figure 6.2: Linear backoff path for the `hand-flm` factored language model.

The perplexity scores of the `hand-flm` factored language model on the factored

development and evaluation sets appear in Table 6.10. Despite the simplistic backoff strategy, the factored language model had a significantly lower perplexity than the trigram baseline model.

model	dev perplexity	eval perplexity
trigram-lm	73.01	71.85
hand-flm	67.84	66.65

Table 6.10: Perplexity scores of a factored language model with linear backoff, `hand-flm`, compared to the baseline trigram language models trained on the plain corpus.

The translations produced using the factored language model with linear backoff, shown in Table 6.11, score higher than the baseline trigram language model output after system tuning, but not before. This is due to the minimum error-rate training procedure assigning the factored language model a relatively low weight; the tuned system is compensating for the language model's deficiency. This was only one of the 6! linear backoff strategies possible, so it is possible that a better backoff strategy exists. However, we are more interested in factored language models with parallel backoff than without.

model	dev BLEU	eval BLEU
trigram-lm	29.42	29.59
hand-flm	29.18	29.55
trigram-lm +MERT	30.12	30.59
hand-flm +MERT	30.29	30.72

Table 6.11: Percent BLEU scores, before and after tuning, of translations with the `hand-flm` factored language model with linear backoff compared against the baseline trigram language model output on the plain corpus. The factored language model performed worse than the baseline before tuning, but the system compensated enough to drive translation scores up after tuning.

6.1.5 Automatically-Generated Factored Language Models

An enormous number of factored language models, differing in the factors used, backoff graph, and smoothing parameters can be constructed from a single factored corpus, as mentioned in Section 3.3.2. Exhaustively searching the model space for the best factored language model is impractical, so we used (Duh and Kirchhoff 2004)'s GA-FLM program to explore combinations of model parameters without doing an exhaustive search.

We ran GA-FLM with the default settings (five iterations over a population size of 40) three times. Selecting the factored language model with the lowest perplexity from each run, measured on the factored corpus, produced three models: `gaA-flm`, `gaB-flm` and `gaC-flm`. Figures 6.3, 6.4 and 6.5 show the backoff graphs of each automatically-selected factored language model.

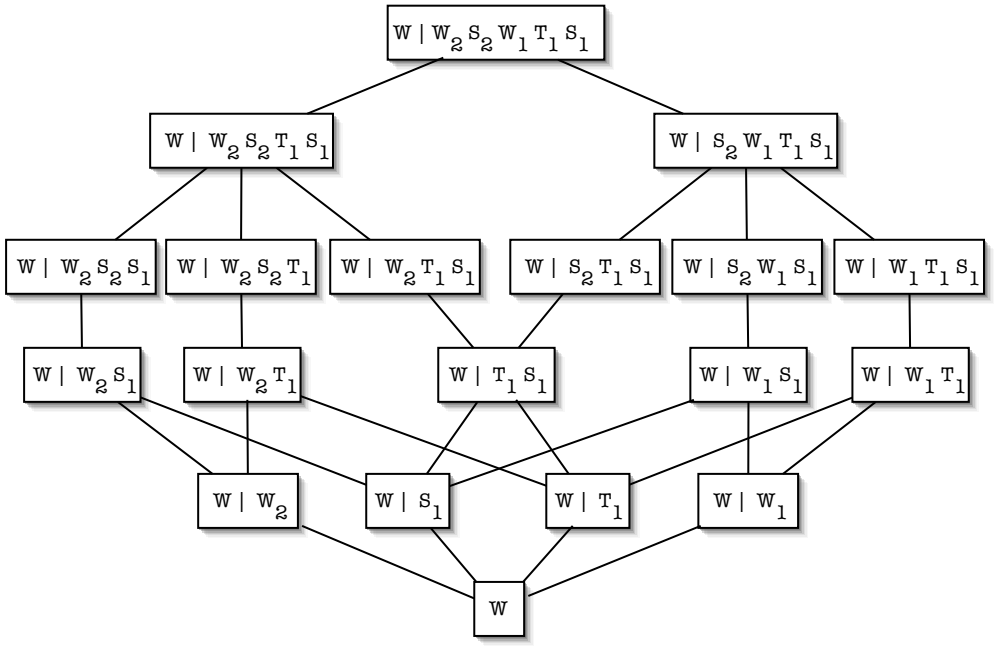

Figure 6.3: Backoff graph for the `gaA-flm` factored language model found by genetic search. This model shows the flexibility of generalized parallel backoff: in several places, the model can choose which feature to drop in order to produce the most reliable likelihood estimate of the next word. Despite that, it only uses 5 of the 6 available factors– T_2 is not considered.

Interestingly, none of the three factored language models use all six available factors (word, part-of-speech, and stem of the preceding two words) to estimate the like-

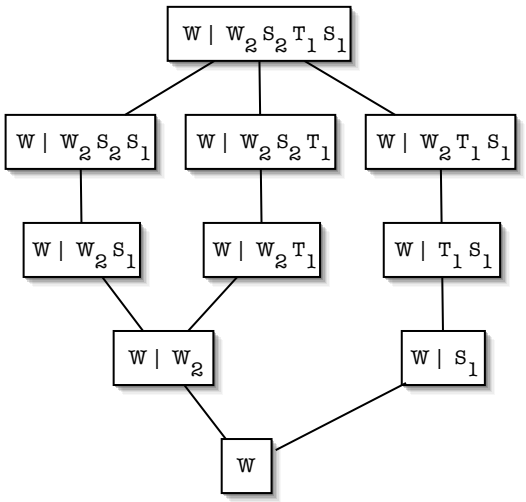

Figure 6.4: Backoff graph for the `gaB-flm` factored language model found by genetic search. This model has an extremely simple backoff graph, where the best of three parallel linear paths is used. Even fewer available factors are used– W_1 and T_2 are ignored.

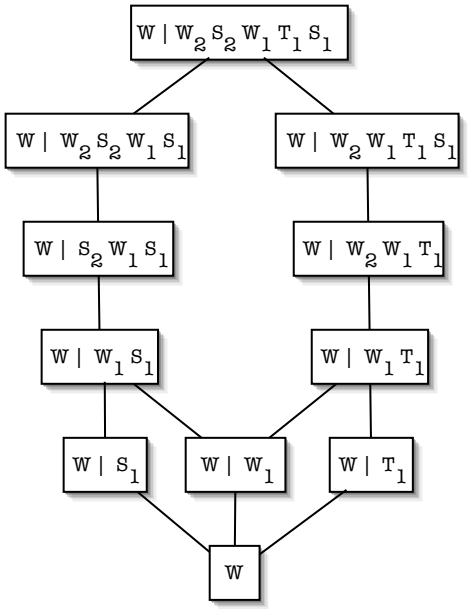

Figure 6.5: Backoff graph for the `gaC-flm` factored language model found by genetic search. This backoff graph is simple and fairly linear. Again, only 5 of the 6 available factors are used– T_2 is not considered.

lihood of the next word. This is due to the genetic search program randomly selecting the starting structure for the factored language models, so the odds of initially generating a factored language model with all six parent factors are $2^6 = 64$ to 1. Despite not using all the available information in the corpus, these three factored language models had perplexity scores that were significantly better than the baseline trigram model, as shown in Table 6.12.

model	dev perplexity	eval perplexity
trigram-lm	73.01	71.85
gaA-flm	65.64	64.45
gaB-flm	66.90	65.50
gaC-flm	63.00	62.24

Table 6.12: Perplexity scores of the factored language models found via genetic search, showing significant improvement over the perplexity of the baseline trigram language model trained on the plain corpus.

The translations generated by the modified decoder using the three automatically-generated factored language models generally scored higher than the baseline translations. Surprisingly, the model with the highest perplexity and which uses the least number of factors, `gaB-flm`, performs slightly better than the rest after tuning. The percent BLEU score increase in Table 6.13, while less dramatic than the perplexity improvements, shows that the genetic search method can produce factored language models that improve translation without manual intervention.

6.1.6 An Improved Factored Language Model with Generalized Parallel Backoff

While genetic search produces better factored language models for translation, it can get stuck exploring strange parts of the model space and thus cannot guarantee an optimal model. For example, none of the three automatically-produced language models in the previous experiment use all six available parent factors in the history (word, part of speech tag, and stem of each of the previous two words). One of them, `gaB-flm`, uses only four. The mantra of statistical approaches, *more data always helps*, suggests that the best factored language model would use all six parent factors to guide translation. At worst, if one of the parent factors has no bearing on translation, then it would

model	dev BLEU	eval BLEU
trigram-lm	29.42	29.59
gaA-flm	29.32	29.79
gaB-flm	29.44	29.75
gaC-flm	29.56	29.78
trigram-lm +MERT	30.12	30.59
gaA +MERT	30.37	30.52
gaB +MERT	30.58	30.72
gaC +MERT	30.64	30.68

Table 6.13: Percent BLEU scores from using factored language models selected by genetic search, compared to the baseline trigram language model trained on the plain corpus.

be dropped speedily during backoff. Using the backoff strategies of the automatically-generated factored language models as a guide, we constructed another factored language model that uses all available factors to predict the next word. The backoff graph for this factored language model, labeled `hand2-flm`, is diagrammed in Figure 6.6.

The perplexity scores of the `hand2-flm` factored language model on the development and evaluation sets are shown in Table 6.14. The perplexity reduction of 15% compared to a trigram model (20% against the factored baseline) verify the results of Bilmes and Kirchhoff (2003) that factored language models with generalized parallel backoff significantly improve language modeling.

model	dev perplexity	eval perplexity
trigram-lm	73.01	71.85
trigram-flm	78.03	76.25
hand2-flm	61.61	60.61

Table 6.14: Perplexity scores of the `hand2-flm` factored language model, compared to the baseline trigram language model trained on the plain corpus and the trigram language model trained on the factored corpus. These results show that factored language models can significantly reduce model perplexity.

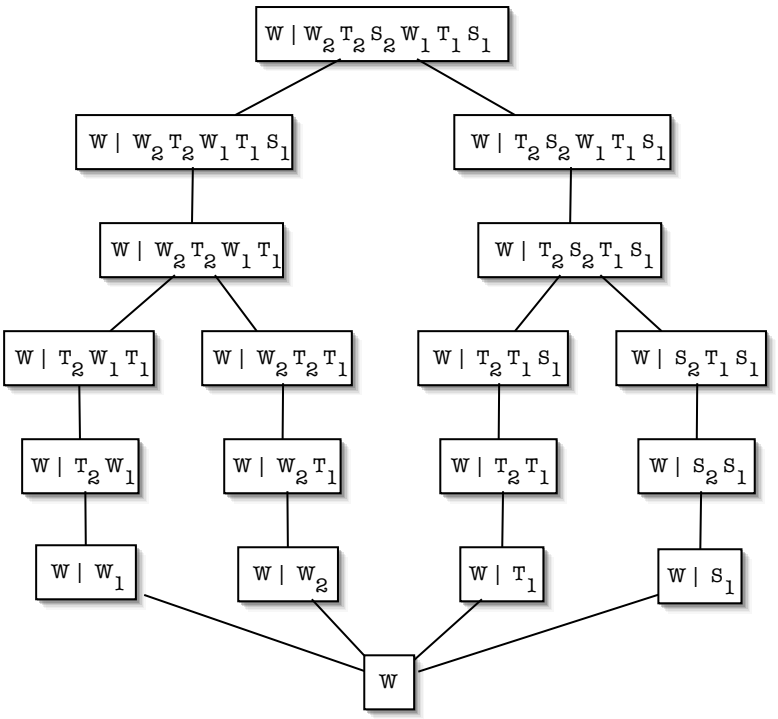

Figure 6.6: Backoff graph for the `hand2-flm` factored language model, constructed by hand but based on the best models found by genetic search. This backoff graph is fairly linear, and all of the 6 available factors are used.

When used by the Pharaoh decoder, the `hand2-flm` model produced the best-scoring tuned translations on the factored corpus, as shown in Table 6.15. These translation scores are similar to that of the trigram language model with generalized parallel backoff, `hand-lm`, copied from Table 6.7. The principal difference is that `hand-lm` uses the plain corpus, whereas `hand2-flm` might be handicapped by using the factored corpus.

However, the translation scores of `hand-lm` and `hand2-flm` are similar only after minimum error-rate training. On the untuned system, the more complex factored language model is clearly better. Inspecting the parameter files shows that minimum error-rate training assigned `hand-lm` a low weight, but gave `hand2-flm` the highest weight of all the language models we tested. This means that the similar performance of the two tuned translation systems is, in the first case, in spite of the `hand-lm` model, and in the second instance it is due to the `hand2-flm` factored language model.

model	dev BLEU	eval BLEU
trigram-lm	29.42	29.59
hand-lm	29.67	29.86
hand2-flm	29.82	30.08
trigram-lm +MERT	30.12	30.59
hand-lm +MERT	30.71	30.85
hand2-flm +MERT	30.72	30.83

Table 6.15: Percent BLEU scores, before and after system tuning, of the translations produced using the `hand2-flm` language model on the factored corpus. The translation scores of the baseline and the best-scoring language model on the plain corpus appear for comparison.

6.2 Summarized Results

The perplexity scores of all the language models appear in Table 6.16. The baseline trigram (`trigram-lm`) and the trigram model with parallel backoff (`hand-lm`), were scored on the plain corpus; the rest of the models used the factored corpus. The best factored language model reduced perplexity on the evaluation set by 11.24 points or 15.65% over the baseline trigram model. Compared against the trigram language

model trained on the factored corpus, however, the perplexity reduction on the evaluation set was 15.64 points (20.51%). These results confirm previous claims that factored language models can significantly reduce language model perplexity (Bilmes and Kirchhoff 2003).

model	dev perplexity	eval perplexity
trigram-lm	73.01	**71.85**
hand-lm	72.17	71.09
trigram-flm	78.03	**76.25**
hand-flm	67.84	66.65
ga-A	65.64	64.45
ga-B	66.90	65.50
ga-C	63.00	62.24
hand2-flm	61.61	**60.61**

Table 6.16: Perplexity scores over the development and evaluation sets of all language models tested in this thesis. The first two models were trained on the vanilla corpus, the rest on the factored corpus. The scores of the trigram language model on each corpus are highlighted, as is the best factored language model perplexity result.

The use of factored language models within a statistical machine translation system can produce better-scoring translations than a trigram language model. The percent BLEU scores of translation systems using all the language models tested in this thesis are listed in Table 6.17. Again, the first two models were trained on the vanilla corpus, the rest on the factored corpus. The best factored language model with generalized parallel backoff trained on the factored corpus is `hand2-flm` (Figure 6.6), which uses all available features in the corpus to score 0.24 percent BLEU higher than the baseline trigram model on the evaluation corpus, after system tuning. The simple factored language model trained on the plain corpus, `hand-lm`, performs marginally better (0.26 percent BLEU improvement) without using the additional features. On the untuned system, `hand-lm` improves the translation score by 0.27 percent BLEU, but texttthand2-flm doubles that, scoring 0.5 percent BLEU above the baseline.

Some of the factored language models did not improve performance after system tuning, such as `gaA-flm`. Additionally, the gains from using some factored language models to translate the development set are nearly twice as large as those on the evaluation set. From this, it seems that there is a wide range of results possible from trans-

model	dev BLEU	eval BLEU
trigram-lm	29.42	*29.59*
hand-lm	29.67	29.86
trigram-flm	29.15	29.48
hand-flm	29.18	29.55
gaA-flm	29.32	29.79
gaB-flm	29.44	29.75
gaC-flm	29.56	29.78
hand2-flm	29.82	**30.08**
trigram-lm +MERT	30.12	*30.59*
hand-lm +MERT	30.71	**30.85**
trigram-flm +MERT	30.51	30.53
hand-flm +MERT	30.29	30.72
gaA-flm +MERT	30.37	30.52
gaB-flm +MERT	30.58	30.72
gaC-flm +MERT	30.64	30.68
hand2-flm +MERT	30.72	**30.83**

Table 6.17: Percent BLEU scores of the translations of the development and evaluation sets, using each of the tuned language models tested in this thesis. The first two models were trained on the vanilla corpus, the rest on the factored corpus. The baseline scores of the trigram language model are highlighted, as are the best factored language model translation results.

lating via factored language models with generalized parallel backoff. Unfortunately, it does not seem easy to judge which factored language models will help translation quality, and which will not. Minimum error-rate training improves translation scores, but this sometimes seems to be due to biasing the system to compensate for a worse language model. At the very least, we can claim that a statistical machine translation system using factored language models with generalized parallel backoff performs at least as well as one using current n-gram language models– along with room for improvement.

6.3 Factored Language Models with Smaller Corpora

We have seen that factored language models can offer slightly better translation performance than trigram language models when trained on a large corpus. However, the amount of training data for some machine translation tasks can be limited. As the use of factored language models in machine translation is still in an exploratory stage, we thought it would be interesting to see how the models perform on smaller data sets.

To see how the performance of the factored language models change with training corpus size, we re-trained all the language models from the experiments in Section 6.1 on smaller subsets of the Europarl corpus. These mini-corpora, prepared as described in Section 5.1, were of approximately 5k, 25k, 50k and 250k sentences each.

model	5k	25k	50k	250k
trigram-lm	164.65	128.04	110.66	83.10
hand-lm	140.54	114.38	101.82	80.33
trigram-flm	176.59	139.22	120.77	89.43
hand-flm	- -	- -	99.05	76.63
gaA-flm	131.33	105.70	94.15	73.73
gaB-flm	138.00	107.06	94.76	74.39
gaC-flm	**120.93**	98.06	88.05	70.34
hand2-flm	121.93	**95.93**	**85.36**	**68.05**

Table 6.18: Perplexity scores over the evaluation set of all language models tested in this thesis, re-trained on smaller corpora. The first two language models were trained on the vanilla corpus, the rest on the factored corpus. Factored language models with generalized parallel backoff had the best perplexity scores.

The perplexity scores over the evaluation set of the language models are reported in Table 6.18. The only exception was the factored language model with linear backoff, `hand-flm`, which could not be trained on the 5k and 25k sentence corpora because some of the required history counts were zero. The perplexity scores were significantly better for the factored language models trained on the smaller corpora than for the trigram models. The factored language model with the lowest perplexity on the full-size corpus, `hand2-flm`, also had the lowest perplexity when trained on smaller amounts of factored data. In general, changing language models produced a larger relative perplexity improvement on the smaller training corpora than on the full-sized one that was used for the main experiments in this thesis (26% reduction on the 5k corpus, versus 15.64% on the full corpus).

model	5k	25k	50k	250k
trigram-lm +MERT	*18.84*	*24.04*	*26.08*	*29.65*
hand-lm +MERT	**19.27**	**24.60**	26.45	**29.76**
trigram-flm +MERT	18.30	23.70	26.02	29.36
hand-flm +MERT	- -	- -	25.57	29.15
gaA-flm +MERT	18.65	24.33	26.19	29.30
gaB-flm +MERT	18.76	24.27	26.47	29.53
gaC-flm +MERT	18.59	24.42	26.17	29.62
hand2-flm +MERT	18.94	**24.63**	**26.58**	**29.74**

Table 6.19: Percent BLEU scores of the translated evaluation set, using each tuned language model. The first two models were trained on the vanilla corpus, the rest on the factored corpus. FLMS with GPB trained on the smaller corpora improved performance by up to 0.6 percent BLEU relative to the trigram baseline.

The factored language models trained on the smaller datasets also perform well, as shown in Table 6.19. To avoid tuning thirty additional translation systems, we reused the MERT weights of the full-size language models in Section 6.1. Translation scores improved by up to 0.6 percent BLEU with a factored language model with generalized parallel backoff instead of a trigram model. These improvements are twice as large as the gains on the full-size corpus (Table 6.17), showing that factored language models provide an extra edge when translating sparse corpora. The perplexity and BLEU scores for the two best language models, `hand-lm` and `hand2-flm`, are graphed as a function of corpus size and compared to the baseline in Figures 6.7 and 6.8.

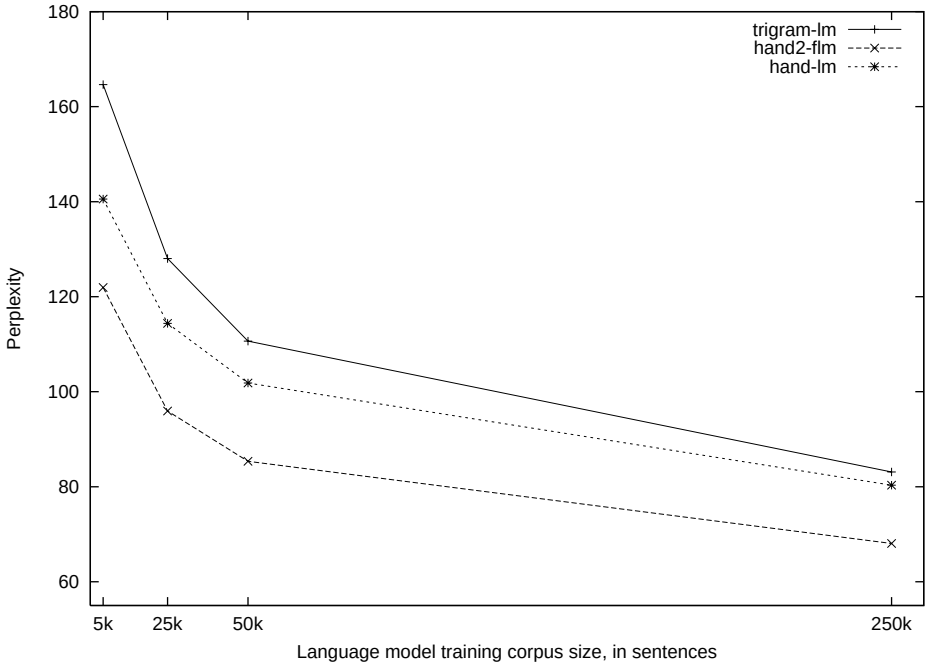

Figure 6.7: Perplexity score as a function of the training corpus size for the two best factored language models and the trigram baseline. While the FLMs are consistently better, their advantage is significantly larger for smaller corpora.

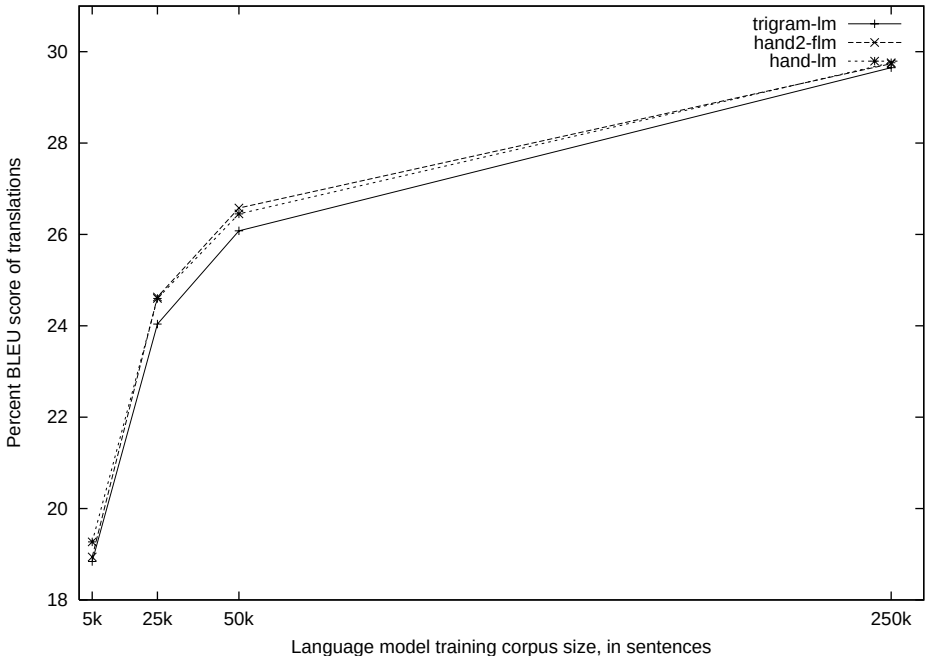

Figure 6.8: Percent BLEU translation score as a function of the training corpus size for the two best factored language models and the trigram baseline. On a tiny corpus all the models do poorly, but factored language models are clearly better overall.

6.4 Summary

We have confirmed with our experiments that factored language models with generalized parallel backoff offer significant perplexity reductions compared to standard n-gram language models. We have also shown that the factored language models improve performance when used instead of a trigram model as part of a state-of-the-art phrase-based statistical machine translation system. These relative improvements are seen even in the absence of additional linguistic information in the training corpus. This is due to the improved smoothing options available to the factored language model to use to estimate the likelihood of previously unseen word sequences. Additionally, we have shown experimentally that the advantages of using factored language models with generalized parallel backoff are more pronounced for smaller corpora. In the final chapter, we discuss the experimental results of this thesis in a broader context, drawing some additional conclusions and suggesting future directions for research.

Chapter 7

Conclusion

This thesis successfully integrated factored language modeling capabilities into a state-of-the-art machine translation system for the first time. We replicated prior results showing that these new language modeling techniques can significantly reduce model perplexity, compared to a similar-sized n-gram language model. We then demonstrated that the factored models and their associated smoothing method, generalized parallel backoff, can improve statistical machine translation performance when used instead of an n-gram language model. Finally, we also found that the relative improvements from using factored language models with generalized parallel backoff increase significantly when using smaller corpora. Our initial results show that factored language models are a new and promising direction for improving the quality of statistical machine translation.

We would have liked to compare the results from integrating factored language models into the phrase-based decoder with the work of Kirchhoff and Yang (2005), which used factored language models only as a second-pass rescoring mechanism. The training corpus and linguistic tools that we used were chosen to match theirs, to ensure the compatibility of the experimental results. However, our baseline system, using a trigram language model, significantly outperformed theirs, improving on their best two-pass system that used 4-grams for rescoring. While this does not restrict the validity of our claims about the use of factored language models, it means we cannot directly compare our results.

It is still worth mentioning that they showed an improvement of up to 1 percent BLEU when using factored language models for rescoring on the evaluation set. We show up to 0.25 percent BLEU improvement over the trigram baseline when factored language models are used within the decoder. They did not detail the factored language

models they used, only that they selected their models by optimizing on the 1st-pass oracle hypotheses. As we were using the factored language models to generate the translation system output in the first place, we could not replicate their model selection method. Another indication that our language models were different is their claim of a perplexity reduction of 8-10% over the baseline trigram model, whereas we found factored language models which reduced perplexity by over 15%.

This difference in perplexity reduction is especially surprising considering that the relative improvement of their translation score was four times larger than ours. However, this highlights one of the difficulties mentioned in Section 4.2.2 of language model research for statistical machine translation: perplexity reduction is not an effective way to guide language model selection for use in translation. The relationship between language model perplexity and the percent BLEU scores of a tuned system using the language model, using the results from Table 6.17 can be seen in Figure 7.1.

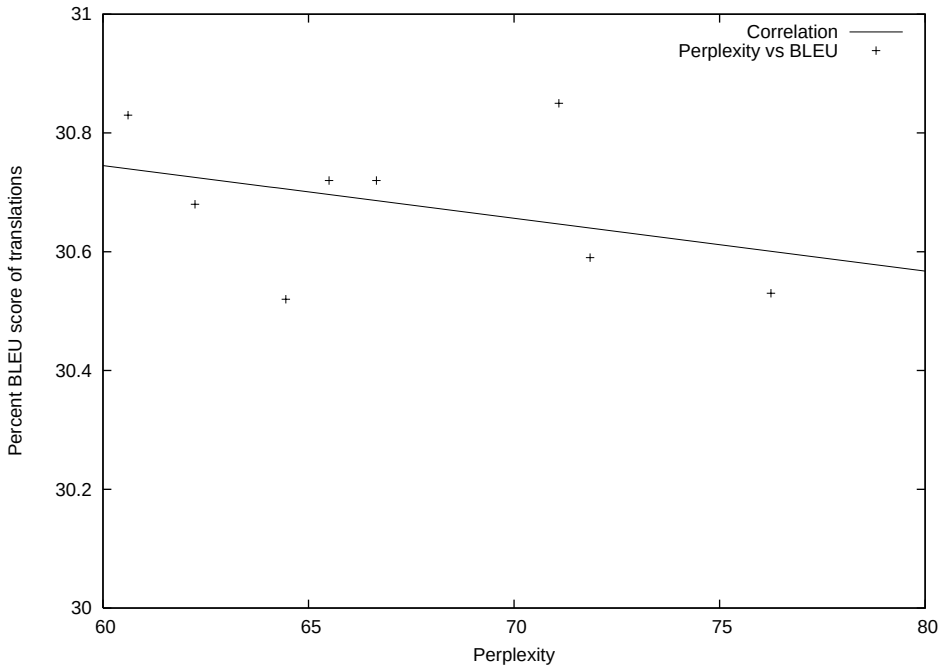

Figure 7.1: Relationship between the perplexity score of the language models and the percent BLEU score of a tuned translation system using that language model, as measured on the evaluation set.

The correlation between perplexity reduction in the language model and BLEU score increase is extremely weak: a significant perplexity reduction of 25% yields only an 0.2 percent BLEU improvement. This means that our metric for selecting language models is not tightly correlated to the overall goal of improving translation.

However, it is computationally cheap; considering 600 factored language models during a genetic search on the basis of perplexity took around 100 hours. Ideally, for each factored language model we would gauge its effectiveness by training the model, performing minimum error-rate training, and then obtaining the BLEU score of the resulting translation system from decoding the development set. This process takes approximately 60 hours per factored language model.

Surprisingly, the relationship between language model perplexity and the translation score is different for machine translation systems without minimum error-rate training, shown in Figure 7.2. On the untuned systems, a perplexity reduction of 25% corresponds to a percent BLEU increase of 0.5, two and a half times as much as on the tuned systems. While still not statistically significant, in practice an improvement of 0.5 percent BLEU is useful. This confirms another obstacle to language model research for statistical machine translation, described in Section 4.2.3: that minimum error-rate training can mask the effects of the language model on the system. Some of the experimental performance improvements were due to the tuned system compensating for the language model's deficiency.

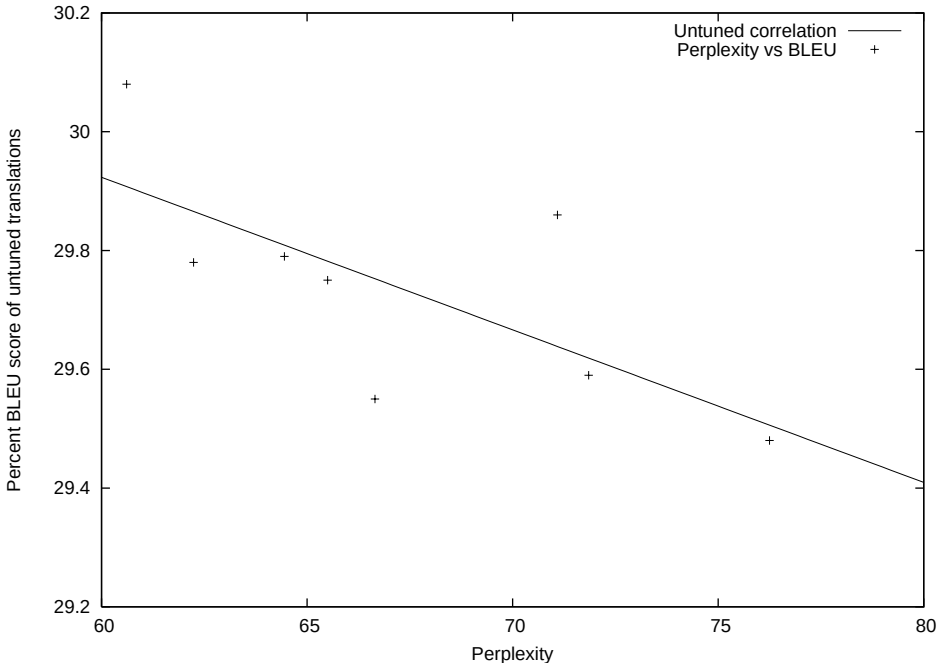

Figure 7.2: Relationship between the perplexity score of a language model and the percent BLEU score of a untuned translation system using that language model, as measured on the evaluation set.

There are two related paths for extending the work in this thesis. The first one is to devise an efficient method for selecting factored language models. The genetic search approach works, but it is a little too randomized. Ideally, it could be replaced with a method which is better informed about the model selection problem and the model parameter space so as to avoid selecting sub-optimal factored models.

Another project that would make factored language models more useful is creating a factored translation system. In a factored translation model, the factors of a word are translated independently and then the target surface form is induced from the target factors. Once factored translation models are available in a decoder, then we will be able to add linguistic information to a corpus without affecting the counts of the surface forms. This means that a factored language model will not start at a disadvantage, as compared to an n-gram model, due to the factored corpus.

A factored translation system and an improved model selection method would enable factored language models to be used as seamlessly and as efficiently as n-gram language models are today.

Bibliography

BAHL, L. R., J. K. BAKER, F. JELINEK, and R. L. MERCER. 1977. Perplexity – a measure of the difficulty of speech recognition tasks. Journal of the Acoustical Society of America 62.S63. Supplement 1.

BAHL, LALIT R., FREDERICK JELINEK, and ROBERT L. MERCER. 1983. A maximum likelihood approach to continuous speech recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5.179–190.

BILMES, JEFF A., and KATRIN KIRCHHOFF. 2003. Factored language models and generalized parallel backoff. In HLT-NAACL 2003: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, 4–6, Edmonton, Canada. Association for Computational Linguistics.

BRILL, ERIC, RADU FLORIAN, JOHN C. HENDERSON, and LIDIA MANGU. 1998. Beyond N-grams: Can linguistic sophistication improve language modeling? In ACL 1998: Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics, ed. by Christian Boitet and Pete Whitelock, 186–190, San Francisco, California. Morgan Kaufmann Publishers.

BROWN, PETER, JOHN COCKE, STEPHEN DELLA PIETRA, VINCENT DELLA PIETRA, FREDERICK JELINEK, JOHN LAFFERTY, ROBERT MERCER, and PAUL ROOSSIN. 1990. A statistical approach to machine translation. Computational Linguistics 16.79–85.

BROWN, PETER, VINCENT DELLA PIETRA, STEPHEN DELLA PIETRA, and ROBERT MERCER. 1993. The mathematics of statistical machine translation: parameter estimation. Computational Linguistics 19.263–311.

BROWN, PETER, JENNIFER LAI, and ROBERT MERCER. 1991. Aligning sentences in parallel corpora. In ACL 1991: Proceedings of the 29th Meeting of the Association for Computational Linguistics, 169–176, Berkeley. University of California.

CALLISON-BURCH, CHRIS, MILES OSBORNE, and PHILIPP KOEHN. 2006. Re-evaluating the role of bleu in machine translation research. In EACL 2006: Proceedings the Eleventh Conference of the European Chapter of the Association for Computational Linguistics.

CHARNIAK, EUGENE, KEVIN KNIGHT, and KENJI YAMADA. 2003. Syntax-based language models for machine translation. In Proceedings of MT Summit IX.

CHIANG, DAVID. 2005. A hierarchical phrase-based model for statistical machine translation. In ACL 2005: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, 263–270, Ann Arbor, Michigan. Association for Computational Linguistics.

DEMPSTER, A., N. LAIRD, and D. RUBIN. 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 1.1–38.

DUH, K., and K. KIRCHHOFF. 2004. Automatic learning of language model structure. In COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics.

DUPONT, P., and R. ROSENFELD, 1997. Lattice based language models.

GALE, WILLIAM A, and KENNETH W CHURCH. 1991. A program for aligning sentences in bilingual corpora. In ACL 1991: Proceedings of the 29th Meeting of the Association for Computational Linguistics (ACL), 177–184, Berkeley. University of California.

KATZ, SLAVA M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing ASSP-35.400–401.

KIRCHHOFF, K., J. BILMES, S. DAS, N. DUTA, M. EGAN, G. JI, F. HE, J. HENDERSON, D. LIU, M. NOAMANY, P. SCHONE, R. SCHWARTZ, and D. VERGYRI. 2003. Novel approaches to arabic speech recognition: Report from the 2002 johns-hopkins workshop. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing.

KIRCHHOFF, KATRIN, and MEI YANG. 2005. Improved language modeling for statistical machine translation. In Proceedings of the ACL Workshop on Building and Using Parallel Texts, 125–128, Ann Arbor, Michigan. Association for Computational Linguistics.

KOEHN, PHILIPP. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In AMTA 2004: Proceedings of the American Machine Translation Association, 115–124.

——, FRANZ JOSEF OCH, and DANIEL MARCU. 2003. Statistical phrase-based translation. In HLT-NAACL 2003: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

MANNING, CHRISTOPHER D., and HINRICH SCHÜTZE. 1999. Foundations of statistical natural language processing. Cambridge, Massachusetts: The MIT Press.

OCH, FRANZ JOSEF. 2003. Minimum error rate training in statistical machine translation. In ACL 2003: Proceedings of the 41st Meeting of the Association for Computational Linguistics, 160–167.

——. 2005. In MT Workshop presentation for the Google Inc. entry in the 2005 NIST evaluation.

——, and HERMANN NEY. 2000. Improved statistical alignment models. In ACL 2000: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, 440–447. Association for Computational Linguistics.

PAPINENI, KISHORE, SALIM ROUKOS, TODD WARD, and WEI-JING ZHU. 2002. Bleu: a method for automatic evaluation of machine translation. In ACL 2002: Proceedings of 40th Meeting of the Association of Computational Linguistics, 311–318.

PIERCE, JOHN R., and JOHN B. CARROLL. 1966. Language and machines: Computers in translation and linguistics. Washington, DC, USA: National Academy of Sciences/National Research Council.

PORTER, MARTIN F. 1980. An algorithm for suffix stripping. Program 14.130–137.

RATNAPARKHI, ADWAIT. 1996. A maximum entropy model for part-of-speech tagging. In Emnlp 1996: Proceedings of the conference on empirical methods in natural language processing, 133–142. Association for Computational Linguistics.

ROSENFELD, RONALD. 1996. A maximum entropy approach to adaptive statistical language modelling. Computer Speech and Language 10.187–228.

——. 2000. Two decades of statistical language modeling: Where do we go from here? Proceedings of the IEEE 88.1270–1278.

SHANNON, CLAUDE. 1948. A mathematical theory of communication. Bell System Technical Journal 27.379–423 and 623–656.

STOLCKE, A., 2002. Srilm – an extensible language modeling toolkit.

VENUGOPAL, ASHISH, STEPHAN VOGEL, and ALEX WAIBEL. 2003. Effective phrase translation extraction from alignment models. In ACL 2003: Proceedings of the 41st Meeting of the Association for Computational Linguistics, 319–326.

WEAVER, WARREN. 1949/1955. Translation. In Machine translation of languages, ed. by William N. Locke and A. Donald Boothe, 15–23. Cambridge, MA: MIT Press. Reprinted from a memorandum written by Weaver in 1949.

YAMADA, KENJI, and KEVIN KNIGHT. 2001. A syntax-based statistical translation model. In ACL 2001: Proceedings of the 39th Meeting of the Association for Computational Linguistics, 523–530.

YANG, MEI, and KATRIN KIRCHHOFF. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In EACL 2006: Proceedings the Eleventh Conference of the European Chapter of the Association for Computational Linguistics.