Discriminative Feature-Rich Modeling for Syntax-Based Machine Translation

Kevin Gimpel

CMU-LTI-12-014

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Noah A. Smith (chair), Carnegie Mellon University
Jaime G. Carbonell, Carnegie Mellon University
David Chiang, University of Southern California, Information Sciences Institute
Stephan Vogel, Qatar Computing Research Institute
Eric P. Xing, Carnegie Mellon University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

Fully-automated, high-quality **machine translation** promises to revolutionize human communication. But as anyone who has used a machine translation system knows, we are not there yet. In this thesis, we address four areas in which we believe translation quality can be improved across a large number of language pairs.

The first relates to **flexible tree-to-tree translation modeling**. Building translation systems for many language pairs requires addressing a wide range of translation divergence phenomena (Dorr, 1994). Recent research has shown clear improvement in translation quality by exploiting **linguistic syntax** for either the source or target language (Yamada and Knight, 2001; Galley et al., 2006; Zollmann and Venugopal, 2006; Liu et al., 2006). However, when using syntax for *both* languages ("tree-to-tree" translation), syntactic divergence hampers the extraction of useful rules (Ding and Palmer, 2005; Cowan et al., 2006; Ambati and Lavie, 2008; Liu et al., 2009a). Recent research shows that using *soft* constraints can substantially improve performance (Liu et al., 2009a; Chiang, 2010; Zhang et al., 2011; Hanneman and Lavie, 2011). Recently, Smith and Eisner (2006a) developed a flexible family of formalisms that they called **quasi-synchronous grammar** (QG). QG treats non-isomorphic structure softly using features rather than hard constraints. While a natural fit for syntactic translation modeling, the increased flexibility of the formalism has proved challenging for building real-world systems. In this thesis, we present the first machine translation system based on quasi-synchronous grammar.

Relatedly, we seek to **unify disparate translation models**. In designing a statistical model for translation, a researcher seeks to capture intuitions about how humans translate. This is typically done by specifying the form of translation **rules** and learning them automatically from large corpora. The current trend is toward larger and increasingly-intricate rules. Some systems use rules with flat phrase mappings (Koehn et al., 2003), while others use rules inspired by linguistic syntax (Yamada and Knight, 2001). Neither is always better than the other (DeNeefe et al., 2007; Birch et al., 2009; Galley and Manning, 2010). In this thesis, we build a system that unifies rules from these two categories in a single model. Specifically, we use rules that combine **phrases** and **dependency syntax** by developing a new formalism called **quasi-synchronous phrase dependency grammar**.

In order to build these models, we need **learning algorithms that can support feature-rich translation modeling**. Due to characteristics of the translation problem, machine learning algorithms change when adapted to machine translation (Och and Ney, 2002; Liang et al., 2006a; Arun and Koehn, 2007; Watanabe et al., 2007; Chiang et al., 2008b), producing a breed of complex

learning procedures that, though effective, are not well-understood or easily replicated. In this thesis, we contribute a new family of learning algorithms based on minimizing the **structured ramp loss** (Do et al., 2008). We develop novel variations on this loss, draw connections to several popular learning methods for machine translation, and develop algorithms for optimization. Our algorithms are effective in practice while remaining conceptually straightforward and easy to implement.

Our final focus area is the use of syntactic structure for translation when linguistic annotations are not available. Syntax-based models typically use automatic parsers, which are built using corpora of manually-annotated parse trees. Such corpora are available for perhaps twenty languages (Marcus et al., 1993; Buchholz and Marsi, 2006; Petrov et al., 2012). In order to apply our models to the thousands of language pairs for which we do not have annotations, we turn to **unsupervised parsers**. These induce syntactic structures from raw text. The statistical NLP community has been doing unsupervised syntactic analysis for years (Magerman and Marcus, 1990; Brill and Marcus, 1992; Yuret, 1998; Paskin, 2002; Klein and Manning, 2002, 2004), but these systems have not yet found a foothold in translation research. In this thesis, we take the first steps in using unsupervised parsing for machine translation.

Acknowledgments[1]

First I thank my advisor Noah Smith. From Noah I learned to pursue challenging research questions, to think beyond the next conference deadline, and to leave things better than how I found them. I appreciate his emphasis on conducting research that is clean and technically sound, especially in my results-oriented area. Noah was patient with me when I would spend time on side-projects and was always willing to brainstorm with me about new ideas. In fact, most of our meetings were spent talking about ideas outside or beyond this thesis. Now when I pursue a research problem, mentor a junior student, prepare a talk, or write a paper, I see that much of my thinking has been shaped by his mentorship and his example. I feel extremely fortunate to have benefited from so much of both over the years.

I also thank the rest of my thesis committee. Jaime Carbonell helped me to take a broad view in my dissertation work and drew my attention to connections that I hadn't seen. Eric Xing encouraged me to relate my work (and machine translation in general) to standard structure prediction formulations in machine learning. Stephan Vogel and David Chiang kept me honest about experimentation and asked many good questions throughout. David's research also inspired several parts of this thesis. I thank other faculty at CMU for sharing their opinions on my work and for many other interesting conversations: William Cohen, Bob Frederking, John Lafferty, Alon Lavie, Lori Levin, and Bryan Routledge.

I am grateful to have been part of an enthusiastic and brilliant research group. I am humbled to join the company of my fellow Ph.D. alumni from the ARK: Mike Heilman, Shay Cohen, Dipanjan Das, and André Martins. Collaboration with them was always enjoyable and illuminating. I also enjoyed and benefited from interacting with many other group members over the years, especially Waleed Ammar, Cari Bader, David Bamman, Victor Chahuneau, Chris Dyer, Daniel Mills, Behrang Mohit, Brendan O'Connor, Nathan Schneider, Yanchuan Sim, Mengqiu Wang, Tae Yano, and Dani Yogatama. Special thanks to those who I had the pleasure of mentoring on small projects: Victor Chahuneau, Naomi Saphra, Lily Scherlis, Shil Sinha, and Dan Tasse.

I enjoyed technical (and non-technical) conversations with many other students at CMU, including Abhaya Agarwal, Amr Ahmed, Jaime Arguello, Shilpa Arora, Nguyen Bach, Jon Clark,

[1]The material in this dissertation is based upon work supported by the National Science Foundation under grants 0836431, 0844507, 0915187, and CAREER grant 1054319, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, and Sandia National Laboratories through a graduate fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the aforementioned organizations.

Michael Denkowski, Jacob Eisenstein, Jeff Flanigan, Qin Gao, Greg Hanneman, Kenneth Heafield, Mahesh Joshi, Anagha Kulkarni, Ben Lambert, Frank Lin, Matt Marge, ThuyLinh Nguyen, Narges Razavian, Avneesh Saluja, Partha Talukdar, Reza Zadeh, Joy Zhang, Bing Zhao, Le Zhao, and Andreas Zollmann. I thank several LTI staff members for their attentiveness to so many details: Mary Jo Bensasi, Dana Houston, Krista McGuigan, Kelly Widmaier, and Stacey Young.

I benefited greatly over the years from conversations with others in the research community, especially Michael Auli, Phil Blunsom, Chris Callison-Burch, Yin-Wen Chang, Colin Cherry, John DeNero, Markus Dreyer, Jason Eisner, Joseph Keshet, Kevin Knight, Zhifei Li, Adam Lopez, Daniel Marcu, David McAllester, Adam Pauls, Matt Post, Chris Quirk, Joe Reisinger, Sasha Rush, David Smith, and Val Spitkovsky. Thanks to Aravind Joshi, Mitch Marcus, and Martha Palmer for introducing me to computational linguistics at Penn and to Dan Rudoy for introducing me to machine learning at MIT Lincoln Labs.

I thank Shankar Kumar for hosting me during an internship at Google in 2009, as well as all those I worked with at Google, especially Wolfgang Macherey, Ashish Venugopal, and my fellow interns. I am extremely grateful to Sandia National Laboratories for a graduate fellowship which supported my research during the last two years. Most of the work in this thesis was done while I was supported by this fellowship. I also thank the ARCS Foundation for a scholarship during my first three years at CMU.

I thank friends in Pittsburgh and elsewhere for their friendship and support over the last six years: Aurora, Christina, Dave, Dave & CC, Eunice, Eve & Richard, Grace, Heather, Jon & Libby, Joya, Julie, June, Kyle & Maggie, Marta, Nikos, Sarah, Smitha, Viv, and Yuko. It is no exaggeration to say that I could not (or would not) have done this without them. I also thank the ballroom dance and Catholic communities in Pittsburgh, especially Frs. Stephen, Joshua, and Michael and all my friends from the Oratory. Thanks also to Sts. André, Joseph, Michael, and Augustine, among many others.

Finally I thank my family: Mom, Dad, Laura, Scott, Julia, and Baby Gimps. They were always interested in and supportive of my work, and they also provided me many opportunities to escape it for a weekend. Their constant love and encouragement since the beginning of this long journey has made its completion possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fully-automated, high-quality **machine translation** promises to revolutionize human communication. But as anyone who has used a machine translation system knows, we are not there yet. Translation systems still make mistakes that result in disfluent or nonsensical output, even for closely-related languages. Systems are complex, and there are many potential sources of error. Sometimes the problem is due to flawed modeling of the translation process, but it might also be due to errors made by linguistic analysis tools or by the statistical techniques used to train the system from data. In this thesis, we seek to improve translation quality by addressing all of these potential sources of error.

While machine translation has a rich history dating back to the mid-twentieth century, modern systems trace their roots to the statistical approach developed by Brown et al. (1990). A **statistical machine translation** (SMT) system learns how to translate based on examples from human translators. An example is a sentence in one language paired with its translation in another. Translation **rules** are extracted from these examples. A rule matches some part of the input sentence and emits text in the target language. Equipped with a massive collection of rules, a system translates a sentence through statistical pattern matching.

Given examples, a statistical system can be built rapidly with minimal knowledge of the languages. This has enabled machine translation for a large number of language pairs. As of this writing, Google Translate[1] supports 64 languages and 64^2 - 64=4032 distinct systems. But not all systems generate translations of the same quality. Improving translation quality across language pairs is a large research area, encompassing many related threads of inquiry.

In this thesis, we address four areas in which we believe translation quality can be improved across a large number of language pairs. We introduce them here and then go into more detail in the following sections:

- **Handling syntactic divergence** (Section 1.1). Building translation systems for many language pairs requires addressing a wide range of translation divergence phenomena (Dorr, 1994). Recent research has shown clear improvement in translation quality by exploiting

[1] `http://translate.google.com`

linguistic syntax for either the source or target language (Yamada and Knight, 2001; Galley et al., 2006; Zollmann and Venugopal, 2006; Liu et al., 2006). However, when using syntax for *both* languages ("tree-to-tree" translation), syntactic divergence hampers the extraction of useful rules (Ding and Palmer, 2005; Cowan et al., 2006; Ambati and Lavie, 2008; Liu et al., 2009a). Non-isomorphic syntactic structures result from genuine syntactic divergence between languages as well as stylistic choices by human translators (Dorr, 1994; Fox, 2002; Wellington et al., 2006; Søgaard and Kuhn, 2009). Recent research shows that using *soft* constraints in tree-to-tree translation substantially improves performance (Liu et al., 2009a; Chiang, 2010; Zhang et al., 2011; Hanneman and Lavie, 2011). Recently, Smith and Eisner (2006a) developed a flexible family of formalisms that they called **quasi-synchronous grammar** (QG). QG treats non-isomorphic structure softly using **features** rather than hard constraints. While a natural fit for syntactic translation modeling, the increased flexibility of the formalism has proved challenging for building real-world systems. In this thesis, we present the first machine translation system based on quasi-synchronous grammar.

- **Unifying disparate translation models** (Section 1.2). In designing a statistical model, a researcher seeks to capture intuitions about how humans translate. The current trend is toward larger and increasingly-intricate rules. Some systems use rules with flat phrase mappings (Koehn et al., 2003), while others use rules inspired by linguistic syntax (Yamada and Knight, 2001). Neither is always better than the other (DeNeefe et al., 2007; Birch et al., 2009; Galley and Manning, 2010). In this thesis, we build a system that unifies rules from these two categories in a single model. Specifically, we use rules that combine **phrases** and **dependency syntax** by developing a new formalism called **quasi-synchronous phrase dependency grammar**.

- **Adding features** (Section 1.3). Syntactic translation models benefit from the addition of large numbers of feature functions to better score syntactic structures (Blunsom and Osborne, 2008; Chiang et al., 2009) and to flexibly handle syntactic divergence (Smith and Eisner, 2006a; Chiang, 2010). Using large feature sets requires adapting algorithms from statistical machine learning to machine translation. Due to characteristics of machine translation, algorithms change during this adaptation (Och and Ney, 2002; Liang et al., 2006a; Arun and Koehn, 2007; Watanabe et al., 2007; Chiang et al., 2008b), producing a breed of complex learning procedures that, though effective, are not well-understood or easily replicated. In this thesis, we contribute a new family of learning algorithms based on minimizing the **structured ramp loss** (Do et al., 2008). We develop novel variations on this loss, draw connections to several popular learning methods for machine translation, and develop algorithms for optimization. Our algorithms are effective in practice while remaining conceptually straightforward and easy to implement.

- **Learning syntactic structure without annotations** (Section 1.4). Syntax-based models require automatic parsers, which are built using corpora of manually-annotated parse trees. Such corpora are available for perhaps twenty languages (Marcus et al., 1993; Buchholz and Marsi, 2006; Petrov et al., 2012). In order to apply our models to the thousands of language

pairs for which we do not have annotations, we turn to **unsupervised parsers**. These induce syntactic structures from raw text. The statistical NLP community has been doing unsupervised syntactic analysis for years (Magerman and Marcus, 1990; Brill and Marcus, 1992; Yuret, 1998; Paskin, 2002; Klein and Manning, 2002, 2004), but these systems have not yet found a foothold in translation research. In this thesis, we take the first steps in using unsupervised parsing for machine translation.

In the following sections, we discuss each of these areas in greater detail and highlight our contributions.

1.1 Handling Syntactic Divergence

Building translation systems for many language pairs requires addressing a wide range of translation divergence phenomena. Several researchers have studied divergence between languages in corpora and found it to be considerable, even for closely-related languages (Dorr, 1994; Fox, 2002; Wellington et al., 2006; Søgaard and Kuhn, 2009). To address this, researchers have begun targeting syntactic divergence by incorporating **linguistic syntax** into translation model design. The statistical natural language processing (NLP) community has developed automatic parsers that can produce syntactic analyses for sentences in several languages (Klein and Manning, 2003; Buchholz and Marsi, 2006; Nivre et al., 2007). The availability of such parsers triggered research interest in **syntax-based** statistical machine translation (Yamada and Knight, 2001).

Syntax-based translation models are diverse, using different grammatical formalisms and features. Some use a parse tree for the source sentence ("tree-to-string"), others produce a parse when generating the target sentence ("string-to-tree"), and others combine both ("tree-to-tree"). We focus on the final category in this thesis. It has proved to be a difficult modeling problem, as initial attempts at tree-to-tree translation underperformed systems that used no syntax at all (Cowan et al., 2006; Ambati and Lavie, 2008; Liu et al., 2009a). Subsequent research showed that substantial performance gains can be achieved by relaxing hard constraints imposed by source and target syntax (Liu et al., 2009a; Chiang, 2010; Zhang et al., 2011; Hanneman and Lavie, 2011). This suggests that hard constraints must be handled with care.

Hard constraints in tree-to-tree translation are implied when using **synchronous grammars**. A synchronous grammar derives two strings simultaneously, one in the source language and one in the target language. A single derivation is used for both strings, which limits the categories of divergence that can be captured. As a result, researchers have developed synchronous grammars with larger rules that model more phenomena, but typically at increased computational expense (Shieber and Schabes, 1990; Eisner, 2003; Gildea, 2003; Ding and Palmer, 2005).

We take a different approach. We build on a flexible formalism called **quasi-synchronous grammar** (QG; Smith and Eisner, 2006a). Unlike synchronous grammar, QG provides a *monolingual* grammar over translations inspired by a fixed input sentence. Therefore, arbitrary non-isomorphic structures are possible. There are few, if any, hard constraints. The model uses feature functions to softly penalize or encourage particular types of syntactic divergence. However, this

added flexibility comes with a cost, as quasi-synchronous grammar relaxes standard constraints on the output space, requiring a careful formulation to achieve computational tractability.

In this thesis, we present the first translation system based on quasi-synchronous grammar. Our initial model is presented in Chapter 4. That chapter also includes our full model based on an extension of QG; we motivate it in the next section.

1.2 Unifying Disparate Translation Models

Since the word-based translation models of Brown et al. (1990), researchers have developed models with richer rules that include larger and increasingly-complex units. Broadly, two approaches currently dominate research in translation modeling. Models that use flat phrase mappings (Koehn et al., 2003) excel at capturing local reordering phenomena and memorizing multi-word translations, such as idioms and non-compositional expressions. Models that employ syntax or syntax-like representations (Yamada and Knight, 2001; Chiang, 2005; Galley et al., 2006; Zollmann and Venugopal, 2006; Huang et al., 2006) handle long-distance reordering better than phrase-based systems (Birch et al., 2009) but often require constraints on the formalism or rule extraction method in order to achieve computational tractability. As a result, certain instances of syntactic divergence are more naturally handled by phrase-based systems (DeNeefe et al., 2007; Birch et al., 2009).

In this thesis we present a new way of combining the advantages of phrase-based and syntax-based translation models. We propose a model in which phrases are organized into a tree structure inspired by dependency syntax (Tesnière, 1959). Instead of standard dependency trees in which *words* are vertices, our trees have *phrases* as vertices. The result captures phenomena like local reordering and idiomatic translations within phrases, as well as long-distance relationships among the phrases in a sentence. Since we integrate this idea with QG, we call our model a **quasi-synchronous phrase dependency grammar**. We present our model description in Chapter 4 and give experimental results and analysis in Chapter 5.

Recent success of system combination in MT evaluations has shown how much can be gained by combining diverse approaches to translation (NIST, 2009). Our approach differs from system combination by allowing the weights for individual features of each system to be learned jointly. While others have worked on combining rules from multiple syntax-based systems (Liu et al., 2009b) or using posteriors from multiple models to score translations (DeNero et al., 2010), we take a step further by defining a new formalism to combine phrase-based and syntax-based machine translation at the modeling level.

1.3 Adding Features

When moving from hard constraints to soft constraints, the translation search space grows in size. The hope is that the enlarged search space contains many more good translations, but in fact many bad translations may be included as well. One effective solution is to add feature functions to better score the hypotheses in the enlarged space. However, the most popular learning algorithm for machine translation—minimum error rate training (MERT; Och, 2003)—is only effective for

small sets of features (Hopkins and May, 2011). Historically, therefore, SMT researchers have rarely experimented with larger feature sets. When they have, however, substantial empirical gains have been obtained, especially for syntax-based translation (Chiang et al., 2009; Chiang, 2010; *inter alia*).

To handle more features, we need alternatives to MERT. The statistical machine learning community has invented a variety of learning techniques that can be applied to tasks that predict linguistic structure (Smith, 2011). However, machine translation is different from other tasks in ways that complicate parameter learning. For example, there are typically multiple valid translations for any given input sentence. Furthermore, the one given by a human translator may not be the best one to use when training our systems. Supervised machine learning often assumes a single correct output for each input, making many standard learning algorithms unsuitable for translation. As a result, researchers have contorted popular training methods to suit the translation problem, leaving behind theoretical guarantees. This has resulted in confusion in the community about what is precisely being optimized by each approach, clouding empirical comparisons and making replication of results more difficult.

Exploring which translation to treat as the "gold standard" during learning has bred a variety of new algorithms inspired by standard machine learning techniques (Och and Ney, 2002; Liang et al., 2006a; Arun and Koehn, 2007; Watanabe et al., 2007; Chiang et al., 2008b, 2009; Eidelman, 2012). Due to the handling of this issue, many of the approaches used in practice can be shown to have strong connections to a particular learning criterion called the **structured ramp loss** (Do et al., 2008; McAllester and Keshet, 2011). In this thesis, we develop new variants of ramp loss inspired by several MT learning algorithms (Och and Ney, 2002; Smith and Eisner, 2006a; Liang et al., 2006a; Chiang et al., 2009). The ramp losses have attractive theoretical properties, support regularization, and scale well to large numbers of features.

We develop simple learning algorithms for all ramp loss variants based on well-understood optimization algorithms with theoretical guarantees. Our family of learning algorithms is a key contribution of this thesis. Experiments show that our algorithms are competitive with the state of the art across several language pairs and models. When adding a large number of features, a strong baseline—pairwise ranking optimization (Hopkins and May, 2011)—overfits to the training data, while our algorithms are stable, even showing improvements in translation quality.

Our contributions to learning in machine translation are orthogonal to the rest of the thesis. While our learning algorithms were developed with our translation model in mind, they can be used for any other model. Indeed, in our empirical comparison, the majority of our experiments focus on two widely-used statistical machine translation systems (Koehn et al., 2007; Chiang, 2007). The algorithms and experiments are presented in Chapter 3.

1.4 Learning Linguistic Structure without Annotations

Syntax-based translation requires automatic parsers. Most parsers are trained on corpora of hand-annotated parse trees, e.g., the Penn Treebank (Marcus et al., 1993). Such corpora are available for perhaps fifteen to twenty languages for certain linguistic theories (Buchholz and Marsi, 2006;

Petrov et al., 2012). The NLP community has also developed a range of techniques to build **unsupervised parsers** (Klein and Manning, 2002, 2004; Smith, 2006; Blunsom and Cohn, 2010; Naseem et al., 2010; Spitkovsky et al., 2010; Cohen, 2011). They require only raw, unannotated text in the language of interest, making them ideal for use in translation. Furthermore, they can be trained directly on the data being used to train translation systems. This is noteworthy because even supervised parsers have been shown to have significantly-reduced performance when tested on data from outside the training domain (Dredze et al., 2007; McClosky et al., 2010).

Integrating unsupervised parsers with machine translation systems offers a great deal of promise for syntax-based translation. Potentially, the improvements from syntactic modeling for high-resource languages can transfer to hundreds of other language pairs. While unsupervised parsing accuracies are still well below those of supervised parsers (Cohen, 2011), certain properties of the translation problem are well-suited to unsupervised learning of syntax. For example, unsupervised parsing can be improved if parallel text is available (Kuhn, 2004; Snyder et al., 2009), which is a prerequisite for nearly all statistical translation systems. Unsupervised parsing can be further improved if we have parallel text *and* a parser for one of the two languages (Yarowsky and Ngai, 2001; Yarowsky et al., 2001; Hwa et al., 2005; Ganchev et al., 2009; Smith and Eisner, 2009).

Furthermore, the use of unsupervised parsing for machine translation can also contribute back to research in NLP. It may be the case that parsing accuracy (measured by comparing to human annotations) is not particularly indicative of utility for machine translation or other downstream applications. We propose syntax-based machine translation as a testbed to evaluate unsupervised syntactic analyzers.

Unsupervised *shallow* syntactic analysis has been used successfully for translation modeling by Zollmann and Vogel (2011), who showed that unsupervised part-of-speech tags could be used to label the hierarchical translation rules of Chiang (2005) to match the performance of a system that uses supervised full syntactic parses. In this thesis, we take additional steps in this direction. We leverage state-of-the-art unsupervised models for deep syntactic analysis (Klein and Manning, 2004; Berg-Kirkpatrick et al., 2010), contribute improvements to unsupervised dependency parsing (Appendix B), and use the resulting parsers in our translation system (Chapter 5). In one experiment we replace an off-the-shelf supervised Chinese parser with our unsupervised parser and see no change in performance. For languages without parsers, unsupervised parsing is necessary even just to use our model, and we report performance improvements over our baseline. These initial results offer promise for researchers to apply syntactic translation models to the hundreds of languages for which we do not have manually-annotated corpora, and naturally suggest future research directions.

1.5 Thesis Statements

The key claims of this thesis follow.

- By understanding the loss functions targeted by successful approaches to learning for machine translation, we can develop learning algorithms that minimize those loss functions directly. Our resulting algorithms are conceptually straightforward, easy to implement, and

offer theoretical guarantees. In a thorough experimental comparison, we demonstrate that they achieve state-of-the-art performance in both small- and large-feature settings.

- We can gain the benefits of both phrase-based and tree-to-tree translation in one model by using **quasi-synchronous phrase dependency grammar**, showing improvements in translation quality across several language pairs.
- Unsupervised parsing can be used effectively within syntax-based machine translation, allowing us to apply syntactic translation models to languages for which supervised parsers do not exist.

1.6 Organization of the Thesis

This thesis is organized as follows.

- Chapter 2 covers relevant background material about statistical machine translation, focusing on two areas: (1) phrase-based and syntax-based translation modeling, and (2) algorithms for learning feature weights. We define notation and establish our formal framework for the remainder of the thesis. While this chapter contains no novel technical material, it is hoped to be useful as a stand-alone introduction to statistical machine translation for the machine learning or statistical natural language processing researcher.
- Chapter 3 includes our contributions to learning algorithms for statistical machine translation. We develop a family of algorithms that minimize variations of the **structured ramp loss**. We relate our algorithms to others developed for machine translation and conduct a thorough experimental comparison, showing that our algorithms are competitive with the state of the art in both small-feature and large-feature experiments, and offer greater stability. This chapter is based on Gimpel and Smith (2012a) but includes additional analysis and experimental results. Appendix A contains related material originally published in Gimpel and Smith (2010a,b).
- Chapter 4 presents our syntax-based translation model based on quasi-synchronous phrase dependency grammar. We define the model, our rule extraction procedure, and features to score phrase dependency trees and tree-to-tree configurations. We give an efficient decoding algorithm based on a coarse-to-fine strategy. This chapter contains material originally published in Gimpel and Smith (2009b, 2011a) and includes additional features and a new decoding algorithm.
- Chapter 5 presents experimental results using the model defined in Chapter 4. We conduct experiments on four language pairs using both supervised and unsupervised parsers. We analyze the translations generated by our system, showing that we improve translation quality in ways that make the meaning more clear, such as improving the translation of verbs. We also measure the speed of our decoding algorithm with various pruning parameters. This

chapter contains material from Gimpel and Smith (2011a, 2012b) and includes additional experimental results and analysis.

- Chapter 6 summarizes the contributions of this thesis and discusses future research directions. In particular, we believe that our learning contributions can expedite large-scale feature exploration for machine translation. We hope that our success in using unsupervised parsing can inspire others to apply syntax-based translation modeling to many more language pairs than has been attempted thus far.

Chapter 2

Statistical Machine Translation

In this chapter, we present background material about statistical machine translation (SMT) necessary for the remainder of the thesis. We describe the roots of SMT research in the late 1980s and chart its evolution to the present day. Along the way, three fundamental problems are highlighted: modeling, decoding, and learning. We describe commonly-used solutions for each problem to lay the groundwork for our contributions in the remaining chapters of the thesis.

While this chapter does not contain novel technical material, it does contribute a consistent formal understanding of statistical machine translation research. Modeling and decoding are presented using declarative specifications that highlight differences among popular approaches. Methods for parameter learning are described using a framework well-known in the machine learning community. This chapter may be useful as a concise introduction to statistical machine translation for researchers from a machine learning or statistical natural language processing background. Koehn (2010) covers the field in depth.

2.1 Background and Formal Framework

While machine translation research has a long history, **statistical machine translation** emerged during the late 1980s from IBM research (Brown et al., 1988, 1990). Brown et al. introduced the idea of building translation systems that learn how to translate from examples generated by human translators. They also introduced a series of statistical translation models that are still widely used.

To describe their approach, we need to define some notation. Let $\mathcal{X}$ denote the set of all strings in a source language and, for a particular $\boldsymbol{x} \in \mathcal{X}$, let $\mathcal{Y}_{\boldsymbol{x}}$ denote the set of its possible translations (correct and incorrect) in the target language. Brown et al. proposed to model the probability $p(\boldsymbol{y} \mid \boldsymbol{x})$ that a human translator will generate $\boldsymbol{y}$ from $\boldsymbol{x}$. Having modeled this probability, a statistical machine translation system should choose the best translation $\boldsymbol{y}^*$ as

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} p(\boldsymbol{y} \mid \boldsymbol{x}) \tag{2.1}$$

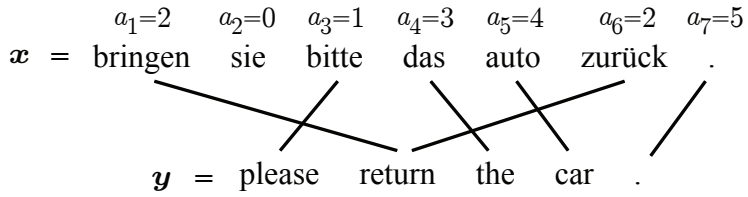


Figure 2.1: Example source and target sentences (tokenized and converted to lowercase) with word alignments. The source sentence is $\boldsymbol{x}$ and the human-generated reference translation is $\boldsymbol{y}$. The word alignment vector $\boldsymbol{a}$ indicates, for each source word, the target word that it is aligned to. Each source word is aligned to exactly one target word or is left unaligned (e.g., *sie* in this example). A target word can be aligned to multiple source words, or left unaligned.

2.1.1 Noisy Channel Models

To attack the difficult problem of building a model of $p(\boldsymbol{y} \mid \boldsymbol{x})$, Brown et al. took inspiration from **noisy channel** models used in statistical modeling for speech recognition (Bahl et al., 1983). In particular, they used Bayes' rule to factor the target quantity $p(\boldsymbol{y} \mid \boldsymbol{x})$ into the product of the **translation model** $p(\boldsymbol{x} \mid \boldsymbol{y})$ and the **language model** $p(\boldsymbol{y})$:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}}{\operatorname{argmax}}\ p(\boldsymbol{y} \mid \boldsymbol{x}) = \underset{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}}{\operatorname{argmax}}\ p(\boldsymbol{x} \mid \boldsymbol{y}) p(\boldsymbol{y}) \tag{2.2}$$

where the $p(\boldsymbol{x})$ term from the denominator is omitted since it does not change the solution of the argmax. Solving Eq. 2.2 is referred to as **decoding** (Cover et al., 1991).

The reason for casting machine translation as a noisy channel problem as in Eq. 2.2 is that it breaks down the problem into intuitive pieces. Instead of relying entirely upon a single model of $p(\boldsymbol{y} \mid \boldsymbol{x})$, the modeler can separately build the translation model $p(\boldsymbol{x} \mid \boldsymbol{y})$ and the language model $p(\boldsymbol{y})$. Intuitively, $p(\boldsymbol{x} \mid \boldsymbol{y})$ encourages faithfulness to the original text, while $p(\boldsymbol{y})$ encourages fluency. Brown et al. used smoothed n-gram models for $p(\boldsymbol{y})$, and such models are still widely used today.

When designing models for $p(\boldsymbol{x} \mid \boldsymbol{y})$, Brown et al. assumed the presence of a **latent variable** representing alignments from words in $\boldsymbol{x}$ to words in $\boldsymbol{y}$. Formally, the latent variable is a vector $\boldsymbol{a}$ of length $n = |\boldsymbol{x}|$ where, if $a_i = j$, we say that x_i is **aligned** to y_j. Alignments roughly capture translational equivalence between words. If $a_i = 0$, we say that x_i is aligned to NULL, indicating that the source word x_i has no explicit correlate in the target sentence. The definition of $\boldsymbol{a}$ implies that each source word is aligned to a single target word (or to NULL), while a target word can be aligned to multiple (or zero) source words. Figure 2.1 shows an example sentence pair with word alignment $\boldsymbol{a}$. The German word *sie* is not aligned to anything on the English side; the two words *bringen* and *zurück* are aligned to the single English word *return*.

The alignments are not observed in the data, so we refer to them as variables that are latent or

hidden.[1] Even though the alignments are unobserved, we can still estimate the parameters of the models. In practice this is done using the expectation-maximization algorithm (Dempster et al., 1977; Brown et al., 1993).

The use of latent variables is common to nearly all statistical translation models. Word alignments are one simple example, but richer models often have richer latent variables. We will discuss examples below when we describe popular translation models in Section 2.2. Regardless of the specific form of the latent variable, we will refer to it as a **derivation** and denote it $\boldsymbol{h} \in \mathcal{H}_{\boldsymbol{x}}$, where $\mathcal{H}_{\boldsymbol{x}}$ is the set of possible values of $\boldsymbol{h}$ for the input sentence $\boldsymbol{x}$. Derivations will always be coupled with translations and therefore we define the set $\mathcal{T}_{\boldsymbol{x}} \subseteq \mathcal{Y}_{\boldsymbol{x}} \times \mathcal{H}_{\boldsymbol{x}}$ of valid $\langle \boldsymbol{y}, \boldsymbol{h} \rangle$ pairs for $\boldsymbol{x}$. By "valid," we mean that the pair is explicitly licensed by a translation model. We will be more precise about $\mathcal{T}_{\boldsymbol{x}}$ later in this chapter. $\mathcal{Y}_{\boldsymbol{x}}$ is a theoretical space, independent of any particular model, while $\mathcal{H}_{\boldsymbol{x}}$ (and therefore $\mathcal{T}_{\boldsymbol{x}}$) is defined by a translation model.

The acknowledgment of latent variables suggests that we rewrite Eq. 2.2 as follows:

$$\langle \boldsymbol{y}^*, \boldsymbol{h}^* \rangle = \underset{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}}}{\operatorname{argmax}} \; p(\boldsymbol{x}, \boldsymbol{h} \mid \boldsymbol{y}) p(\boldsymbol{y}) \tag{2.3}$$

where, for the IBM models, $\boldsymbol{h} = \boldsymbol{a}$. That is, in addition to returning a translation, we also select a derivation for that translation. This is actually the equation solved in practice by Brown et al., not Eq. 2.2. We will discuss this issue in more depth when we discuss decoding in Section 2.1.3.

2.1.2 Log-Linear Models

Och and Ney (2002) generalized the noisy channel approach by moving to **log-linear models**, probabilistic models that allow the use of arbitrary features. Instead of using Bayes' rule to decompose $p(\boldsymbol{y}, \boldsymbol{h} \mid \boldsymbol{x})$ into the translation and language models, they modeled $p(\boldsymbol{y}, \boldsymbol{h} \mid \boldsymbol{x})$ directly using a log-linear model parameterized by a parameter vector $\boldsymbol{\theta} \in \boldsymbol{\Theta}$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{h} \mid \boldsymbol{x}) = \frac{\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})\}}{\sum_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}}} \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{h}')\}} \tag{2.4}$$

where $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ is a vector of feature functions on $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{h}$. When used as here for a conditional distribution, the model is typically called a **conditional log-linear model**. The noisy channel model from Section 2.1.1 is a special case with the following two features:

$$f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) = \log \tilde{p}(\boldsymbol{x}, \boldsymbol{h} \mid \boldsymbol{y})$$
$$f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) = \log \tilde{p}(\boldsymbol{y})$$

and with $\theta_1 = \theta_2 = 1$, and where these two features have their own estimated parameters. Rather than naming the parameters in these models via subscripts, we use the notation $\tilde{p}$ to generically suggest that these distributions contain a large set of parameters that are estimated from data. The

[1] There exist some corpora with manually-aligned sentence pairs (e.g., Ayan and Dorr, 2006), but they are small and only available for certain language pairs.

important thing to note is that the parameters implicit in the definition of any $\tilde{p}$ distribution are distinct from $\boldsymbol{\theta}$, which we always use to indicate the parameters (feature weights) of the log-linear model. Och and Ney (2002) added novel features beyond these, including the length of $\boldsymbol{y}$, and noted that many others are possible.

Like noisy channel models, the denominator can be dropped when decoding with a log-linear model:

$$\begin{aligned}\langle \boldsymbol{y}^*, \boldsymbol{h}^* \rangle &= \underset{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}}}{\operatorname{argmax}} \; p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{h} | \boldsymbol{x}) \\ &= \underset{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}}}{\operatorname{argmax}} \; \frac{\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})\}}{\sum_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}}} \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{h}')\}} \\ &= \underset{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}}}{\operatorname{argmax}} \; \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) \end{aligned} \tag{2.5}$$

where, as before, we maximize over both $\boldsymbol{y}$ and $\boldsymbol{h}$.

While the denominator is not needed to solve argmax problems of this form, it is needed for alternative decoding criteria (discussed in Section 2.1.3) and for certain approaches to learning $\boldsymbol{\theta}$ (discussed in Section 2.3). For example, Och and Ney (2002) maximized conditional likelihood, which requires computing the denominator in Eq. 2.4, but the learning algorithms used most frequently today ignore the denominator (Och, 2003; Watanabe et al., 2007; Chiang et al., 2008b, 2009; Hopkins and May, 2011; Cherry and Foster, 2012). When using such learning procedures and Eq. 2.5 for decoding, it becomes more appropriate to simply use the term **linear model**, since the denominator is always being ignored. We will usually use this term in the rest of this thesis.

2.1.3 Decoding

While not made explicit in their formulation, it is clear from their decoding algorithm that Brown et al. (1990) did not actually solve Eq. 2.2. They actually solved Eq. 2.3, replicated below:

$$\langle \boldsymbol{y}^*, \boldsymbol{h}^* \rangle = \underset{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}}}{\operatorname{argmax}} \; p(\boldsymbol{x}, \boldsymbol{h} \mid \boldsymbol{y}) p(\boldsymbol{y})$$

where $\boldsymbol{h} = \boldsymbol{a}$. That is, they maximized jointly over both output strings $\boldsymbol{y}$ and derivations $\boldsymbol{h}$. To actually solve Eq. 2.2, we must marginalize out $\boldsymbol{h}$ during decoding:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}}{\operatorname{argmax}} \; p(\boldsymbol{x} \mid \boldsymbol{y}) p(\boldsymbol{y}) = \underset{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}}{\operatorname{argmax}} \sum_{\boldsymbol{h} \in \{\boldsymbol{h}' : \langle \boldsymbol{y}, \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}}\}} p(\boldsymbol{x}, \boldsymbol{h} \mid \boldsymbol{y}) p(\boldsymbol{y}) \tag{2.6}$$

With a log-linear model, this problem takes the following form:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}}{\operatorname{argmax}} \sum_{\boldsymbol{h} \in \{\boldsymbol{h}' : \langle \boldsymbol{y}, \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}}\}} p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{h} | \boldsymbol{x}) = \underset{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}}{\operatorname{argmax}} \sum_{\boldsymbol{h} \in \{\boldsymbol{h}' : \langle \boldsymbol{y}, \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}}\}} \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})\} \tag{2.7}$$

However, solving optimization problems of this form is intractable when structured output spaces are used (Casacuberta and de la Higuera, 2000; Lyngsø and Pedersen, 2002; Sima'an, 2002). While

some researchers have developed approximate algorithms for solving Eq. 2.7 (May and Knight, 2006; Blunsom et al., 2008; Blunsom and Osborne, 2008; Arun et al., 2009; Li et al., 2009b), most instead maximize over both translations and derivations, as we described in previous sections and replicate below:

$$\langle \boldsymbol{y}^*, \boldsymbol{h}^* \rangle = \underset{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}}}{\operatorname{argmax}} \; \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) \tag{2.8}$$

Unfortunately, this problem is also intractable for typical translation models. Knight (1999) showed that solving Eq. 2.8 is NP-complete for the more complex of the IBM models, due to the size of the search space and weak independence assumptions of the models. Many modern translation models can be similarly shown to have intractable decoding problems (Koehn et al., 2003; Huang and Chiang, 2007), or have such high polynomial orders as to be infeasible to solve exactly in practice (Chiang, 2007; Huang and Chiang, 2007). As a result, approximate decoding algorithms abound in statistical machine translation research, and are essential to consider when designing new models.

We refer to solving Eq. 2.8 as **Viterbi decoding**, since it is analogous to finding the best complete sequence in hidden Markov models, a problem solved by the well-known Viterbi algorithm (Viterbi, 1967). Alternative decoding criteria have also been used for machine translation, such as **minimum Bayes risk decoding** (Kumar and Byrne, 2004), but in this thesis we use Viterbi decoding exclusively.

2.1.4 Translation Model Components

In the previous sections, we presented the statistical machine translation framework as introduced by Brown et al. and developed by other researchers in the years since. We now summarize and lay groundwork for the remainder of the thesis. The key components of a translation model include the following:

- A definition of $\mathcal{T}_{\boldsymbol{x}} \subseteq \mathcal{Y}_{\boldsymbol{x}} \times \mathcal{H}_{\boldsymbol{x}}$, i.e., the set of allowable translations of $\boldsymbol{x}$, coupled with allowable derivations
- A vector $\boldsymbol{f}$ of feature functions that score tuples $\langle \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{X} \times \mathcal{T}_{\boldsymbol{x}}$
- A vector $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ of feature function weights ($|\boldsymbol{\theta}| = |\boldsymbol{f}|$)

Given these components, we now identify the three fundamental problems of statistical machine translation:

- The **modeling problem** consists of defining $\mathcal{T}_{\boldsymbol{x}}$ and $\boldsymbol{f}$. The modeler defines the form of the latent derivation variable $\boldsymbol{h}$ and develops a method to populate $\mathcal{T}_{\boldsymbol{x}}$ for a given $\boldsymbol{x}$. This is where the modeler can impose constraints on the allowable translations for a given input. The feature functions $\boldsymbol{f}$ are then designed to score allowable translations.

- The **decoding problem** is that of designing an efficient algorithm to (approximately) solve Eq. 2.8. The algorithm depends on $\mathcal{T}_{\boldsymbol{x}}$ and $\boldsymbol{f}$, coupling the decoding and modeling problems. Therefore, we discuss models and decoding algorithms together in Section 2.2. Our contributions to modeling and decoding are presented in Chapter 4.

- The **learning problem** is to choose $\boldsymbol{\theta}$. In principle, learning is orthogonal to modeling and decoding, but certain learning algorithms may be better suited to certain classes of models. We discuss learning for machine translation in Section 2.3. Our novel contributions to learning in machine translation are presented in Chapter 3. What we describe as "learning" is often called "tuning" or "training" in the MT literature. In this thesis, we use these three terms interchangeably to mean the selection of $\boldsymbol{\theta}$.

In the remainder of this chapter, we discuss the modeling, decoding, and learning problems in more detail and present some well-known examples from the literature.

2.2 Modeling and Decoding

Researchers constantly invent new translation models, with the trend being to build models with richer and richer derivational structure. The IBM models (Brown et al., 1993) were fundamentally word-based models of translation. Recent models move beyond words to multi-word units. The simplest use flat word sequences called **phrases** (Koehn et al., 2003). We review these models in Section 2.2.1. Chiang (2005) added hierarchical structure to phrases and to the way they combine to form translations; we discuss this model in Section 2.2.2. More recently, many researchers have incorporated linguistic syntax into translation modeling. While this area is too large to cover exhaustively, we list efforts in Section 2.2.3 that are most relevant to the model presented in Chapter 4.

As noted in Section 2.1.4 above, the modeling problem for statistical machine translation consists of defining the output space $\mathcal{T}_{\boldsymbol{x}}$ and the feature vector $\boldsymbol{f}$. Given these, algorithms must be developed for efficiently solving Eq. 2.8, i.e., searching $\mathcal{T}_{\boldsymbol{x}}$ while incorporating all of the features in $\boldsymbol{f}$. As discussed in Section 2.1.3, decoding for typical models is NP-complete or otherwise too computationally expensive to be solved exactly in practice, necessitating approximate algorithms.

In more detail, we describe a translation model by addressing each of the following.

- **Derivation variable and output space**. The starting place for defining a model is providing a description of the derivation variable $\boldsymbol{h}$ and specifying the procedure for generating $\mathcal{T}_{\boldsymbol{x}}$ for a given $\boldsymbol{x}$. The construction of $\mathcal{T}_{\boldsymbol{x}}$ is often discussed using translation **rules**. Informally, a rule consumes part of the input text and emits text in the output language. A rule typically carries with it some piece of the derivational structure $\boldsymbol{h}$, whether linguistic structure or otherwise. Building a machine translation system typically requires collecting a massive set of rules from **parallel data**—examples of sentences along with their translations. This process is called **rule extraction**. We will be more precise below when discussing rules and rule extraction in the context of particular translation models.

- **Features and feature decomposition**. We describe the entries in the feature vector $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ and how they decompose additively across the "parts" of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{h}$. This decomposition is essential to enable efficient approximate decoding algorithms.
- **Decoding algorithms**. We give algorithms that incorporate all features $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ while efficiently searching over $\mathcal{T}_{\boldsymbol{x}}$ to find the highest-scoring $\langle \boldsymbol{y}, \boldsymbol{h} \rangle$ pair. In this chapter, we reference the decoding algorithms used in practice and sketch their execution using examples, but we do not provide procedural specifications or pseudocode. Rather, we focus on what any algorithm must capture based on the definitions of $\boldsymbol{h}$ and $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ and on the decomposition of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$. When describing our model in Chapter 4 we will be more explicit about decoding algorithms.

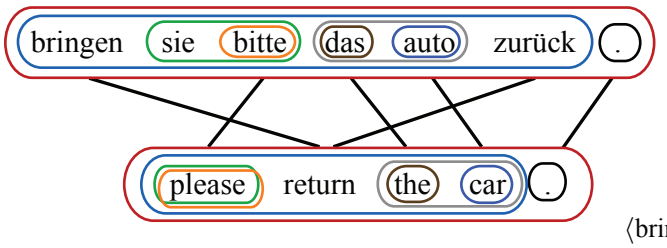
Before proceeding to describe particular models, we introduce some additional notation. We use boldface for vectors and we denote individual elements in vectors using subscripts; e.g., the source and target sentences are denoted $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ and $\boldsymbol{y} = \langle y_1, \ldots, y_m \rangle$. We denote sequences of elements in vectors using subscripts and superscripts; e.g., the sequence from source word i to source word j (inclusive) is denoted $\boldsymbol{x}_i^j$, and therefore $\boldsymbol{x}_i^i = x_i$. We denote the set containing the first k positive integers as $[k]$. All of our notation and definitions are summarized in Appendix D.

2.2.1 Phrase-Based Models

The IBM models were fundamentally word-based translation models, offering limited support for multi-word units. Improvements in translation quality were found by moving from word-based models to so-called **phrase-based** translation models, which are driven by flat multi-word units called **phrases**. Early phrase-based models included the alignment template system (Och and Ney, 2004) and extensions (Zens et al., 2002). Modern phrase-based translation systems are typified by the Moses system (Koehn et al., 2007), based on the approach presented by Koehn et al. (2003). It is this model that we describe in this section. The model is a log-linear model in the style of Och and Ney (2002) but uses different features and rules.

Derivation Variable

In phrase-based models, the latent variable $\boldsymbol{h}$ contains a segmentation of $\boldsymbol{x}$ into n' segments called **phrases**, a segmentation of $\boldsymbol{y}$ into the same number of segments, and a one-to-one alignment from segments in $\boldsymbol{y}$ to segments in $\boldsymbol{x}$. We define a (source-language) **phrase** π as a word sequence $\boldsymbol{x}_j^k$, for j and k such that $1 \leq j \leq k \leq n$, where $n = |\boldsymbol{x}|$. The number of words in phrase π is denoted $|\pi|$. We denote the segmentation of $\boldsymbol{x}$ into phrases as $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_{n'} \rangle$ such that for $i \in [n']$, $\pi_i = \boldsymbol{x}_j^k$ is a source-language phrase and $\pi_1 \cdot \ldots \cdot \pi_{n'} = \boldsymbol{x}$, where $\cdot$ denotes string concatenation. Similarly, the segmentation of $\boldsymbol{y}$ into phrases is denoted $\boldsymbol{\phi} = \langle \phi_1, \ldots, \phi_{n'} \rangle$ such that for $i \in [n']$, $\phi_i = \boldsymbol{y}_j^k$ is a target-language phrase and $\phi_1 \cdot \ldots \cdot \phi_{n'} = \boldsymbol{y}$. The one-to-one alignment (bijection) from phrases in $\boldsymbol{\phi}$ to phrases in $\boldsymbol{\pi}$ is denoted $\boldsymbol{b} : \{1, \ldots, n'\} \to \{1, \ldots, n'\}$. For all $i \in [n']$, if $\boldsymbol{b}(i) = j$, then π_j is a subvector of $\boldsymbol{x}$ and ϕ_i is a subvector of $\boldsymbol{y}$. No longer is $\boldsymbol{a}$ part of the derivation variable,



Extracted phrase pairs:

⟨sie bitte, please⟩
⟨bitte, please⟩
⟨das auto, the car⟩
⟨das, the⟩
⟨auto, car⟩
⟨., .⟩
⟨bringen sie bitte das auto zurück, please return the car⟩
⟨bringen sie bitte das auto zurück ., please return the car .⟩

Figure 2.2: Example of phrase extraction in phrase-based translation. A sentence pair is shown on the left along with word alignments obtained from the IBM word alignment models. Extracted phrase pairs are circled with matching colors and the full set is listed on the right. All phrase pairs that are **consistent** with the word alignments are extracted. See text for details.

since word-to-word alignments are not explicitly represented in phrase-based models. The phrase alignment $\boldsymbol{b}$ has replaced $\boldsymbol{a}$.

The fact that $\boldsymbol{b}$ is a bijection implies that overlapping phrase pairs are not modeled. This is the most common approach, but notable efforts to model overlapping phrase pairs include the Sinuhe system (Kääriäinen, 2009) and context-based machine translation (Carbonell et al., 2006).

Given these definitions, $\boldsymbol{h} \stackrel{\text{def}}{=} \langle \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b} \rangle$. Given an input $\boldsymbol{x}$, a $\langle \boldsymbol{y}, \boldsymbol{h} \rangle = \langle \boldsymbol{y}, \langle \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b} \rangle \rangle$ tuple that satisfies the above conditions is allowed to be contained in $\mathcal{T}_{\boldsymbol{x}}$, though it does not have to be.

Populating the Output Space

The definitions above constrain the structure of $\mathcal{T}_{\boldsymbol{x}}$, but they do not tell us how to populate $\mathcal{T}_{\boldsymbol{x}}$ for a given $\boldsymbol{x}$. One simple approach would be to take everything—i.e., instantiate all possible pairs of source and target phrases based on possible word sequences in the two languages, and allow them to be pieced together arbitrarily. But the usual approach is to constrain the output space based on a large set of translation rules called **phrase pairs**. A phrase pair $\langle \pi, \phi \rangle$ is an ordered pair such that $\pi \in \mathcal{X}$ and $\phi \in \mathcal{Y}(\pi)$. That is, a phrase pair $\langle \pi, \phi \rangle$ licenses a translation option: π can be translated into ϕ.

The full set $\mathcal{P}$ of phrase pairs is called a **phrase table**. Given a phrase table $\mathcal{P}$, an output $\langle \boldsymbol{y}, \langle \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b} \rangle \rangle$ is in $\mathcal{T}_{\boldsymbol{x}}$ if, for all $i \in [n']$, $(\boldsymbol{b}(i) = j) \Rightarrow \langle \pi_j, \phi_i \rangle \in \mathcal{P}$. That is, every pair of aligned phrases in the derivation must be in the phrase table $\mathcal{P}$. To handle unknown words, it is common to add "identity" phrase pairs $\{\langle x_i, x_i \rangle\}_{i=1}^{n}$ to $\mathcal{P}$ when translating $\boldsymbol{x}$. Such phrase pairs are harshly penalized so that they will only be used to translate words for which no other phrase pairs are available.

Rule Extraction

How do we construct the phrase table? Phrase pairs are extracted from a parallel corpus using a procedure like the following from Koehn et al. (2003). First, word alignments are obtained for each sentence pair in the corpus using a word alignment system. There are many word aligners available, including the GIZA++ toolkit (Och and Ney, 2003), which is an implementation of the IBM models (Brown et al., 1993), and the Berkeley aligner (Liang et al., 2006b; DeNero and Klein, 2007). Given word-aligned sentence pairs, we extract phrase pairs that are **p-consistent** with (i.e., do not violate) the word alignments. Formally, let R denote a relation between the two sets $[n]$ and $[m]$, where $n = |\boldsymbol{x}|$ and $m = |\boldsymbol{y}|$. If a pair $(i, j) \in R$, for some $i \in [n]$ and $j \in [m]$, then we say that x_i is aligned to y_j. We define new notation R here instead of using $\boldsymbol{a}$ or $\boldsymbol{b}$ because R allows many-to-many word alignments, which are typically used for phrase extraction.[2] A phrase pair $\langle \boldsymbol{x}_i^j, \boldsymbol{y}_k^l \rangle$ is **p-consistent** with R if, $\forall u$ such that $i \leq u \leq j$, and all v such that $(u, v) \in R$, it is the case that $k \leq v \leq l$. P-consistency implies that one can start at one of the two phrases, follow any of the word alignments leaving it, and land somewhere in the other phrase. Figure 2.2 shows an example of extracted phrase pairs that are consistent with word alignments.

Given a bag of extracted phrase pairs, various statistics are computed to score the phrase pairs. For example, the conditional probability of one phrase given the other is often used, estimated using relative frequencies:

$$\tilde{p}(\phi \mid \pi) = \frac{\#\{\langle \pi, \phi \rangle\}}{\sum_{\phi'} \#\{\langle \pi, \phi' \rangle\}} \tag{2.9}$$

where $\#\{\langle \pi, \phi \rangle\}$ denotes the count of the phrase pair $\langle \pi, \phi \rangle$ in the bag of extracted phrase pairs. The analogous quantity $\tilde{p}(\pi \mid \phi)$ is also often used, computed using a similar formula. These scores are stored in the phrase table with each phrase pair and are used when defining features, which we discuss next.

Features and Feature Decomposition

In structured models, the feasibility of inference depends upon the choice of feature functions $\boldsymbol{f}$. Typically these feature functions are chosen to *factor* into local parts of the overall structure.

The feature vector $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \langle \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b} \rangle)$ includes several types of features:

- **phrase pair features** $\boldsymbol{f}_p(\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b})$. This is a vector of scores associated with each phrase pair in the phrase alignment $\langle \pi_{b(i)}, \phi_i \rangle_{i=1}^{n'}$. These typically include $\log \tilde{p}(\pi_{b(i)} \mid \phi_i)$ and $\log \tilde{p}(\phi_i \mid \pi_{b(i)})$ computed as in the previous section, lexical weighting scores (Koehn et al., 2003), and the phrase pair count n'. All of these features decompose additively across the aligned phrase pairs in the derivation.

- **word count** $f_w(\boldsymbol{y})$. This feature simply counts the number of words $|\boldsymbol{y}|$ in the translation; it decomposes additively across the words in $\boldsymbol{y}$.

[2] Many-to-many word alignments can be obtained from certain alignment models or, more frequently, by using heuristics to combine alignments from one-to-many and many-to-one alignments (Koehn et al., 2003).

- **language model** $f_\ell(\boldsymbol{y})$. This returns the log-probability of the translation under an N-gram language model, i.e., where $m = |\boldsymbol{y}|$, $f_\ell(\boldsymbol{y}) = \log \tilde{p}(\boldsymbol{y}) = \sum_{i=1}^{m+N-1} \log \tilde{p}(y_i \mid y_{i-N+1}^{i-1})$, where $y_i \triangleq \langle \text{BOS} \rangle$ for $i < 1$ and $y_i \triangleq \langle \text{EOS} \rangle$ for $i > m$. It decomposes additively across the N-grams in $\boldsymbol{y}$.

- **linear distortion penalty** $f_d(\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b})$. This is a simple linear distortion feature that counts the number of words skipped in $\boldsymbol{x}$ when considering the sequence of source phrases aligned to target phrases. When generating a target phrase ϕ_i, this potentially requires looking at the entirety of $\boldsymbol{\pi}$ and the range of $b(j)$ for $1 \leq j \leq i$. That is, it only decomposes additively across subsequences $\boldsymbol{\phi}_1^i$, for $1 \leq i \leq |\boldsymbol{y}|$.

- **lexicalized reordering features** $\boldsymbol{f}_r(\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b})$. There have been several lexicalized reordering models developed for phrase-based translation. The default model in the Moses machine translation system (Koehn et al., 2005) includes features that consider consecutive target phrases ϕ_i and ϕ_{i+1} and the source phrases they are aligned to, namely source phrases $b(i)$ and $b(i+1)$. For example, if $b(i)+1 = b(i+1)$, we have a **monotone** reordering configuration. If $b(i) = b(i+1)+1$, the phrases are in a **swap** configuration. Finally, if neither of the first two conditions is true, we have a **discontinuous** configuration. There are six feature functions that score these configurations in various ways. These features decompose additively across all bigrams of aligned phrase pairs in the derivation.

Given these features, we define

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \langle \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b} \rangle) \triangleq \langle \boldsymbol{f}_p(\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b}), f_w(\boldsymbol{y}), f_\ell(\boldsymbol{y}), f_d(\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b}), \boldsymbol{f}_r(\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b}) \rangle$$

Decoding and Phrase Lattices

The most common decoding strategy for phrase-based models is to use beam search (Koehn et al., 2003), although others have been proposed. For example, Zaslavskiy et al. (2009) formulated phrase-based decoding as a traveling salesman problem and Chang and Collins (2011) gave an algorithm based on Lagrangian relaxation that is capable of exact decoding.

Figure 2.3 shows an example of the sort of beam search performed by phrase-based decoders. The search is performed by choosing entries from the phrase table and applying them to translate source phrases into the target language. Decoding begins with an empty string and no words translated (the leftmost box) and ends at a state in which all tokens in the source sentence have been translated exactly once (the double-bordered box on the right). Multiple paths corresponding to different derivations (different values of $\boldsymbol{h}$) are possible for the same translation. **Coverage vectors** have to be maintained during decoding to ensure that all derivations obey the constraints of $\mathcal{T}_{\boldsymbol{x}}$, namely the constraint that each word appear in exactly one phrase pair. Computing the linear distortion feature also requires looking at a part of the coverage vector.

It is often convenient to build a packed representation of the (pruned) search space explored during decoding. For phrase-based models, this representation takes the form of a **phrase lattice** (Ueffing et al., 2002), a finite-state acceptor in which each path corresponds to a derivation.

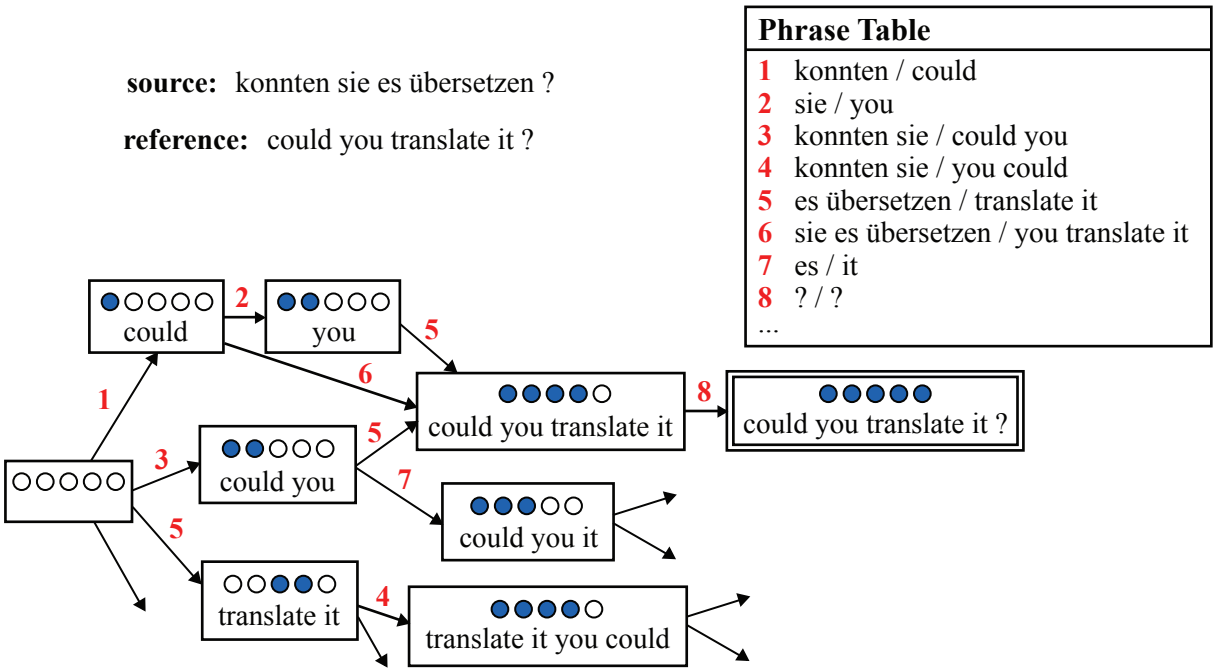


Figure 2.3: Example of search in phrase-based translation. A snippet of a phrase table is shown in the upper right. Each box corresponds to a state in the search, and each circle corresponds to a token in the source sentence and is filled when the token gets translated. The search begins at the empty box on the left. Each directed edge applies an entry from the phrase table to translate some part of the source sentence. When all circles are filled (the double-bordered box on the right), a complete translation has been found.

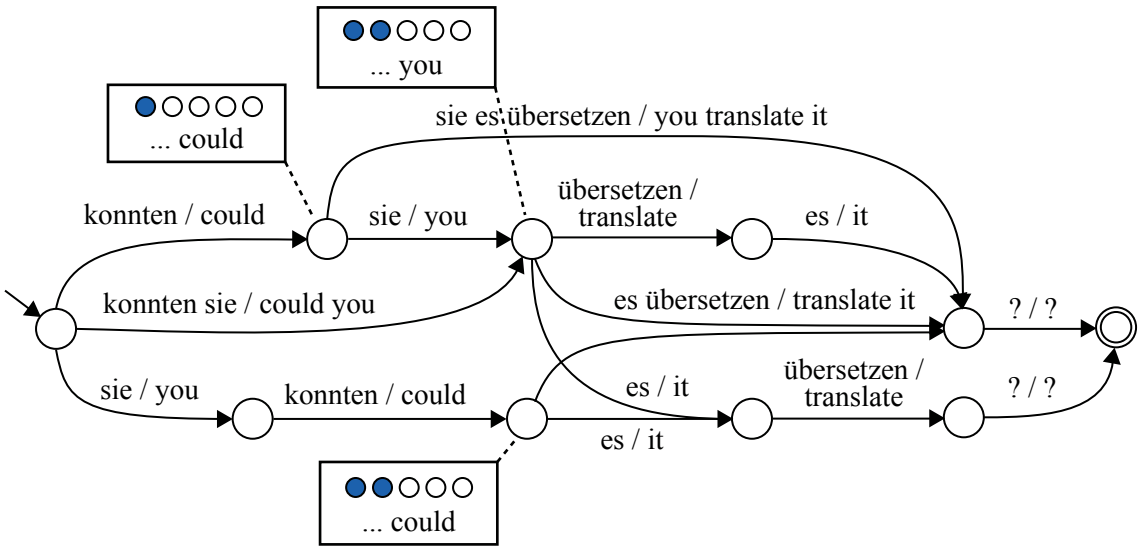


Figure 2.4: Example phrase lattice for the example in Figure 2.3. Each node contains an N-gram history for computing N-gram language model features and a coverage vector representing the source words that have been translated so far. For clarity, the N-gram history ($N = 2$) and coverage vector are only shown for three nodes.

Figure 2.4 shows a sample phrase lattice for the example from Figure 2.3. Each lattice edge corresponds to a phrase pair. Each path from the start node on the left to a final node corresponds to a derivation under the model. Since only derivations in $\mathcal{T}_x$ are included in the lattice, every path obeys the coverage constraints. Further, all paths leading to a given node in the lattice must agree in the set of source words that have been translated thus far. So, every node in the lattice can be annotated with the coverage vector of all paths that end at it. This is shown for three of the nodes in the figure.

The lattice is constructed such that all features in the model are locally computable on individual lattice edges. This has different consequences for the different feature types:

- **phrase pair / word count features**: the phrase table features and the word count feature are already local, since each lattice edge corresponds to a single phrase pair from the phrase table.
- **language model**: to make N-gram language model features local, the lattice is constructed such that all paths leading to a given node end in the same $N - 1$ words; in the example, even though two of the nodes with coverage vectors shown have the same coverage, they are separated in the lattice because they end in different words (*you* vs. *could*). While this can lead to massive state-splitting when using N-gram models with large N, we only need

to split states when the N-gram history is contained within the language model. Typically, as N grows, it becomes more likely for the N-gram history to be absent from the language model. If the language model has no knowledge of the full history, less state-splitting can be used; this is related to **recombination** of partial hypotheses that are indistinguishable as far as features are concerned (Och et al., 2001; Li and Khudanpur, 2008).

- **linear distortion penalty**: this feature requires knowing the index in the source sentence of the leftmost untranslated word; this information is contained within the coverage vector, so no additional information is required beyond that needed by the coverage constraints.
- **lexicalized reordering model**: the default lexicalized reordering features implemented in Moses require information about the phrase that was translated previously. For nodes whose outgoing edges all form monotone configurations, all paths to the node must end in edges with the same ending index in their source-side phrases. If the next phrase forms a swap configuration, all paths to the node must end in an edge with the same *start* index in their source phrases. For phrases leading to discontinuous configurations, all paths to a node must end in an edge with a source phrase that has a different start index and a different end index to all of the next phrases, i.e., phrases on edges emanating from the current node.

Decoders like Moses can easily output phrase lattices like these during decoding, since the lattice simply encodes the paths explored during the beam search. A simple way to add features to a phrase-based model is to first generate a phrase lattice using a phrase-based decoder, then incorporate the features through dynamic programming on the phrase lattice. We use this strategy in Chapters 3 and 4 to add additional features and derivation structure on top of the standard phrase-based model described here.

Feature Locality and Approximate Search

While a full discussion is beyond the scope of this thesis, we make some remarks here on the nature of feature locality in the context of machine translation. We suggest that MT researchers could benefit by being more concrete about how their features decompose across parts of the output structures. Traditionally, since approximate beam search is used in practice for many models, especially for phrase-based models, researchers need not always think about how "local" a particular feature is. When a researcher adds a feature, since approximate beam search is used anyway, the states are split according to the requirements of the new feature. This is made precise in recombination during decoding, and is evident from data structures like phrase lattices, but it is seldom made explicit in model descriptions.

In literature on graphical models and factor graphs, feature locality is well-understood and in fact is made explicit in drawing the factor graph. This explicit representation of feature locality encourages general research in approximate inference algorithms, and leads to better understanding of the computational expense of adding particular features. This is why we have chosen to discuss phrase-based models in terms of their features and feature decomposition across steps in the search. While the maintenance of coverage vectors necessitates approximate inference, we

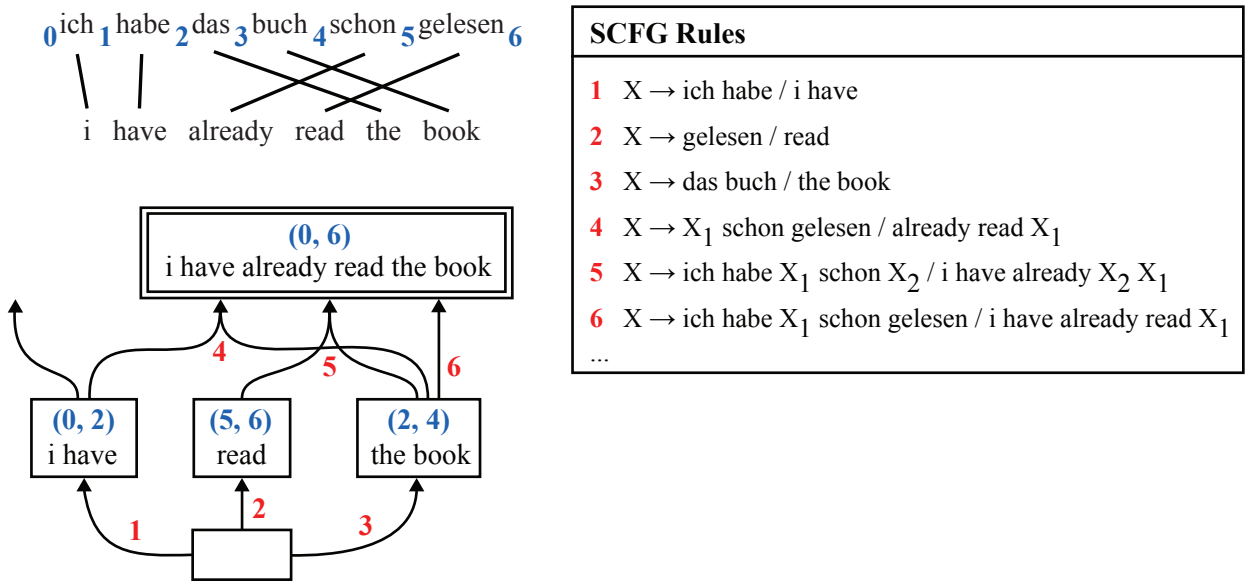


Figure 2.5: Example of search in hierarchical phrase-based translation. A snippet of a rule table is shown in the upper right. Each box corresponds to a state in the search and contains the span (shown as an ordered pair) in the source sentence that has already been translated. The search begins at the empty box on the bottom. Each directed hyperedge applies an entry from the rule table to translate some part of the source sentence. When we reach a box whose span is the entire source sentence (the double-bordered box at the top), a complete translation has been found.

argue that, in general, reducing state-splitting reduces search error. Therefore, it can still be useful to better understand the relationship between features and the amount of state-splitting they cause. For example, using higher-order N-gram language models frequently improves translation quality, but it leads to more state-splitting and therefore yields greater risk of search error.

2.2.2 Hierarchical Phrase-Based Models

A second category of translation models use rules with hierarchical structure, but without linguistic syntax. Wu (1997) developed an early example and today most systems in this category follow the **hierarchical phrase-based translation** system of Chiang (2005, 2007). This system, called Hiero, uses translation rules with hierarchical structure. Several open-source implementations have been developed and are widely-used, such as Joshua (Li et al., 2009a), `cdec` (Dyer et al., 2010), and Moses (Hoang et al., 2009). We give a brief discussion of hierarchical phrase-based translation in this section, as we use a Hiero system in our experimental comparison of learning algorithms in Chapter 3, but further details are not essential to the rest of the thesis.

The rules in Hiero reside in a grammatical formalism called **synchronous context-free grammar** (SCFG), which is a straightforward extension of context-free grammar to a bilingual setting; two strings are generated using a single synchronous derivation. Each rule in an SCFG has flat sequences of terminals and non-terminals for each language. The non-terminals are in a 1-to-1 correspondence, and there is only a single left-hand side non-terminal for each rule.

Hiero rules are extracted from parallel text without any syntactic annotations, and are compiled into an SCFG. The latent derivation variable is a derivation under this SCFG. Many of the features used are similar to those in phrase-based systems, including N-gram language models and rule probabilities derived from relative frequency estimates on extracted rules.

Figure 2.5 shows an example of some hierarchical rules as well as the search performed by decoders for hierarchical phrase-based models. In each rule, the right-hand side contains the source-language yield followed by the target-language yield. Non-terminals are all "X," and subscripts indicate alignment between non-terminals in the source and target yields. The search is performed by choosing entries from the rule table and applying them to translate spans of source words into the target language. Decoding begins with an empty string and no words translated (the box at the bottom) and ends at a state in which all words in the source sentence have been translated (the double-bordered box at the top). Multiple paths corresponding to different derivations are possible for the same translation.

As with phrase lattices for phrase-based decoding, researchers often make use of packed representations of the model's search space. For models based on monolingual or synchronous grammars, this has the form of a (directed) **hypergraph** (Klein and Manning, 2001). A hypergraph contains nodes connected by **hyperedges**, which have a single destination node but can have multiple tail nodes. In the search in Figure 2.5, each hyperedge is labeled with an integer indicating the SCFG rule that it corresponds to.

2.2.3 Syntax-Based Models

While the models above use structure beyond mere word-pairs, namely phrases and hierarchical rules, they do not use **linguistic syntax**. **Syntax-based** translation models date to Yamada and Knight (2001, 2002), who designed a model and a decoder for translating a source-language string into a target-language string along with its phrase structure parse. They used an automatic parser trained from the Penn Treebank (Marcus et al., 1993) to parse the target (English) side of the parallel corpus. Approaches of this sort are called **string-to-tree**, because translation proceeds by transducing a source-side string into a target tree (with its string yield). Many other string-to-tree systems have been developed, both using phrase structure trees (Galley et al., 2004, 2006; Zollmann and Venugopal, 2006) and dependency trees (Shen et al., 2008). In such models, $\boldsymbol{h}$ includes a target-sentence parse tree and decoding algorithms must therefore search over a space containing trees in addition to translations.

If instead the *source* sentence is parsed using an automatic parser and translation proceeds by producing a target-side string, the approach is called **tree-to-string** or **syntax-directed** translation (Huang et al., 2006; Liu et al., 2006; Lin, 2004). The final category, **tree-to-tree**, translates a parsed source sentence into a target sentence with a syntactic parse (Ding and Palmer, 2005;

Cowan et al., 2006; Liu et al., 2009a).

Tree-to-tree translation has proved difficult, as initial attempts underperformed phrase-based systems (Cowan et al., 2006; Ambati and Lavie, 2008; Liu et al., 2009a). As Ding and Palmer (2005) note, when using syntactic parses for both the source and target languages, we need to address syntactic divergence. Otherwise, the syntactic structures constrain the extracted rules too severely. Subsequent research showed that substantial performance gains can be achieved by relaxing hard constraints imposed by source and target syntax (Liu et al., 2009a; Chiang, 2010; Zhang et al., 2011; Hanneman and Lavie, 2011). We follow this philosophy in our tree-to-tree model, as we forgo isomorphism constraints and manage the enlarged search space using additional features.

Aside from tree-to-string, string-to-tree, and tree-to-tree systems, researchers have added features derived from linguistic syntax to phrase-based and hierarchical phrase-based systems. In work not included in this thesis, we added features based on source-language dependency parses and phrase structure parses to a phrase-based system (Gimpel and Smith, 2008). Haque et al. (2009, 2010, 2011) used source-side features based on combinatory categorial grammar (Steedman, 2000) and lexicalized tree-adjoining grammar (Schabes, 1992), both for a phrase-based system and a hierarchical phrase-based system.

Several others used source-side phrase structure parses to add features to hierarchical phrase-based translation (Marton and Resnik, 2008; Chiang et al., 2008b; Blunsom and Osborne, 2008). Xiong et al. (2008) used linguistic syntax within a model based on bracketing transduction grammar (BTG); they used a source-side parser and projected the annotations to the BTG derivation structure on the target side. Also related is work on using syntax to reorder the source sentence as a preprocessing step before translation (Xia and McCord, 2004).

Of course, not all systems that use syntax fit into these categories. For example, classical **transfer** systems (e.g., Probst et al., 2002) use syntactic rules without necessarily using parsers for either language.

Synchronous Grammars for Translation Modeling

To model the syntactic transformation process, researchers have developed powerful grammatical formalisms. Many are variations of **synchronous grammars**. Our starting point is synchronous context-free grammar (SCFG), which we introduced briefly in Section 2.2.2. Several types of SCFGs have been defined for translation. For example, Wu (1997) proposed inversion-transduction grammar, a restricted type of SCFG that only permits two types of rules. Melamed (2003) designed an SCFG class that captures constructions that frequently arise in translation. But several researchers have noted that real data exhibits divergences that cannot be captured by SCFG (Wellington et al., 2006; Søgaard and Kuhn, 2009). As a result, several synchronous formalisms have been proposed for translation modeling that offer more expressive power.

Shieber and Schabes (1990) developed synchronous tree adjoining grammar (STAG) and Abeillé et al. (1990) used lexicalized tree adjoining grammars for machine translation. Eisner (2003) proposed synchronous tree substitution grammar (STSG) for translation, which is like STAG but omits adjunctions. These two formalisms offer **extended domain of locality** (Joshi and Schabes, 1997) over SCFG, meaning that each rule allows richer structures (e.g., "elementary

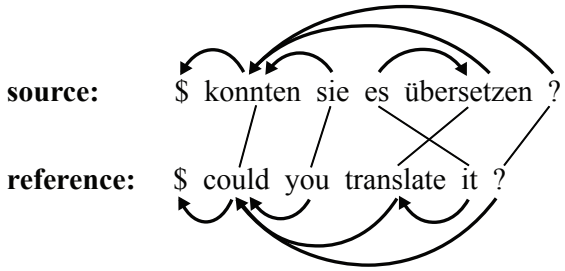


Figure 2.6: Examples of (unlabeled) dependency trees. Arrows are drawn from children to parents. A child word is a modifier of its parent. Each word has exactly one parent and $ is a special "wall" symbol indicating the root of the tree. These trees are projective; see text for details.

trees") than the flat sequences of terminals and non-terminals in SCFG rules. Graehl and Knight (2004) and Graehl et al. (2008) similarly allowed productions with multi-level tree structures for one of the two languages. They worked directly with tree transducers (Gécseg and Steinby, 1984) which are a lower-level representation than synchronous grammars.

Synchronous grammars with extended domain of locality, such as STAG and STSG, capture non-isomorphic structure using the elementary trees within individual rules, but the rules still must be pieced together in a single isomorphic derivation. While one can permit rules to be arbitrarily large, larger rules tend to be more specific and will apply less frequently when translating new data. Furthermore, Søgaard and Kuhn (2009) noted examples in manually-aligned parallel corpora that even STSG and STAG cannot handle.

We have discussed these synchronous formalisms in order to more clearly differentiate ***quasi-synchronous grammar*** (QG; Smith and Eisner, 2006a), which we will describe in Chapter 4. Briefly, QG assumes that the source tree is provided; a QG induces a *monolingual* grammar that scores translations and syntactic trees over them. Because the model does not generate the two strings simultaneously, arbitrary non-isomorphic structures are permitted.

We next discuss dependency syntax because it underlies the quasi-synchronous grammar models we develop in this thesis.

Dependency Syntax

There are many syntactic translations models; we focus in this thesis on those founded upon **dependency syntax** (Tesnière, 1959; Kübler et al., 2009). Dependency syntax is a lightweight formalism that builds trees consisting of a set of directed arcs from words to their syntactic heads. The arcs frequently have labels, such as "adjectival modifier", "determiner", "direct object", "temporal modifier", etc., but in this thesis we use only unlabeled dependency trees. This is also the choice made by most others (Ding and Palmer, 2005; Shen et al., 2008; Galley and Manning, 2009) who use dependency syntax for machine translation. Examples of dependency trees are

shown in Figure 2.6. Each word has exactly one parent and \$ is a special "wall" symbol indicating the root of the tree. Formally, a **dependency tree** on an m-word sentence $\boldsymbol{y}$ is a function $\tau_{\boldsymbol{y}} : \{1, \ldots, m\} \to \{0, \ldots, m\}$ where $\tau_{\boldsymbol{y}}(i)$ is the index of the parent of word y_i. If $\tau_{\boldsymbol{y}}(i) = 0$, we say word y_i is the **root** of the tree. The function $\tau_{\boldsymbol{y}}$ is not permitted to have **cycles**. To define a cycle, we first define the **ancestor path** for a word y_i as a length-q vector $\boldsymbol{\nu}$ such that $\nu_1 = i$, $\nu_q = 0$, and for all $1 \leq j < q$, $\nu_{j+1} = \tau_{\boldsymbol{y}}(\nu_j)$. An ancestor path starts at the word and follows its ancestors up the dependency tree to the root. Assuming the above definitions of $\tau_{\boldsymbol{y}}$ and $\boldsymbol{\nu}$, all ancestor paths have finite length if and only if there are no cycles in $\tau_{\boldsymbol{y}}$.

In this thesis we primarily deal with **projective** dependency trees, which are informally defined as dependency trees with no crossing arcs when all dependencies are drawn on one side of the sentence. More formally, a **projective dependency tree** is a dependency tree $\tau_{\boldsymbol{y}}$ in which, if $\tau_{\boldsymbol{y}}(i) > i$, then $\forall j \in [m]$, the following two hold:

$$(i < j < \tau_{\boldsymbol{y}}(i)) \Rightarrow (i \leq \tau_{\boldsymbol{y}}(j) \leq \tau_{\boldsymbol{y}}(i))$$
$$((j < i) \vee (j > \tau_{\boldsymbol{y}}(i))) \Rightarrow ((\tau_{\boldsymbol{y}}(j) \leq i) \vee (\tau_{\boldsymbol{y}}(j) \geq \tau_{\boldsymbol{y}}(i)))$$

The first condition requires that the parent of word y_j cannot be outside the subtree that contains y_j, and the second condition specifies that no word outside the subtree has its parent inside the subtree. Two analogous conditions hold for the case when $\tau_{\boldsymbol{y}}(i) < i$. A **non-projective dependency tree** simply drops the projectivity conditions. Since we focus on projective dependency trees in this thesis, we will typically drop the "projective" modifier and use the term "dependency tree" to mean "projective dependency tree."

We define a (projective) **dependency grammar** as a model that assigns scores to sentences and projective dependency trees over them. In this thesis, we will be more specific and define a dependency grammar as a linear model parameterized by a vector of parameters $\boldsymbol{\theta}$ and with feature vector $\boldsymbol{f}$. It need not be probabilistic; we only specify that it be a linear model and provide scores to sentences and dependency trees. We also assume the features are **arc-factored**, meaning that they additively decompose across arcs in the dependency tree. Thus the best sentence and dependency tree under the dependency grammar are chosen as follows:

$$\langle \boldsymbol{y}^*, \tau_{\boldsymbol{y}}^* \rangle = \operatorname*{argmax}_{\boldsymbol{y}, \tau_{\boldsymbol{y}}} \sum_{i=1}^{|\boldsymbol{y}|} \boldsymbol{\theta}^\top \boldsymbol{f}(y_i, y_{\tau_{\boldsymbol{y}}(i)}, i, \tau_{\boldsymbol{y}}(i)) \tag{2.10}$$

We note that this form looks unusual to those familiar with standard (monolingual) dependency parsing, because it maximizes over sentences and dependency trees over them. Typically, in monolingual parsing the sentence $\boldsymbol{y}$ is fixed, allowing any feature in $\boldsymbol{f}$ to look at any part of $\boldsymbol{y}$. Also, in order to be useful for choosing the best tree, all features in $\boldsymbol{f}$ must look at some part of $\tau_{\boldsymbol{y}}$. So the arc-factored dependency parsing problem is usually written as below, given a sentence $\boldsymbol{y}$ with m words (McDonald et al., 2005; *inter alia*):

$$\tau_{\boldsymbol{y}}^* = \operatorname*{argmax}_{\tau_{\boldsymbol{y}}} \sum_{i=1}^{m} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{y}, i, \tau_{\boldsymbol{y}}(i)) \tag{2.11}$$

We presented the more general form in Eq. 2.10 first because it is more appropriate for translation modeling. For translation, $\boldsymbol{f}$ includes features that score parts of $\boldsymbol{y}$, like language model features. Monolingual dependency parsing (Eq. 2.11) does not; all features must look at some part of $\tau_{\boldsymbol{y}}$ to affect the argmax.

While our use of the term "dependency grammar" in relation to Eq. 2.10 may seem alien to parsing researchers, we note here that probabilistic context-free grammars similarly assign scores to both sentences and trees. When used for monolingual parsing, the sentence is fixed ("conditioned on") and the grammar is used to choose the best tree for the sentence. When discussing dependency grammars for monolingual parsing, the generative formulation is usually skipped in favor of a formulation like that in Eq. 2.11, but we note that weighted grammars are generally understood to assign scores to strings in a language (or more accurately, to *derivations* of strings in a language).

Dependency Syntax in Machine Translation

Researchers have shown that dependency trees are better preserved when projecting across word alignments than phrase structure trees (Fox, 2002). This makes dependency syntax appealing for translation modeling, but to date there are not many tree-to-tree translation models that use dependency syntax on both sides. One key exception is the system of Ding and Palmer (2005), who use an STSG designed for dependency syntax, capturing non-isomorphic structure within rules using elementary trees. Another exception is the system of Riezler and Maxwell III (2006), who use lexical-functional dependency trees on both sides and also include phrase translation rules in their model.

But most use dependency syntax on either the source or target side only. For example, Shen et al. (2008, 2010) presented an extension to the hierarchical phrase-based model from Section 2.2.2 in which SCFG rules have dependency syntax on the target side, and added a dependency language model. Su et al. (2010) and Tu et al. (2010) also presented string-to-tree models with dependency syntax on the target side.

Other string-to-tree systems have closer integration with phrase-based models. Galley and Manning (2009) added a dependency language model to a phrase-based translation system. Carreras and Collins (2009) presented a string-to-dependency system that permits non-projective dependency trees (thereby allowing a larger space of translations) and used a rule extraction procedure that includes rules for every phrase in the phrase table. Hunter and Resnik (2010) presented a formalism that combines dependency syntax with phrases, and their formalism is similar to the one we present in Chapter 4.

Others developed tree-to-string models using dependency syntax on the source side (Lin, 2004; Xiong et al., 2007; Xie et al., 2011), or using features derived from source dependency parses in other models (Gao et al., 2011). Quirk et al. (2005) used a source-side dependency parser and projected automatic parses across word alignments in order to model dependency syntax on phrase pairs.

In earlier work, Alshawi et al. (2000) presented a synchronous dependency grammar defined using collections of head automata. It is not technically a syntax-based model according to our

definition because no syntactic parsers are used. Rather, the model induces synchronous dependency structure from unannotated parallel text, so it is closer to the formally-syntactic hierarchical phrase-based model of Chiang (2007) than to modern dependency-syntax-based models.

In the model we present in Chapter 4, we include several sets of features that relate to this prior work. One set only looks at the dependency structure on the target side; these are similar to dependency language model features. Another group of features is tied to rules that contain source phrases and target dependency syntax; these are similar to those used in string-to-tree systems. Finally, we include tree-to-tree features that look at local configurations, allowing the model to score particular patterns of non-isomorphic structure.

2.3 Learning

The learning problem for statistical machine translation corresponds to choosing $\boldsymbol{\theta}$ in Eq. 2.8. The original noisy channel models (covered in Section 2.1.1) used 1 for each entry in $\boldsymbol{\theta}$. The only learning involved was estimating the parameters of generative models for the translation model $\tilde{p}(\boldsymbol{x} \mid \boldsymbol{y})$ and the language model $\tilde{p}(\boldsymbol{y})$, whose parameters are distinct from $\boldsymbol{\theta}$. The translation model parameters were estimated using the expectation-maximization algorithm (Dempster et al., 1977; Brown et al., 1993) and the language model parameters were estimated using maximum likelihood estimation with smoothing. Much modern research continues these trends, using generative models for word alignment and language modeling and estimating their parameters in similar ways. However, these parameters are only used *within* feature functions such as the phrase pair and N-gram language model features defined in Section 2.2.1, or for word alignment before rule extraction. Given feature functions and their estimated parameters, we still need a way to choose feature weights $\boldsymbol{\theta}$.

So our focus here (and in Chapter 3) is on choosing values for feature weights $\boldsymbol{\theta}$ in the linear model formulation of Och and Ney (2002). When Och and Ney proposed the use of a log-linear model for SMT, they trained it to maximize the **conditional likelihood** of the translations given the input sentences, which is the most commonly-used learning criterion for log-linear models. However, conditional likelihood cannot incorporate a task-specific evaluation metric. So Och (2003) introduced **minimum error rate training** (MERT), which permits learning to be tailored to a particular metric and is currently the most frequently-used approach. The MERT training procedure ignores the normalization in the model, so when MERT is used, it is more appropriate to use the term "linear model" rather than "log-linear model." In this thesis, we will call the model a linear model except when discussing learning procedures that require the log-linear formulation.[3]

The remainder of this section is organized as follows. We begin by discussing how the machine translation problem differs from other structure prediction tasks in ways that complicate learning (Section 2.3.1). We then discuss automatic evaluation metrics for machine translation and their role in learning and evaluation (Section 2.3.2). We propose the general framework of **empirical risk minimization** as a way to discuss and compare training methods for MT (Section 2.3.3), and

[3]We clarify this point because many MT researchers still refer to the model as a "log-linear model" even when using MERT for learning, which may lead to confusion.

we present three examples in the following sections: MERT, **Bayes risk minimization** (Smith and Eisner, 2006b), and **pairwise ranking optimization** (PRO; Hopkins and May, 2011). We delay our presentation of other learning algorithms for MT (e.g., the perceptron and the margin-infused relaxed algorithm) until Chapter 3 as they relate more closely to the ones we introduce. While much of this section focuses on declarative specifications of learning methods, we close with remarks on practical issues related to designing learning algorithms for machine translation (Section 2.3.7).

2.3.1 Challenges of Learning in Machine Translation

Machine translation differs from other structure prediction problems in several ways, complicating the application of machine learning techniques. Supervised learning typically assumes a single correct output for each input, but there are often multiple valid translations for any given sentence. Several commonly-used corpora (such as those produced for the NIST evaluations) contain multiple reference translations for each input sentence, produced independently by several human translators.

Furthermore, even if we are given a set of acceptable translations for each input, it is unclear which (if any) should be treated as ground truth during training. One translation may be preferred by a human, but may involve colloquialisms or a degree of freedom in translation that would make it unsuitable for training our machine translation systems. Another translation may have a high score under an automatic evaluation metric, but might be disfluent or may only match function words and omit critical content words. Even after we decide which translation (or set of translations) to use as truth during learning, the translations might not be **reachable** by the model, i.e., might not be in the set $\{\boldsymbol{y} : \langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}}\}$.

Liang et al. (2006a) discussed these issues in detail and showed empirically that even when the provided references *are* reachable, better performance is found by instead using translations that are both similar to the references *and* score highly under the model using the current $\boldsymbol{\theta}$. In particular, they used the translation on a k-best list with the highest automatic metric score. Och and Ney (2002) were the first to use this strategy, and it has been found to work well by others (Arun and Koehn, 2007; Watanabe et al., 2007; *inter alia*). As we will discuss in Chapter 3, this approach actually changes the objective function being optimized during learning.

Another difference between machine translation and many other structure prediction tasks is the presence of latent variables, namely the derivation variable $\boldsymbol{h}$. Statistical translation models have used latent variables since the original IBM models (Brown et al., 1993), as described in Section 2.1.1. Generally, learning is accomplished by optimizing a function. The presence of latent variables complicates learning as it typically necessitates non-convex optimization. While convex optimization is widely-studied (Boyd and Vandenberghe, 2004) and used in machine learning, non-convex learning algorithms have received less attention from the machine learning community. In general, convexity enables easier proofs of convergence and proofs of other theoretical properties of learning algorithms.

In Chapter 3 we discuss how these two characteristics have caused a disconnect between the function minimized by an algorithm in machine learning and the function minimized when the algorithm is adapted for machine translation.

2.3.2 Automatic Evaluation Metrics

The goal in designing and training statistical translation models is to assign scores to translations in proportion to their quality. Human judgments are the gold standard for measuring translation quality, but for parameter estimation and system evaluation it is common to use an automatic metric as a surrogate. Most metrics compute similarity (or distance) between a system's output and references produced by a human translator. While metric design is an active area of research, the BLEU score (BiLingual Evaluation Understudy; Papineni et al., 2001) is the most common choice in practice and is the primary metric used in this thesis. BLEU scores translation output by measuring simple statistics like N-gram overlap with a set of reference translations. The higher the BLEU score, the better the presumed quality of the translations.

For reference translations $\boldsymbol{Y} = \{\boldsymbol{y}^{(i)}\}_{i=1}^{N}$ and a corpus of system outputs $\hat{\boldsymbol{Y}} = \{\hat{\boldsymbol{y}}^{(i)}\}_{i=1}^{N}$, the BLEU score is defined as follows:

$$\text{BLEU}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \text{BP}\left(\sum_{i=1}^{N} |\boldsymbol{y}^{(i)}|, \sum_{i=1}^{N} |\hat{\boldsymbol{y}}^{(i)}|\right) \cdot \exp\left\{\sum_{k=1}^{4} \frac{1}{4} \log \text{prec}_k(\boldsymbol{Y}, \hat{\boldsymbol{Y}})\right\}$$

where the brevity penalty BP is defined:

$$\text{BP}(a, b) = \begin{cases} 1 & \text{if } b > a \\ \exp\{1 - \frac{a}{b}\} & \text{if } b \leq a \end{cases}$$

and where the modified precision prec_k is defined:

$$\text{prec}_k(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{\sum_{i=1}^{N} \sum_{\boldsymbol{z} \in \text{grams}(\hat{\boldsymbol{y}}^{(i)}, k)} \min\left(\text{count}(\boldsymbol{z}, \hat{\boldsymbol{y}}^{(i)}), \text{count}(\boldsymbol{z}, \boldsymbol{y}^{(i)})\right)}{\sum_{j=1}^{N} \sum_{\boldsymbol{z} \in \text{grams}(\hat{\boldsymbol{y}}^{(j)}, k)} \text{count}(\boldsymbol{z}, \hat{\boldsymbol{y}}^{(j)})}$$

where $\text{grams}(\boldsymbol{y}, k)$ returns the set of length-k subsequences of $\boldsymbol{y}$ (i.e., the set of k-grams in $\boldsymbol{y}$) and $\text{count}(\boldsymbol{z}, \boldsymbol{y})$ returns the number of occurrences of n-gram $\boldsymbol{z}$ in $\boldsymbol{y}$. The $\min$ in the numerator causes **count clipping**, meaning that multiple instances of a correct n-gram are only rewarded if each can be matched to an instance in the reference translation. We have shown BLEU for a single set of reference translations for simplicity, but the original intent of BLEU was for use with multiple references, and the definition is easily adapted for this case.

While the counts of n-gram matches and n-gram totals do decompose additively across the sentences in a corpus, the BLEU score as a whole does *not* additively decompose across the sentences. This can cause problems for learning algorithms. A standard desideratum in learning is to tailor the learning procedure to the final task evaluation metric. However, for computational tractability, most learning algorithms that can do this are limited to instance-level metrics that decompose additively across the instances. One idea is to simply compute the BLEU score separately

for each sentence and then add them, but when one of the N-gram precisions is zero (which happens frequently in practice for individual sentences and a single reference translation), the BLEU score is zero. So, for a sentence-level metric it is common to use a smoothed version of BLEU, such as BLEU$_{+1}$ (Lin and Och, 2004). This variation uses a modified version of prec_k that adds 1 to its numerator and denominator. We call this prec_k^{+1}; when computed for a single sentence with reference translation $\boldsymbol{y}$ and hypothesis $\hat{\boldsymbol{y}}$, it is defined:

$$\text{prec}_k^{+1}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1 + \sum_{\boldsymbol{z} \in \text{grams}(\hat{\boldsymbol{y}}, k)} \min\left(\text{count}(\boldsymbol{z}, \hat{\boldsymbol{y}}), \text{count}(\boldsymbol{z}, \boldsymbol{y})\right)}{1 + \sum_{\boldsymbol{z} \in \text{grams}(\hat{\boldsymbol{y}}, k)} \text{count}(\boldsymbol{z}, \hat{\boldsymbol{y}})}$$

Then BLEU$_{+1}$ is defined

$$\text{BLEU}_{+1}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \text{BP}\left(|\boldsymbol{y}|, |\hat{\boldsymbol{y}}|\right) \cdot \exp\left\{\sum_{k=1}^{4} \frac{1}{4} \log \text{prec}_k^{+1}(\boldsymbol{y}, \hat{\boldsymbol{y}})\right\}$$

Other smoothed versions of BLEU are possible, but BLEU$_{+1}$ has been shown in practice to work well for learning (Hopkins and May, 2011) and is the primary sentence-level metric we use in this thesis.

2.3.3 Empirical Risk Minimization

In this section we present the framework of **empirical risk minimization** (Vapnik, 1995) which we use to describe and compare learning algorithms for machine translation. We will describe each way to choose $\boldsymbol{\theta}$ in terms of a particular **loss function** $\text{loss} : \mathcal{X} \times \mathcal{Y} \times \boldsymbol{\Theta} \to \mathbb{R}$ that maps an input sentence, its reference translation, and the model parameters to a real value indicating the quality of the parameters.[4]

First, we define **risk minimization** as choosing

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\text{argmin}} \; \mathbb{E}_{p(\boldsymbol{x}, \boldsymbol{y})}\left[\text{loss}\left(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}\right)\right] \tag{2.12}$$

where $p(\boldsymbol{x}, \boldsymbol{y})$ is the (unknown) true joint distribution over input sentences $\boldsymbol{x} \in \mathcal{X}$ and possible translations $\boldsymbol{y} \in \mathcal{Y}$. Since in practice we do not know $p(\boldsymbol{x}, \boldsymbol{y})$, but we do have access to an actual corpus of sentence pairs $\{\langle \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)} \rangle\}_{i=1}^{N}$, we instead consider **regularized empirical risk minimization**:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\text{argmin}} \; \sum_{i=1}^{N} \text{loss}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\theta}) + R(\boldsymbol{\theta}) \tag{2.13}$$

[4]There may be multiple reference translations for each input sentence but for simplicity of notation we assume a single reference translation for each input sentence in the training data. Our discussion can be extended in a straightforward manner to permit multiple references.

where $R(\boldsymbol{\theta})$ is the **regularization function** used to mitigate overfitting. The regularization function is frequently a norm or squared norm of the parameter vector, such as the ℓ_1 or squared-ℓ_2 norm, but many other choices are possible. Empirically, while absolute performance numbers do not vary widely among different regularization approaches for most tasks (Gao et al., 2007), they do achieve different desiderata in the learned parameters; for example, ℓ_1-regularization leads to sparse solutions in which many of the parameters are zero. In this thesis, we use squared-ℓ_2-regularization; it has been shown to be empirically effective for a variety of tasks and is easily handled by gradient methods due to its differentiability.

The learning approaches we consider differ in their definition of loss and R. Models are evaluated using a task-specific notion of error, here encoded as a **cost function**, $\mathrm{cost} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, that compares a system output to a reference translation. The worse a translation is, the larger the cost. The cost function will typically make use of an automatic evaluation metric for machine translation; e.g., $\mathrm{cost}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ might equal $1 - \mathrm{BLEU}_{+1}(\boldsymbol{y}, \hat{\boldsymbol{y}})$.

We will abuse notation and allow cost to operate on both sets of sentences as well as individual sentences. For notational convenience we also let cost accept derivation variables but assume that the derivations do not affect the value; i.e., $\mathrm{cost}(\langle \boldsymbol{y}, \boldsymbol{h} \rangle, \langle \boldsymbol{y}', \boldsymbol{h}' \rangle) = \mathrm{cost}(\boldsymbol{y}, \langle \boldsymbol{y}', \boldsymbol{h}' \rangle) = \mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}')$.

To increase readability when discussing different loss functions, we will often refer to the model score of a derivation using the following:

$$\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) \triangleq \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$$

2.3.4 Minimum Error Rate Training

The most commonly-used training algorithm for machine translation is **minimum error rate training** (MERT), which seeks to directly minimize the cost of the predictions on the training data. This idea has been used in the pattern recognition and speech recognition communities (Duda and Hart, 1973; Juang et al., 1997); its first application to MT was by Och (2003). The loss function for direct cost minimization takes the following form:

$$\mathrm{loss}_{\mathrm{cost}}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\theta}) = \mathrm{cost}\left(\boldsymbol{y}^{(i)}, \operatorname*{argmax}_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \mathrm{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta})\right) \tag{2.14}$$

However, this is not actually the loss function minimized by MERT. Since MERT was developed to handle cost functions like 1-BLEU that do not factor additively across the sentences in a corpus, MERT instead optimizes the following loss which looks at an entire corpus of sentence pairs $\{\langle \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)} \rangle\}_{i=1}^N$:

$$\mathrm{loss}_{\mathrm{MERT}}\left(\{\boldsymbol{x}^{(i)}\}_{i=1}^N, \{\boldsymbol{y}^{(i)}\}_{i=1}^N, \boldsymbol{\theta}\right) = \mathrm{cost}\left(\{\boldsymbol{y}^{(i)}\}_{i=1}^N, \left\{\operatorname*{argmax}_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \mathrm{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta})\right\}_{i=1}^N\right) \tag{2.15}$$

That is, MERT directly minimizes a corpus-level cost function of the best outputs from the decoder. Exactly minimizing these losses is NP-complete, even in the simple case of binary classification (Chen and Mangasarian, 1996).

The two losses are non-convex and non-differentiable for cost functions like 1-BLEU, so Och (2003) used k-best lists in place of $\mathcal{T}_{\boldsymbol{x}^{(i)}}$ and developed a coordinate descent procedure with a specialized line search to optimize one dimension at a time. Subsequent work explored optimizing several parameters at a time and in many random directions during each iteration (e.g., Cer et al., 2008), and this is the approach most often used in practice. Lattice and hypergraph versions of MERT have also been developed (Macherey et al., 2008; Kumar et al., 2009), which offer increased stability. MERT uses no explicit regularization (i.e., $R(\boldsymbol{\theta}) = 0$), though Cer et al. (2008) and Macherey et al. (2008) achieved a sort of regularization by altering MERT's line search.

MERT avoids the need to compute the score for the reference translations, so it does not break if the references are unreachable. Also, MERT allows corpus-level metrics like BLEU to be easily incorporated. However, the complexity of the loss and the difficulty of the search lead to instabilities during learning. Remedies have been suggested, typically involving additional search directions and experiment replicates (Cer et al., 2008; Moore and Quirk, 2008; Foster and Kuhn, 2009; Clark et al., 2011). But despite these improvements, MERT is ineffectual for training weights for large numbers of features; in addition to anecdotal evidence from the MT community, Hopkins and May (2011) illustrated with synthetic data experiments that MERT struggles increasingly to find an effective solution as the number of parameters grows. For these reasons, researchers have explored alternatives, some of which we discuss in the remainder of this chapter. In Chapter 3 we cover the rest and introduce new loss functions and algorithms.

2.3.5 Bayes Risk Minimization

Our use of the term "risk" in presenting empirical risk minimization above should not be confused with the **Bayes risk** learning framework, which seeks to minimize the expected value of the cost function with respect to the model's log-linear probability distribution (Eq. 2.4) over outputs:

$$\begin{aligned}\text{loss}_{\text{Bayes risk}} &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y},\boldsymbol{h}|\boldsymbol{x}^{(i)})}\Big[\text{cost}(\boldsymbol{y}^{(i)},\boldsymbol{y})\Big] \\ &= \sum_{\langle \boldsymbol{y},\boldsymbol{h}\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \text{cost}(\boldsymbol{y}^{(i)},\boldsymbol{y}) \cdot \frac{\exp\{\text{score}(\boldsymbol{x}^{(i)},\boldsymbol{y},\boldsymbol{h};\boldsymbol{\theta})\}}{\sum_{\langle \boldsymbol{y}',\boldsymbol{h}'\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)},\boldsymbol{y}',\boldsymbol{h}';\boldsymbol{\theta})\}} \end{aligned} \tag{2.16}$$

where we have suppressed the arguments on the left-hand side for readability, and will do so throughout (i.e., $\text{loss}_{\text{Bayes risk}} \stackrel{\text{def}}{=} \text{loss}_{\text{Bayes risk}}\left(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)},\boldsymbol{\theta}\right)$).

The use of this loss is often simply called "risk minimization" in the speech and MT communities but we will refer to it as the **Bayes risk** to avoid confusion. Bayes risk is non-convex, whether or not latent variables are present. Like MERT, it naturally avoids the need to compute features for $\boldsymbol{y}^{(i)}$ and uses a cost function, making it appealing for MT. Bayes risk minimization first appeared in the speech recognition community (Kaiser et al., 2000; Povey and Woodland, 2002) and more recently has been applied to MT (Smith and Eisner, 2006b; Zens et al., 2007; Li and Eisner, 2009). Smith and Eisner (2006b) minimized Bayes risk with annealing to address the non-convexity of the loss and Li and Eisner (2009) introduced a novel semiring for minimizing Bayes risk using dynamic programming.

2.3.6 Pairwise Ranking Optimization

Hopkins and May (2011) recently introduced an algorithm called **pairwise ranking optimization** (PRO). The algorithm samples pairs of translations from k-best lists and trains a binary classifier to rank pairs according to the cost function. The goal is for the model score ranking of two translations to indicate which has lower cost. The loss function underlying PRO depends on the choice of binary classifier and also on the sampling strategy. Hopkins and May only sampled pairs with cost difference above a certain threshold. They used BLEU$_{+1}$ in their sentence-level cost function. Given a sample $\mathcal{S}(\boldsymbol{x}^{(i)})$ of pairs $\langle\langle \boldsymbol{y}, \boldsymbol{h}\rangle, \langle \boldsymbol{y}', \boldsymbol{h}'\rangle\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}} \times \mathcal{T}_{\boldsymbol{x}^{(i)}}$, the loss optimized by PRO when using a logistic regression classifier is

$$\begin{aligned}\text{loss}_{\text{PRO}} = \sum_{\substack{\langle\langle \boldsymbol{y}, \boldsymbol{h}\rangle, \langle \boldsymbol{y}', \boldsymbol{h}'\rangle\rangle \\ \in \mathcal{S}(\boldsymbol{x}^{(i)})}} & -\mathbb{I}\left[\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) < \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}')\right]\left(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta})\right) \\ & + \log(1 + \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta})\})\end{aligned}$$

PRO is simple to implement and has been shown to work well in practice (Hopkins and May, 2011). Bazrafshan et al. (2012) trained a regression model instead of a binary classifier and found similar results with faster convergence.

We will empirically compare MERT, Bayes risk, and PRO to the learning algorithms we present in Chapter 3.

2.3.7 Practical Issues

The computational complexity of inference affects how learning is done in practice. For example, most learning algorithms for machine translation do not work with the entire output space $\mathcal{T}_{\boldsymbol{x}^{(i)}}$ for each input $\boldsymbol{x}^{(i)}$. Rather, they call the decoder to generate k-best lists or packed representations like phrase lattices (Section 2.2.1) or hypergraphs, and perform several steps of optimization on these snippets of the search space. Then the decoder is called again to produce new k-best lists or lattices, which are typically merged with those from the previous iterations, and the process repeats until convergence or until a set number of iterations is reached.[5]

In this thesis we follow this trend, and in particular we restrict our attention to k-best lists. While lattices and hypergraphs have been used with success in the past for certain algorithms (Macherey et al., 2008; Chiang et al., 2009; Kumar et al., 2009; Li and Eisner, 2009; Cherry and Foster, 2012; Chiang, 2012), k-best lists are more commonly used by practitioners and therefore by focusing on k-best lists we hope to make our results as broadly useful as possible. Also, using lattices and hypergraphs with training frequently places limitations on cost functions that can be used.[6] For most lattice implementations of learning algorithms, the cost function must decompose across the edges in the lattice, just like the feature functions. This precludes the use of

[5] Notable exceptions to this pattern use Monte Carlo sampling to perform learning while taking the full output space into account (Arun et al., 2009, 2010a,b).

[6] Exceptions are lattice- and hypergraph-based MERT (Kumar et al., 2009), which can use BLEU unmodified.

any cost function involving BLEU. Approximations of BLEU have been developed that *do* factor, but other issues arise. For example, Tromble et al. (2008) proposed a BLEU approximation, but it requires tuning hyperparameters for each language pair; in preliminary experiments we found performance to be sensitive to their values. To avoid clouding performance differences with this additional confound, we simply use k-best lists for all training experiments in this thesis.

We note that our analysis in this thesis is applicable for understanding the loss function being optimized *given a fixed set of k-best lists*. Cherry and Foster (2012) performed a similar analysis based on empirical risk minimization. However, when training procedures invoke the decoder to generate new k-best lists and merge them with those from previous training iterations, it is unclear how this affects the function being optimized by the procedure as a whole. For example, even if the loss function is convex for a fixed set of k-best lists, the overall learning algorithm will be solving a non-convex problem that will depend on the initial values for $\boldsymbol{\theta}$.

Another practical issue relates to the amount of data used for learning. While parallel data is plentiful for several language pairs, it is typically too computationally-expensive to use all of it for learning $\boldsymbol{\theta}$. Furthermore, the data is not all of the same quality or the same value for the task of interest. Typically, a small subset (500 to 2,500 sentence pairs) is chosen for learning $\boldsymbol{\theta}$. This data is ensured to be of high quality and is usually drawn from the same domain as the test data. This is the most common approach taken in the literature and all of the learning experiments in this thesis follow this trend. There are, however, notable exceptions (Liang et al., 2006a; Blunsom and Osborne, 2008; Kääriäinen, 2009), including some that have achieved impressive performance improvements by scaling learning to large data sets (Wuebker et al., 2010; Simianer et al., 2012).

2.4 Summary

In this chapter we included introductory and background information needed for the rest of this thesis. In Chapter 3 we present our contributions to solving the learning problem for machine translation; we give new loss functions and algorithms and empirically compare them to the state-of-the-art for five language pairs. In Chapter 4, we present our contributions to translation modeling, including a new tree-to-tree translation model based on quasi-synchronous grammar. We present experiments with our model in Chapter 5. In the rest of the thesis, we present learning before modeling because our learning algorithms are applicable to any translation model, and we use our learning algorithms for training the models in Chapter 4.

Chapter 3

Structured Ramp Loss Minimization for Machine Translation

This chapter describes our contributions to learning in machine translation. We propose minimizing the **structured ramp loss** (Do et al., 2008), a loss function with attractive formal properties that has connections to several commonly-used MT learning algorithms. We begin in Section 3.1 by motivating our focus on loss functions, and on ramp loss in particular. In Section 3.2 we present the structured ramp loss, introduce novel variants, and give simple algorithms for minimizing each variant for machine translation. We draw connections between ramp loss and other MT learning algorithms—including the perceptron, the margin-infused relaxed algorithm, conditional likelihood, and Bayes risk—in Section 3.3. We report on experiments in Sections 3.4 and 3.5. Our algorithms are conceptually straightforward, easy to implement, and effective in practice in both small-feature and large-feature settings.

This chapter includes and expands on material originally presented in Gimpel and Smith (2012a) and, to a lesser extent, Gimpel and Smith (2010a,b, 2011b). Appendix A contains some additional exposition and proofs relating to softmax-margin (Gimpel and Smith, 2010a).

3.1 Introduction

In Chapter 2 we presented the learning framework of **empirical risk minimization** and used it to describe commonly-used learning algorithms for machine translation, including **minimum error rate training** (Och, 2003), **Bayes risk minimization** (Smith and Eisner, 2006b), and **pairwise ranking optimization** (Hopkins and May, 2011). We did so by identifying the **loss functions** optimized by each algorithm. Identifying underlying loss functions helps us to better understand and compare algorithms, and to understand the guarantees associated with particular algorithms and their variants.

While not all learning algorithms correspond to the optimization of some function, many well-known algorithms do, even if they were not originally developed with a particular function in mind. Yet it is not always easy to identify the function optimized by an algorithm. In this chapter,

we show that this is frequently the case in machine translation. For example, well-known machine learning algorithms such as the **structured perceptron** (Collins, 2002), the **margin-infused relaxed algorithm** (MIRA; Crammer et al., 2006), and **conditional likelihood** training (Lafferty et al., 2001) can all be shown to minimize particular loss functions. However, the loss functions changed when these algorithms were adapted for MT, due to the presence of latent variables and the potential unreachability of the references (Section 2.3.1). This makes theoretical guarantees of the original algorithms uncertain and clouds what is actually being done by a particular MT learning algorithm.

McAllester and Keshet (2011) suggested a connection between certain MT learning algorithms (Liang et al., 2006a; Chiang et al., 2009) and particular forms of the **structured ramp loss**. We give these forms of ramp loss and present these connections in greater detail. We also introduce new forms of ramp loss inspired by extant MT learning algorithms. We aim to distill the loss functions implicit in particular MT learning algorithms and to design simple algorithms for optimizing them directly that have theoretical guarantees.

The remainder of this chapter is laid out as follows. We present the ramp losses and our algorithms in the next section, then show how they relate to MT learning algorithms in Section 3.3. We conduct experiments comparing the ramp losses to one another and to state-of-the-art learning algorithms in Sections 3.4 and 3.5.

3.2 Structured Ramp Losses

The **structured ramp loss** (Do et al., 2008) is a non-convex loss function with certain attractive theoretical properties. It is an upper bound on $\text{loss}_{\text{cost}}$ (Eq. 2.14) and is a tighter bound than most other commonly-used loss functions, such as the perceptron, hinge, and log losses (discussed below) (Collobert et al., 2006). Ramp loss has been shown to be **statistically consistent** in the sense that, in the limit of infinite training data, minimizing structured ramp loss reaches the minimum value of $\text{loss}_{\text{cost}}$ that is achievable with a linear model (McAllester and Keshet, 2011). This is true whether or not latent variables are present.

Consistency in this sense is not a common property of loss functions. Commonly-used convex loss functions such as the perceptron, hinge, and log losses are *not* consistent, because they are all sensitive to outliers or otherwise noisy training examples. Ramp loss is better at dealing with outliers in the training data (Collobert et al., 2006). For MT, the ramp losses are appealing because they do not require computing the feature vector of $\boldsymbol{y}^{(i)}$, so they can be used even when $\boldsymbol{y}^{(i)}$ is not reachable by the model (Section 2.3.1).

In the rest of this section, we present three forms of structured ramp loss (Section 3.2.1), give three additional forms based on "softening" transformations (Section 3.2.2), and then give algorithms to minimize all six ramp losses for machine translation (Section 3.2.3).

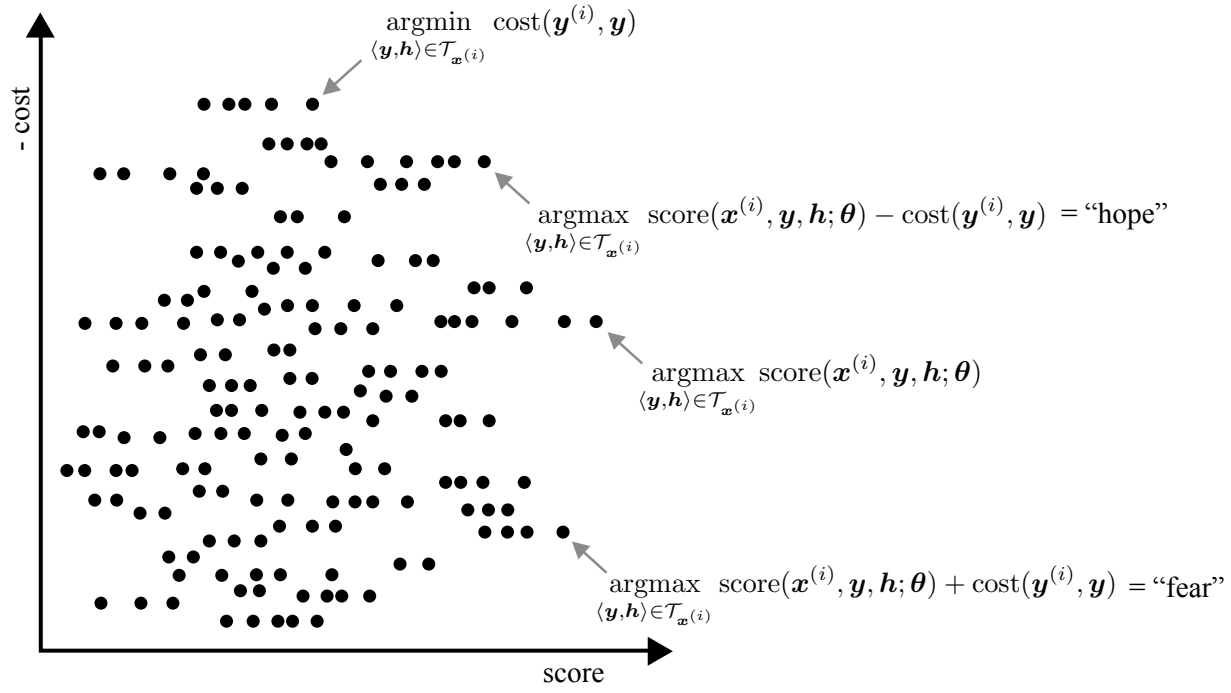


Figure 3.1: Hypothetical output space of a translation model for an input sentence $\boldsymbol{x}^{(i)}$. Each point corresponds to a single $\langle \boldsymbol{y}, \boldsymbol{h} \rangle$ pair. Horizontal "bands" are caused by output pairs with the same translation (and hence the same cost) but different derivations. Certain outputs are marked, including the so-called "hope" and "fear" translations, which have high model score as well as low and high cost, respectively.

3.2.1 Three Forms

We give three forms of latent structured ramp loss; two are from prior work and the third is a novel loss function inspired by extant MT learning algorithms (Chiang et al., 2008b, 2009). Figure 3.1 gives a general visualization of some of the key output pairs in the ramp losses. Learning alters the score function, or, in the figure, moves points *horizontally* so that scores approximate negated costs.

The first we present was first used in a structured setting by Do et al. (2008):

$$\text{loss}_{\text{ramp 1}} = -\max_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) + \max_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}') \Big) \quad (3.1)$$

This loss is minimized by boosting the score of the model's current prediction (the first max) and penalizing the score of an output that is both favored by the model *and* has high cost (the second

max). In MT this latter output is often called the "fear translation" (Chiang, 2012). Finding such an output is often called **cost-augmented decoding**; this term originated with the development of the structured hinge loss (Taskar et al., 2003), which we discuss in Section 3.3.2 below.

The second form of the latent structured ramp loss follows:

$$\text{loss}_{\text{ramp 2}} = -\max_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \Big) + \max_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta}) \quad (3.2)$$

This form *penalizes* the model prediction and favors an output pair that has both high model score and low cost, the so-called "hope translation." Finding this translation is the converse of cost-augmented decoding and therefore we call it **cost-*diminished* decoding**.

This second form of ramp loss was used by McAllester et al. (2010), who noted its similarity to the perceptron-like algorithm of Liang et al. (2006a). We return to this point in Section 3.3.1 below. McAllester et al. (2010) found $\text{loss}_{\text{ramp 2}}$ to work better empirically than $\text{loss}_{\text{ramp 1}}$, which is consistent with our experimental results in Section 3.5. Current theoretical results are richer for $\text{loss}_{\text{ramp 1}}$ than for $\text{loss}_{\text{ramp 2}}$, but this is still an active area of research (McAllester and Keshet, 2011).

The third form is inspired by the algorithms of Chiang et al. (2008b, 2009):

$$\text{loss}_{\text{ramp 3}} = -\max_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \text{cost}_i(\boldsymbol{y}) \Big) + \max_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta}) + \text{cost}_i(\boldsymbol{y}') \Big) \quad (3.3)$$

where we have used $\text{cost}_i(\boldsymbol{y})$ as shorthand for $\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$. This third ramp loss encourages the hope translation and penalizes the fear translation; the model prediction does not appear in the loss. We extracted this loss from the MIRA-like algorithms developed by Chiang et al.; we discuss those algorithms further in Section 3.3.2. In addition, we can see immediately that $\text{loss}_{\text{ramp 3}} = \text{loss}_{\text{ramp 1}} + \text{loss}_{\text{ramp 2}}$.

Each ramp loss is non-convex, being the difference of two convex functions. To compute and optimize the ramp losses, we never have to compute the score of the reference translation $\boldsymbol{y}^{(i)}$. It is only accessed through the cost function. We discuss how to optimize the ramp losses for machine translation in Section 3.2.3.

3.2.2 Softened Ramp Losses

We can derive new loss functions from the above by converting any "max" operator to a "softmax" ($\log \sum \exp$, where the set of elements under the summation is the same as under the max) (Gimpel and Smith, 2010a).

Softening the three ramp losses results in the following **soft ramp losses**:

$$\text{loss}_{\text{softramp 1}} = - \log \sum_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta})\} + \log \sum_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta}) + \text{cost}_i(\boldsymbol{y}')\} \tag{3.4}$$

$$\text{loss}_{\text{softramp 2}} = - \log \sum_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \text{cost}_i(\boldsymbol{y})\} + \log \sum_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta})\} \tag{3.5}$$

$$\text{loss}_{\text{softramp 3}} = - \log \sum_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \text{cost}_i(\boldsymbol{y})\} + \log \sum_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta}) + \text{cost}_i(\boldsymbol{y}')\} \tag{3.6}$$

where we have again used $\text{cost}_i(\boldsymbol{y})$ as shorthand for $\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$. As with the hard ramp losses, $\text{loss}_{\text{softramp 3}} = \text{loss}_{\text{softramp 1}} + \text{loss}_{\text{softramp 2}}$.

Why do we introduce these softened losses? One motivation is differentiability: the three ramp losses are not differentiable, but the three soft ramp losses are. Another motivation is that these softened losses have clearer relationships to loss functions like Bayes risk (Section 2.3.5) and conditional likelihood (Lafferty et al., 2001). We return to this in Sections 3.3.3 and 3.3.4.

3.2.3 Learning Algorithms

We now present algorithms to optimize each of the ramp losses for machine translation. Aside from high performance, our goals are conceptual simplicity and ease of implementation. In designing these algorithms, we follow the practice mentioned in Section 2.3.7 of using k-best lists as an approximation to the full output space, merging k-best lists across iterations to improve the approximation as learning proceeds.

Ramp Losses 1-3

The three (hard) ramp losses are continuous but non-convex and non-differentiable, so gradient-based optimization methods are not available. For non-differentiable, continuous, *convex* functions, **subgradient**-based methods can be used, such as stochastic subgradient descent (SSD), and it is tempting to apply them here. However, non-convex functions are not everywhere subdifferentiable and so a straightforward application of SSD may encounter problems in practice.

Fortunately, we can minimize the ramp losses using a **concave-convex procedure** (CCCP; Yuille and Rangarajan, 2002). CCCP is a batch optimization algorithm for any function that is the sum of a concave and a convex function, as are each of the ramp losses. The idea is to approximate the sum as the convex term plus a tangent line to the concave function at the current parameter values; the resulting sum is convex and can be optimized with (sub)gradient methods.

Input: inputs $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$, references $\{\boldsymbol{y}^{(i)}\}_{i=1}^{N}$, initial weights $\boldsymbol{\theta}^{(0)}$, k-best list size k, step size η, ℓ_2 regularization coefficient C, number of iterations T, number of CCCP iterations T', number of SSD iterations T''
Output: learned weights: $\boldsymbol{\theta}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$;
for $i \leftarrow 1$ **to** N **do**
  $\mathcal{K}_i \leftarrow \emptyset$;
end
// run algorithm for T iterations (extra iteration is for computing BLEU)
for $t \leftarrow 0$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    // accumulate k-best lists across iterations, discarding duplicates
    $\mathcal{K}_i \leftarrow \mathcal{K}_i \cup \texttt{Decode}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, k)$;
    $\hat{\boldsymbol{y}}_i \leftarrow \texttt{Decode}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, 1)$; // save the 1-best for computing BLEU
  end
  $b_t \leftarrow \text{BLEU}(\{\boldsymbol{y}^{(i)}\}_{i=1}^{N}, \{\hat{\boldsymbol{y}}_i\}_{i=1}^{N})$; // compute BLEU
  for $t' \leftarrow 1$ **to** T' **do** // run CCCP for T' iterations
    for $i \leftarrow 1$ **to** N **do** // impute hope translations for all sentences
      $\langle \boldsymbol{y}_i^+, \boldsymbol{h}_i^+ \rangle \leftarrow \operatorname*{argmax}_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{K}_i} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \mathbb{I}\left[\text{loss}_{\text{ramp 2}} \vee \text{loss}_{\text{ramp 3}}\right] \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$;
    end
    for $t'' \leftarrow 1$ **to** T'' **do** // run SSD for T'' epochs
      for $i \leftarrow 1$ **to** N **do**
        // find fear translation
        $\langle \boldsymbol{y}^-, \boldsymbol{h}^- \rangle \leftarrow \operatorname*{argmax}_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{K}_i} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) + \mathbb{I}\left[\text{loss}_{\text{ramp 1}} \vee \text{loss}_{\text{ramp 3}}\right] \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$;
        $\boldsymbol{\theta} \mathrel{-}= \eta C \left(\frac{\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)}}{N}\right)$; // regularize back to $\boldsymbol{\theta}^{(0)}$, not $\mathbf{0}$
        $\boldsymbol{\theta} \mathrel{+}= \eta \left(\boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}_i^+, \boldsymbol{h}_i^+) - \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^-, \boldsymbol{h}^-)\right)$; // subgradient update for loss
      end
    end
  end
  $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}$;
end
$t^* \leftarrow \text{argmax}_{0 \le t \le T}\, b_t$; // return the parameters that led to the highest BLEU
return $\boldsymbol{\theta}^{(t^*)}$;

Algorithm 1: Algorithm for minimizing $\text{loss}_{\text{ramp 1}}$, $\text{loss}_{\text{ramp 2}}$, or $\text{loss}_{\text{ramp 3}}$. The expression $\mathbb{I}\left[\text{loss}_{\text{ramp 2}} \vee \text{loss}_{\text{ramp 3}}\right]$ is 1 if either $\text{loss}_{\text{ramp 2}}$ or $\text{loss}_{\text{ramp 3}}$ is being used, and 0 otherwise.

CCCP has an intuitive form when used with the hard ramp losses. On each iteration, the negated max expressions (the concave terms in each loss) are solved for the entire data set. For

ramp losses 2 and 3, this means finding "hope" translations for the sentences in the tuning data. Given the hope translations fixed, the loss is convex and any convex optimization procedure can be used. We use SSD, but MIRA or a batch algorithm could be easily used instead. After the convex optimizer is run, the negated max expression are solved again with the updated parameters, and the process repeats.

Our algorithm is shown as Algorithm 1. The first step done on each iteration is to generate k-best lists for the full tuning set and merge them with those from previous iterations (line 7). We then run CCCP on the k-best lists for T' iterations (lines 11–22). This involves first finding the translation to update towards for all sentences in the tuning set (lines 12–14), then making parameter updates in an online fashion with T'' epochs of stochastic subgradient descent (lines 15–21). The subgradient update for the ℓ_2 regularization term is done in line 18 and then for the loss in line 19.[1]

We store the BLEU score achieved on the data after each iteration of the algorithm, and return the parameters that maximize BLEU. The BLEU scores we record are obtained by running the decoder using the given parameters and taking the 1-best outputs; we do *not* record BLEU scores obtained from reranking the *merged* k-best lists with the new parameters, because these scores do not take search error into account. That is, we want to simulate how the system would perform when using these parameters on new data using its approximate search, and to choose the parameters that give the best BLEU score under this setting.

Unlike prior work that targeted similar loss functions (Watanabe et al., 2007; Chiang et al., 2008b, 2009), we do not use a fully online algorithm such as MIRA in an outer loop because we are not aware of an online learning algorithm with theoretical guarantees for non-differentiable, non-convex loss functions like the ramp losses. CCCP is fundamentally a batch optimization algorithm and has been used for solving many non-convex learning problems, such as that found when training latent structured support vector machines (Yu and Joachims, 2009).

Soft Ramp Losses 1-3

To minimize the soft ramp losses we use Algorithm 2. CCCP could be used, but for simplicity we simply use stochastic gradient descent (SGD) directly.

The gradient of each soft ramp loss is a difference of expectations. For example, the derivative of $\text{loss}_{\text{softramp 1}}$ with respect to a single parameter θ_j is:

$$\frac{\partial \text{loss}_{\text{softramp 1}}}{\partial \theta_j} = -\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y},\boldsymbol{h}|\boldsymbol{x}^{(i)})}\left[f_j(\boldsymbol{x}^{(i)},\boldsymbol{y},\boldsymbol{h})\right] + \mathbb{E}_{p_{\boldsymbol{\theta}}^{+\text{cost}}(\boldsymbol{y},\boldsymbol{h}|\boldsymbol{x}^{(i)})}\left[f_j(\boldsymbol{x}^{(i)},\boldsymbol{y},\boldsymbol{h})\right]$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{y},\boldsymbol{h} \mid \boldsymbol{x}^{(i)})$ is the log-linear probability distribution from Eq. 2.4 and

$$p_{\boldsymbol{\theta}}^{+\text{cost}}(\boldsymbol{y},\boldsymbol{h} \mid \boldsymbol{x}^{(i)}) = \frac{\exp\{\text{score}(\boldsymbol{x}^{(i)},\boldsymbol{y},\boldsymbol{h};\boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)},\boldsymbol{y})\}}{\sum_{\langle \boldsymbol{y}',\boldsymbol{h}'\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)},\boldsymbol{y}',\boldsymbol{h}';\boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)},\boldsymbol{y}')\}}$$

[1] ℓ_2 regularization done here regularizes toward $\boldsymbol{\theta}_0$, not $\mathbf{0}$.

Input: inputs $\{\boldsymbol{x}^{(i)}\}_{i=1}^N$, references $\{\boldsymbol{y}^{(i)}\}_{i=1}^N$, initial weights $\boldsymbol{\theta}^{(0)}$, k-best list size k, step size η, ℓ_2 regularization coefficient C, number of iterations T, number of SSD iterations T'
Output: learned weights: $\boldsymbol{\theta}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$;
for $i \leftarrow 1$ **to** N **do**
  $\mathcal{K}_i \leftarrow \emptyset$;
end
// run algorithm for T iterations (extra iteration is for computing BLEU)
for $t \leftarrow 0$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    // accumulate k-best lists across iterations, discarding duplicates
    $\mathcal{K}_i \leftarrow \mathcal{K}_i \cup \texttt{Decode}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, k)$;
    $\hat{\boldsymbol{y}}_i \leftarrow \texttt{Decode}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, 1)$; // save the 1-best for computing BLEU
  end
  $b_t \leftarrow \text{BLEU}(\{\boldsymbol{y}^{(i)}\}_{i=1}^N, \{\hat{\boldsymbol{y}}_i\}_{i=1}^N)$; // compute BLEU
  for $t' \leftarrow 1$ **to** T' **do** // run SSD for T' epochs
    for $i \leftarrow 1$ **to** N **do** // go through all sentences
      $\boldsymbol{\nabla}^+, \boldsymbol{\nabla}^- \leftarrow \mathbf{0}$; // will hold numerators in expectations
      $Z^+, Z^- \leftarrow 0$; // will hold denominators in expectations
      // compute expectations over entries in k-best list
      foreach $\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{K}_i$ **do**
        $\text{curr}^+ \leftarrow \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \mathbb{I}\left[\text{loss}_{\text{softramp 2}} \vee \text{loss}_{\text{softramp 3}}\right] \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\}$;
        $\boldsymbol{\nabla}^+ \mathrel{+}= \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}) \text{curr}^+$;
        $Z^+ \mathrel{+}= \text{curr}^+$;
        $\text{curr}^- \leftarrow \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) + \mathbb{I}\left[\text{loss}_{\text{softramp 1}} \vee \text{loss}_{\text{softramp 3}}\right] \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\}$;
        $\boldsymbol{\nabla}^- \mathrel{+}= \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}) \text{curr}^-$;
        $Z^- \mathrel{+}= \text{curr}^-$;
      end
      $\boldsymbol{\nabla} \leftarrow \frac{\boldsymbol{\nabla}^+}{Z^+} - \frac{\boldsymbol{\nabla}^-}{Z^-}$; // (negated) gradient is difference of expectations
      $\boldsymbol{\theta} \mathrel{-}= \eta C \left(\frac{\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)}}{N}\right)$; // regularize back to $\boldsymbol{\theta}^{(0)}$, not $\mathbf{0}$
      $\boldsymbol{\theta} \mathrel{+}= \eta \boldsymbol{\nabla}$; // gradient update for loss
    end
  end
  $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}$;
end
$t^* \leftarrow \text{argmax}_{0 \le t \le T} b_t$; // return the parameters that led to the highest BLEU
return $\boldsymbol{\theta}^{(t^*)}$;

Algorithm 2: Algorithm for minimizing $\text{loss}_{\text{softramp 1}}$, $\text{loss}_{\text{softramp 2}}$, or $\text{loss}_{\text{softramp 3}}$. The expression $\mathbb{I}\left[\text{loss}_{\text{softramp 2}} \vee \text{loss}_{\text{softramp 3}}\right]$ is 1 if either $\text{loss}_{\text{softramp 2}}$ or $\text{loss}_{\text{softramp 3}}$ is being used, and 0 otherwise.

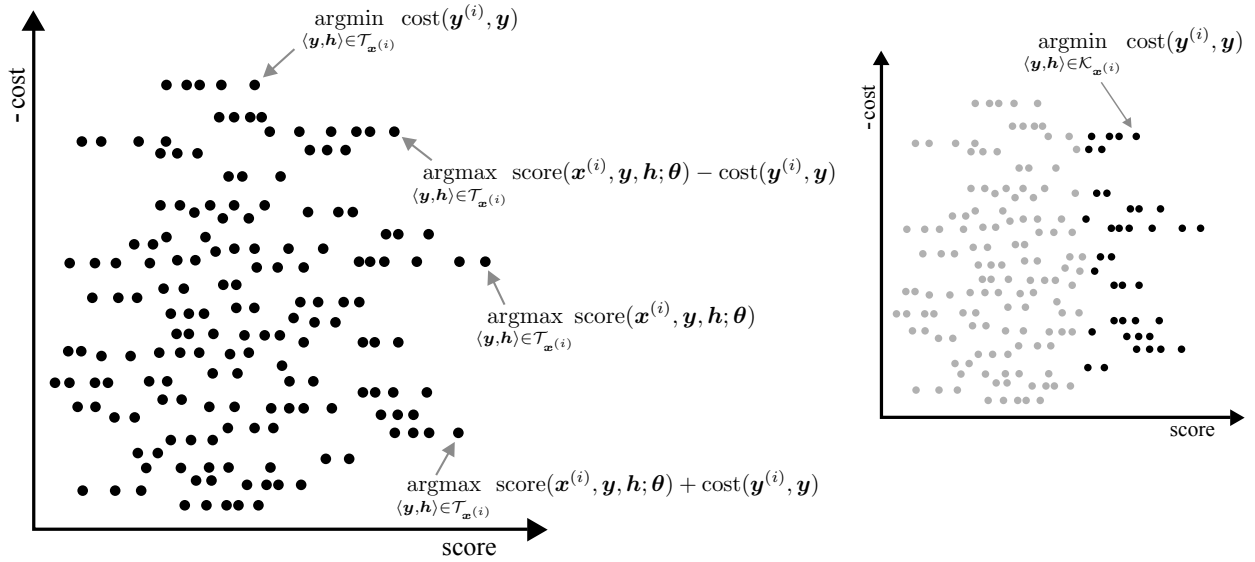


Figure 3.2: Hypothetical translation output space from Figure 3.1, along with another plot highlighting outputs in the k-best list. Choosing the output with the lowest cost in the k-best list is similar to finding the "hope" translation in the full space.

When using this formula within gradient descent, we seek to decrease the weights of features that have larger expected values under the "cost-augmented" distribution $p_{\boldsymbol{\theta}}^{+\text{cost}}(\boldsymbol{y}, \boldsymbol{h} \mid \boldsymbol{x}^{(i)})$ than under the ordinary distribution without the cost function. The effect is that features that appear frequently in high-cost structures are penalized more heavily than features that mostly appear in low-cost structures. The gradients for the other two soft ramp losses can be similarly derived. In Algorithm 2, these expectations are computed for each entry on the k-best list in lines 13–23. The rest of the algorithm is similar to Algorithm 1, minus the CCCP loop.

3.3 Relationship to Other Loss Functions

In the previous section, we presented six forms of latent structured ramp loss. We now relate the ramp losses to other structured losses used in prior work on learning for MT. We begin with the structured perceptron (Collins, 2002) in Section 3.3.1. We discuss connections to large-margin learning, including structured support vector machines and the margin-infused relaxed algorithm (MIRA), in Section 3.3.2. We relate the soft ramp losses to conditional likelihood in Section 3.3.3 and Bayes risk in Section 3.3.4.

3.3.1 Structured Perceptron

The structured perceptron algorithm (Collins, 2002) was considered by Liang et al. (2006a) as an alternative to MERT. It is a simple online algorithm that requires only a decoder; some convergence guarantees and error bounds on held-out data are available (Freund and Schapire, 1999; Collins, 2002; Sun et al., 2009). Liang et al. modified the perceptron in several ways for use in MT, as we now review.

We begin with the loss function optimized by the perceptron algorithm. Though typically described and analyzed procedurally, it is straightforward to show that Collins' perceptron (without latent variables) equates to SSD with fixed step size 1 on the following loss, which we call the **perceptron loss**:

$$\text{loss}_{\text{perc}} = -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \max_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{\theta}) \tag{3.7}$$

This loss is convex but ignores cost functions.

To adapt it for MT, Liang et al. (2006a) began by adding latent variables to the algorithm. Adding latent variables to the loss gives us the following **latent perceptron loss**:

$$\text{loss}^{\text{latent}}_{\text{perc}} = -\max_{\{\boldsymbol{h}' : \langle \boldsymbol{y}^{(i)}, \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}\}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{h}'; \boldsymbol{\theta}) + \max_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) \tag{3.8}$$

This loss is a difference of two convex functions and is therefore no longer convex. So the interpretation of the perceptron update as SSD no longer applies, due to the non-subdifferentiability of the loss. However, Sun et al. (2009) generalized the perceptron algorithm to handle latent variables. Since it is not provided in the training data, the latent derivation $\boldsymbol{h}'$ must be imputed for the true output $\boldsymbol{y}^{(i)}$ before each update. While their algorithm does not have an interpretation as SSD for any particular loss, they established convergence guarantees and derived mistake bounds on held-out data. It is similar to the application of CCCP to the loss in Eq. 3.8, except for the difference between batch and online optimization. CCCP relies on its batch optimization for a guarantee that it will reach a local optimum of its objective. Sun et al. 's algorithm has convergence guarantees, but it is not guaranteed to find a local optimum of Eq. 3.8.

After adding latent variables, Liang et al. still needed to cope with unreachable reference translations. In place of the reference, they used a surrogate translation, namely the translation on the k-best list with the highest BLEU score:

$$\text{loss}^{\text{latent}}_{\text{perc } k\text{best}} = -\text{score}\left(\boldsymbol{x}^{(i)}, \operatorname*{argmin}_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{K}_{\boldsymbol{x}^{(i)}}} \left(\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\right); \boldsymbol{\theta}\right) + \max_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) \tag{3.9}$$

where $\mathcal{K}_{\boldsymbol{x}^{(i)}} \subseteq \mathcal{T}_{\boldsymbol{x}^{(i)}}$ is a set containing the k best output pairs for $\boldsymbol{x}^{(i)}$. Now, the loss only depends on $\boldsymbol{y}^{(i)}$ through the cost function. Even without hidden variables, this loss can only be convex when the k-best list is fixed, keeping the surrogate unchanged across iterations. Updating the k-best lists makes the choice of the surrogate depend on $\boldsymbol{\theta}$, resulting in a non-convex loss.

It appears that Eq. 3.9 is the loss that Liang et al. (2006a) *sought* to optimize, using SSD. But since the loss is non-convex, we have no theoretical guarantee that SSD will find a (local) optimum.

We note that Eq. 3.9 is similar to

$$-\max_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \gamma_i \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \Big) + \max_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) \tag{3.10}$$

where the first $\max$ finds an output that is both favored by the model and has low cost and each γ_i is used to trade off between model and cost. Figure 3.2 illustrates the similarity by showing that the min-cost output on a k-best list resides in a similar region of the output space as the "hope" translation computed from the full output space.

While it is not the case that we can always choose γ_i so as to make Eq. 3.10 equivalent to Eq. 3.9, the two losses are similar in that they update towards an output with high model score and low cost. Eq. 3.10 is very nearly $\text{loss}_{\text{ramp 2}}$ (Eq. 3.2), suggesting a similarity between Liang et al.'s algorithm and the second form of the structured ramp loss. Another interpretation is given by McAllester et al. (2010), who showed that making updates based on computing subgradients of each term in Eq. 3.10 approaches direct cost minimization (i.e., minimizing $\text{loss}_{\text{cost}}$, Eq. 2.14) in the limiting case as each γ_i goes to zero.

3.3.2 Structured Max-Margin

A related family of approaches for training MT models involves the **margin-infused relaxed algorithm** (MIRA; Crammer et al., 2006), an online large-margin training algorithm. It has recently shown success for MT, particularly when training models with large feature sets (Watanabe et al., 2007; Chiang et al., 2008b, 2009). In order to apply it to MT, Watanabe et al. and Chiang et al. made modifications similar to those made by Liang et al. for perceptron training, namely the extension to latent variables and the use of a surrogate reference with high model score and low cost.

We begin with the loss function underlying MIRA. It has been shown that (1-best) MIRA corresponds to dual coordinate ascent for the **structured hinge loss** when using ℓ_2 regularization (Hsieh et al., 2008; Martins et al., 2010). The structured hinge is the loss underlying maximum-margin Markov networks (Taskar et al., 2003); the "margin rescaling" version without latent variables follows:[2]

$$\text{loss}_{\text{hinge}} = -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \max_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \Big) \tag{3.11}$$

Unlike the perceptron losses, which penalize the highest-scoring outputs, hinge loss penalizes an output that is both favored by the model *and* has high cost, i.e., the "fear translation" in MT. Structured hinge loss is convex, can incorporate a cost function, and can be optimized with several algorithms, including SSD (Ratliff et al., 2006).

Although MIRA optimizes the structured hinge loss and prior work has used MIRA-like algorithms for training machine translation systems, it is unclear what loss function these algorithms actually optimized, for similar reasons to those mentioned above for the perceptron: latent variables and surrogate references.

[2] An alternative form is obtained via "slack rescaling" (Tsochantaridis et al., 2005), which we will not discuss further here.

Adding latent variables to the structured hinge results in the **latent structured hinge loss** which is the loss underlying the latent structured SVM (Yu and Joachims, 2009):

$$\text{loss}_{\text{hinge}}^{\text{latent}} = -\max_{\{\boldsymbol{h}:\langle \boldsymbol{y}^{(i)},\boldsymbol{h}'\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}\}} \text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{h}'; \boldsymbol{\theta}) + \max_{\langle \boldsymbol{y},\boldsymbol{h}\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\Big) \tag{3.12}$$

Like the latent perceptron loss, this loss is non-convex and inappropriate for MT because it requires computing the feature vector for $\boldsymbol{y}^{(i)}$. When using a surrogate in place of $\boldsymbol{y}^{(i)}$, following a similar argument as with the perceptron, the actual loss optimized by these MIRA-like algorithms is more similar to $\text{loss}_{\text{ramp 3}}$ (Eq. 3.3) than to the latent hinge loss.

In choosing the surrogate, most have used k-best oracles like Liang et al. (Watanabe et al., 2007; Arun and Koehn, 2007; Cherry and Foster, 2012; Eidelman, 2012), but Chiang et al. (2008, 2009) used a different approach, explicitly defining the surrogate as the "hope" translation in Figure 3.1 and using hypergraph rescoring to find it.

While the method of Chiang et al. showed impressive performance improvements, its implementation is non-trivial, involving a complex cost function and a parallel architecture, and it has not yet been embraced by the MT community. Indeed, the complexity of Chiang et al's algorithm was one of the reasons cited for the development of PRO (Hopkins and May, 2011). In this chapter, we sought to isolate the loss functions used in prior work like that by Chiang et al. and others and to identify simple, generic optimization procedures for minimizing them. We propose our Algorithm 1 for $\text{loss}_{\text{ramp 3}}$ as an alternative to Chiang et al's MIRA that is simpler to implement and achieves empirical success in experiments (§3.5).

3.3.3 Softened Losses and Conditional Likelihood

In Section 3.2.2, we developed new ramp losses by converting each max to a "softmax" ($\log \sum \exp$). Apart from the ramp losses, softening can reveal connections among other loss functions, including those well-known to the machine learning community. For example, the softmax version of the perceptron loss is the well-known **log loss**, the loss underlying the **conditional likelihood** training criterion which is frequently used when a probabilistic interpretation of the learned model is desired, as in conditional random fields (Lafferty et al., 2001):

$$\text{loss}_{\log} = -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{\theta})\} \tag{3.13}$$

The softmax version of the latent perceptron loss (Eq. 3.8) is the **latent log loss** inherent in latent-variable CRFs (Quattoni et al., 2004):

$$\text{loss}_{\log}^{\text{latent}} = -\log \sum_{\substack{\{\boldsymbol{h}':\langle \boldsymbol{y}^{(i)},\boldsymbol{h}'\rangle \\ \in \mathcal{T}_{\boldsymbol{x}^{(i)}}\}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{h}'; \boldsymbol{\theta})\} + \log \sum_{\langle \boldsymbol{y},\boldsymbol{h}\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta})\} \tag{3.14}$$

Och and Ney (2002) popularized the use of log-linear models for MT and initially sought to optimize log loss, but faced the usual problem of reference reachability. In place of $\boldsymbol{y}^{(i)}$ they chose

the translation (and highest-scoring derivation) on a k-best list with the highest metric score. By doing so, we argue that their loss was closer to the loss obtained by softening *only* the second max in $\text{loss}_{\text{ramp 2}}$ in Eq. 3.2. The result is not $\text{loss}_{\text{softramp 2}}$, but rather the following new loss:

$$\text{loss}_{\text{hardsoftramp 2}} = \\ -\max_{\langle \boldsymbol{y},\boldsymbol{h}\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \Big(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta}) - \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\Big) + \log \sum_{\langle \boldsymbol{y}',\boldsymbol{h}'\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta})\} \quad (3.15)$$

The same is true for most others who aimed to optimize log loss for MT (Smith and Eisner, 2006b; Zens et al., 2007; Cer, 2011). An exception is Blunsom et al. (2008) and Blunsom and Osborne (2008), who did actually optimize latent log loss for MT, discarding training examples for which $\boldsymbol{y}^{(i)}$ was unreachable.

Similar loss functions can be generated by only softening one of the max expressions in $\text{loss}_{\text{ramp 1}}$ and $\text{loss}_{\text{ramp 3}}$. In initial experiments we tried minimizing these "hybrid" losses for machine translation, but we found that ramp losses in which both terms use a max or both terms use a softmax worked much better than those in which one max was softened and the other was not. These latter losses performed poorly, possibly due to the use of k-best lists in the optimization algorithm.

Finally, the softmax version of the hinge loss is the **softmax-margin loss** (Gimpel and Smith, 2010a), which is the generalization of a loss used by Povey et al. (2008) for speech recognition:

$$\text{loss}_{\text{softhinge}} = -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \quad (3.16)$$

Due to the need to compute the features for $\boldsymbol{y}^{(i)}$, softmax-margin is not suitable for machine translation, but it has been used effectively for other tasks, including named-entity recognition (Gimpel and Smith, 2010a), CCG parsing (Auli and Lopez, 2011), and congressional vote prediction (Stoyanov and Eisner, 2012). Softmax-margin training has a probabilistic interpretation, seen by situating it within the minimum divergence framework, which is a generalization of the well-known maximum entropy framework for learning (Jelinek, 1997). This connection, along with additional discussion of the properties of softmax-margin and a proof of convexity, is included in Appendix A.

3.3.4 Bayes Risk

There are also connections between the soft ramp losses and Bayes risk (Eq. 2.16, reproduced below):

$$\begin{aligned}\text{loss}_{\text{Bayes risk}} &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y},\boldsymbol{h}|\boldsymbol{x}^{(i)})}\Big[\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\Big] \\ &= \sum_{\langle \boldsymbol{y},\boldsymbol{h}\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \cdot \frac{\exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta})\}}{\sum_{\langle \boldsymbol{y}',\boldsymbol{h}'\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta})\}}\end{aligned}$$

In particular, $\text{loss}_{\text{softramp 1}}$ is an upper bound on $\text{loss}_{\text{Bayes risk}}$ and $\text{loss}_{\text{softramp 2}}$ is a lower bound on $\text{loss}_{\text{Bayes risk}}$. To see this, we rewrite $\text{loss}_{\text{softramp 1}}$ as

$$\begin{aligned}\text{loss}_{\text{softramp 1}} &= \log \frac{\sum_{\langle \boldsymbol{y}', \boldsymbol{h}' \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}', \boldsymbol{h}'; \boldsymbol{\theta}) + \text{cost}_i(\boldsymbol{y}')\}}{\sum_{\langle \boldsymbol{y}, \boldsymbol{h} \rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}; \boldsymbol{\theta})\}} \\ &= \log \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{h} | \boldsymbol{x}^{(i)})} \left[\exp\{\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \right]\end{aligned}$$

Since $\log$ is concave, we can use Jensen's inequality to obtain:

$$\log \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{h} | \boldsymbol{x}^{(i)})} \left[\exp\{\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \right] \geq \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{h} | \boldsymbol{x}^{(i)})} \left[\log \exp\{\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \right] = \text{loss}_{\text{Bayes risk}} \quad (3.17)$$

Therefore, soft ramp 1 is an upper bound on Bayes risk. An analogous argument via Jensen's can be used to show that soft ramp 2 is a lower bound.

Thus minimizing soft ramp 1 can be viewed as an approximation to Bayes risk minimization. This can be useful because minimizing Bayes risk can be computationally-expensive and require new algorithms to compute its gradient. When using k-best lists within optimization, it is easy to compute gradients for Bayes risk and all of the soft ramp losses, but in the general structured case, computing gradients for Bayes risk becomes harder. Consider the partial derivative of the Bayes risk loss for a single parameter θ_j:

$$\frac{\partial \mathbb{E}_{p_{\boldsymbol{\theta}}} \left[\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \right]}{\partial \theta_j} = \mathbb{E}_{p_{\boldsymbol{\theta}}} \left[f_j(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}) \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \right] - \mathbb{E}_{p_{\boldsymbol{\theta}}} \left[f_j(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}) \right] \mathbb{E}_{p_{\boldsymbol{\theta}}} \left[\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \right] \quad (3.18)$$

where we have used $p_{\boldsymbol{\theta}}$ as shorthand for $p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{h} | \boldsymbol{x}^{(i)})$. Compare Eq. 3.18 with the formula for computing the gradient of soft ramp 1, given in Eq. 3.7, which is similar to that of the other two soft ramp losses. The presence of expectations of products makes computing Eq. 3.18 difficult for structured models. Xiong et al. (2009) gave an algorithm for sequence labeling and Li and Eisner (2009) developed a novel semiring to compute expectations of products. But the gradients of the soft ramp losses can be computed using the same computational machinery as conditional likelihood maximization, namely a summing algorithm over the output space.[3]

Finally, since it is clear that $\text{loss}_{\text{softramp 3}} > \text{loss}_{\text{softramp 1}}$, $\text{loss}_{\text{softramp 3}}$ is also an upper bound on $\text{loss}_{\text{Bayes risk}}$ as well as an upper bound on $\text{loss}_{\text{softramp 1}}$ and $\text{loss}_{\text{softramp 2}}$.

3.4 Experimental Setup

We now turn to an experimental comparison of the loss functions and learning algorithms discussed in this thesis. We compare these loss functions to one another and to state-of-the-art methods for training MT systems, in both small- and large-feature settings. In Section 3.4.1 we discuss

[3] This discussion assumes that the cost function factors over the structures in the same way as the features. If not, all loss functions that use a cost function require approximate inference in the structured case; when using k-best lists, any cost function that decomposes across the instances can be used.

the language pairs and MT systems used in these experiments. Our baseline training methods are discussed in Section 3.4.2. The experimental results are presented in Section 3.5.

3.4.1 Language Pairs and Systems

We use both phrase-based models (Koehn et al., 2003) and hierarchical phrase-based models (Chiang, 2005) as implemented in the Moses toolkit (Koehn et al., 2007; Hoang et al., 2009). We consider Arabic→English (AR→EN), Chinese→English (ZH→EN), German→English (DE→EN), Urdu→English (UR→EN), and English→Malagasy (EN→MG) translation.

Full dataset details are provided in Appendix C. Briefly, AR→EN is a large-data scenario with 4.3M sentence pairs of news and UN data, ZH→EN is a medium-data scenario with 300k sentence pairs of data from the FBIS corpus (LDC2003E14), DE→EN is a large-data scenario with 850k sentence pairs of Europarl and 150k sentence pairs of news commentary parallel data, UR→EN is a small-data scenario using the data from the NIST MT08 evaluation consisting of approximately 1M tokens in each language, and EN→MG is another small-data scenario with a low-resource target language. EN→MG uses a language model trained only on the target side of the parallel corpus, whereas all other language pairs use 600M words of additional data from the Gigaword v4 corpus. We use a single tuning set and multiple test sets for each language pair, chosen from standard test sets for these language pairs. All language pairs use four references except DE→EN and EN→MG which each use one. Details of tuning and test sets are in Appendix C.

For phrase-based models, we mostly used default settings and features, including the default lexicalized reordering model. Word alignment was performed using GIZA++ (Och and Ney, 2003) in both directions, the `grow-diag-final-and` heuristic was used to symmetrize the alignments, and a max phrase length of 7 was used for phrase extraction. The only exception to the defaults was setting the distortion limit to 10 in all experiments. For hierarchical phrase-based models, we used default settings and features except for setting the `max-chart-span` parameter to 10.

We estimated 5-gram language models using the SRI toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998). The minimum count cut-off for unigrams, bigrams, and trigrams was one and the cut-off for four-grams and five-grams was three. We used KenLM (Heafield, 2011) for language model inference. For all languages we evaluated translation output using case-insensitive IBM BLEU (Papineni et al., 2001).

3.4.2 Baselines

Our baselines are MERT, Bayes risk minimization, and PRO. We used the MERT and PRO implementations in the Moses toolkit (Koehn et al., 2007). MERT used k-best lists of size 100 and was run to convergence. For all algorithms, previous iterations' k-best lists were merged.

For PRO we used the hyperparameter settings from Hopkins and May (2011), including k-best lists of size 1500. The Moses implementation uses 25 training iterations by default, so we kept this value. While Hopkins and May used 30 training iterations for PRO, they showed that BLEU scores had generally converged by 25 iterations. Hopkins and May did not discuss regularization, but they used the logistic regression classifier implemented in the MegaM package (Daumé

III, 2008), which uses a default regularization strength of $C = 1$. So we also used $C = 1$ in all PRO experiments. Since we used a different decoding pipeline for large-feature experiments (Section 3.5.3), we used our own implementation of PRO for those, which uses the logistic regression implementation in the Classias toolkit (Okazaki, 2009).

We minimized Bayes risk using stochastic gradient descent. In particular, we used a modified version of Algorithm 2 in which the gradient is computed for Eq. 2.16 instead of for the soft ramp losses. The gradient is straightforward to compute when k-best lists are used. The hyperparameters in Algorithm 2 were the same as those used for the soft ramp losses, described in the next section.

3.4.3 Hyperparameter Settings

To minimize the ramp losses, we need to set several hyperparameters and determine our cost function. We first discuss the cost function. We follow Hopkins and May (2011) in using the BLEU$_{+1}$ approximation from Lin and Och (2004) for a smoothed sentence-level BLEU score, defining cost as

$$\text{cost}(\boldsymbol{y}, \boldsymbol{y}') = \alpha(1 - \text{BLEU}_{+1}(\boldsymbol{y}, \boldsymbol{y}')) \tag{3.19}$$

where BLEU$_{+1}(\boldsymbol{y}, \boldsymbol{y}')$ returns the BLEU$_{+1}$ score (Lin and Och, 2004) for reference $\boldsymbol{y}$ and hypothesis $\boldsymbol{y}'$ (see Section 2.3.2), and α is a hyperparameter of the learning algorithm. We experimented with various values of α and chose the best for each loss function based on tuning BLEU score for the UR→EN phrase-based translation task. These experiments are discussed below in Section 3.5.1.

We note that the cost function could use any sentence-level evaluation metric for machine translation, since cost is exclusively computed on complete translations from a k-best list. In particular, it does not need to decompose additively across the translation in the way that the feature functions do. As such, it would be simple to instead use a different sentence-level approximation of BLEU, such as that proposed by Tromble et al. (2008), or translation edit rate (TER; Snover et al., 2006).

For Algorithms 1 and 2, we used $C = 1$, $\eta = 0.0001$, and $k = 500$. We did not extensively tune C, but tried a few values on the small-feature UR→EN task and found performance to generally be robust across C values; we used $C = 1$ for simplicity and to match the value used in PRO experiments. We set the fixed learning rate $\eta = 0.0001$ early on and did little tuning to it. We revisit this choice in Section 3.5.4, which discusses natural directions for future experimentation.

We chose $k = 500$ for all experiments that used Algorithms 1 and 2. In initial hyperparameter tuning, we did not aggregate k-best lists across iterations, and performance appeared sensitive to k. In this case, we found $k \in \{300, 500\}$ to work better than $k = 100$, particularly for hierarchical phrase-based models. We did not do any re-tuning of k for the primary setting used in this thesis, namely that in which we aggregate k-best lists across iterations. We conjecture that performance is less sensitive to k in this case, but we have not run such experiments.

For the number of optimization iterations, we used $T = 20$, $T' = 10$, and $T'' = 5$ for Algorithm 1 and used $T = 20$ and $T' = 50$ for Algorithm 2. We found performance to be sensitive to the values of T' and T''. If they are too small, there is not enough optimization performed per

iteration. If they are too large, we risk overfitting to the particular set of k-best lists. While it is possible to foil the algorithm by choosing extreme values, it was not difficult to find reasonable values for the small-feature UR→EN task and we found that these values worked well for the others.

For our algorithms, after training for $T = 20$ iterations, we took the parameters that gave the best results on the tuning set and used them to decode the test sets. We also did this for large-feature experiments using PRO, but we suspect it does not make much difference since the sequence of tuning set BLEU scores produced during PRO was generally very stable across the latter iterations.

For the initial parameters $\boldsymbol{\theta}^{(0)}$ in all experiments, we used the default Moses weights, i.e., 0.3 for reordering features, 0.2 for phrase table features, 0.5 for the language model, and -1 for the word penalty. For large-feature experiments, we initialized the weights of the additional features to 0.0.

3.5 Experimental Results

We present three sets of experimental results. In Section 3.5.1, we present experiments designed to choose α for each loss function in subsequent experiments. These experiments use a small-feature phrase-based model for UR→EN translation. Using these hyperparameters, in Section 3.5.2 we compare our ramp losses to the baselines on additional language pairs (AR→EN, ZH→EN, DE→EN, and EN→MG) for both phrase-based and hierarchical phrase-based models. We use the best-performing loss functions and compare them to the baselines in large-feature experiments in Section 3.5.3.

3.5.1 Tuning Hyperparameters

Table 3.1 shows the effects of varying α (the multiplier on the cost function) and the choice of whether or not to merge k-best lists across iterations. All experiments use UR→EN phrase-based translation. The purpose of these experiments is to both inform choices for the remaining experiments and observe stability across these values.

The weights used to translate the test sets are those from the iteration that gave the highest BLEU score on the tuning set. This selected iteration is shown in parentheses following the tuning BLEU score. In some cases the selected iteration was 0, indicating that the initial weights reached higher tuning BLEU than any weights found during learning.

We note that the minimum of the Bayes risk loss function (Eq. 2.16) is unaffected by α for $\alpha > 0$. However, since we use stochastic gradient descent for a fixed number of iterations and add ℓ_2 regularization to the loss, the choice of α will affect the parameters found. In fact, for all of the loss functions considered here, α interacts with the ℓ_2 regularization strength C and the stepsize η, so it should not be necessary to tune all three hyperparameters independently with an exhaustive grid search. These experiments are our best effort at hyperparameter tuning, but unfortunately a comprehensive comparison of loss functions, with each of their hyperparameters tuned appropri-

loss	α	aggregation of k-best lists				no aggregation			
		tune	(iter.)	test 1	test 2	tune	(iter.)	test 1	test 2
$\text{loss}_{\text{ramp 1}}$	**1**	22.8	(19)	22.3	22.5	22.8	(8)	22.4	22.5
	10	22.4	(4)	22.2	21.7	22.4	(5)	22.2	21.8
	20	22.3	(0)	22.3	22.4	22.3	(0)	22.3	22.4
$\text{loss}_{\text{ramp 2}}$	**1**	23.5	(20)	22.7	23.1	24.4	(20)	23.9	24.0
	10	23.2	(14)	22.5	23.0	22.3	(0)	22.3	22.4
	20	22.5	(7)	22.8	22.8	22.3	(0)	22.3	22.4
$\text{loss}_{\text{ramp 3}}$	1	23.1	(18)	22.4	22.5	24.0	(20)	23.1	23.4
	10	**24.6**	(17)	**24.6**	24.5	**24.6**	(10)	**24.3**	24.3
	20	24.0	(12)	24.3	**24.6**	24.3	(15)	**24.3**	**24.5**
$\text{loss}_{\text{softramp 1}}$	**1**	22.6	(2)	22.9	22.5	22.5	(1)	22.7	22.6
	10	22.3	(0)	22.3	22.4	22.4	(10)	22.2	21.9
	20	22.3	(0)	22.3	22.4	22.3	(0)	22.3	22.4
$\text{loss}_{\text{softramp 2}}$	1	22.7	(2)	22.9	22.5	22.6	(19)	23.1	22.7
	10	24.2	(13)	23.7	24.1	23.8	(19)	23.8	23.9
	20	24.2	(3)	23.9	24.2	24.2	(10)	24.1	23.9
$\text{loss}_{\text{softramp 3}}$	1	23.2	(20)	23.2	23.1	22.7	(19)	23.0	22.8
	10	23.8	(11)	24.0	23.7	23.7	(13)	23.9	23.8
	20	23.2	(4)	23.6	23.3	23.6	(14)	23.7	23.8
$\text{loss}_{\text{Bayes risk}}$	1	22.6	(2)	22.9	22.4	22.5	(18)	23.0	22.6
	10	23.8	(20)	23.5	23.6	23.6	(20)	23.6	23.5
	20	23.8	(18)	23.4	23.7	24.1	(16)	23.5	23.6

Table 3.1: %BLEU on tune and test sets for UR→EN phrase-based translation, showing the effects of varying α (cost function multiplier) and whether or not k-best lists are aggregated across iterations. The highest value in each BLEU score column is bold, as is the α selected for each particular loss function for all subsequent experiments; α is selected based on tuning BLEU.

ately, is beyond our computational means. So while we may not be truly discovering the best loss function for machine translation, we are at least finding a loss function and hyperparameters that succeed in practice for several language pairs and models.

The numbers show that performance does depend on α, and therefore it is necessary to tune α for each loss function. But we note that we can do so effectively based on the BLEU score of the *tuning* set alone. We focus on a comparison of loss functions in the next section; our primary use of these experiments is to choose α for each loss function for subsequent experiments. In Section 3.5.4, we report experiments with additional α values for each loss function in order to provide a fuller picture of the experimental space, but our experiments in the intervening sections

use the α values chosen using the experiments reported here.

We do not observe much difference between aggregating k-best lists across iterations and not. In experiments not reported here, we found a larger difference between these two strategies when using larger data sets, richer models, and not using the default Moses weights for initialization. In such cases, aggregating k-best lists across iterations was more stable, so we use this approach for all subsequent experiments in this thesis.

3.5.2 Small-Feature Experiments

We compared minimizing the ramp losses to Bayes risk, PRO, and MERT for two standard, small-feature settings: phrase-based and hierarchical phrase-based models with default features. Tables 3.2 and 3.3 show the results. MERT was run 3 times with differing random seeds and average BLEU scores are shown.

For phrase-based translation (Table 3.2), MERT achieves the highest BLEU score on average across all test sets, which is 0.10 higher than PRO, which is in turn 0.10 higher than $\text{loss}_{\text{softramp 2}}$. Two other ramp losses, $\text{loss}_{\text{softramp 3}}$ and $\text{loss}_{\text{ramp 3}}$, are close behind it, and Bayes risk follows. We find that softening the ramp losses tends to help performance, or at least not hurt. For hierarchical phrase-based translation (Table 3.3), $\text{loss}_{\text{softramp 2}}$ is now best, followed by MERT, $\text{loss}_{\text{ramp 3}}$, and then PRO, though the differences are not large.

In both tables, ramp loss is in second place or better for each language pair. For hierarchical phrase-based translation, $\text{loss}_{\text{softramp 2}}$ is best for 4 out of 5 language pairs, and second-best for the remaining. For both models, MERT achieves the highest average *tuning* BLEU score by a clear margin, showing evidence of systematic overfitting. In experiments not reported here, we also found $\text{loss}_{\text{ramp 3}}$ to be more stable across random initializers than MERT and PRO; furthermore, our learning algorithms are deterministic, while both MERT and PRO exhibit variance across random seeds (Gimpel and Smith, 2012a).

The purpose of these experiments is to validate our algorithms for standard models and feature sets. Our intent in designing learning algorithms is to support feature-rich modeling, so our primary interest is the large-feature setting, which we describe next.

phrase-based													
loss	α	AR→EN		ZH→EN		DE→EN		UR→EN		EN→MG		avg	
		tune	test	tune	test	tune	test	tune	test	tune	test	tune	test
$\text{loss}_{\text{ramp 1}}$	1	42.8	45.1	35.4	32.9	16.0	19.8	22.8	22.4	17.1	14.6	26.82	29.42
$\text{loss}_{\text{ramp 2}}$	1	42.9	45.3	35.2	33.2	16.1	19.8	23.5	22.9	17.1	14.6	26.96	29.59
$\text{loss}_{\text{ramp 3}}$	10	43.4	45.3	36.0	33.7	16.2	19.6	24.6	**24.5**	17.6	15.1	27.55	29.98
$\text{loss}_{\text{softramp 1}}$	1	43.1	**45.8**	34.9	32.8	16.4	20.3	22.6	22.7	17.1	14.6	26.82	29.72
$\text{loss}_{\text{softramp 2}}$	20	43.1	45.3	35.8	33.8	16.5	**20.4**	24.2	24.0	17.3	14.7	27.37	30.09
$\text{loss}_{\text{softramp 3}}$	10	43.2	45.6	35.4	33.7	16.5	20.0	23.8	23.8	17.2	14.5	27.22	29.99
$\text{loss}_{\text{Bayes risk}}$	20	43.0	45.6	35.4	33.8	16.5	19.9	23.8	23.6	17.1	14.6	27.17	29.94
MERT	-	**43.6**	**45.8**	35.9	33.7	**17.0**	20.3	**24.9**	**24.5**	**18.1**	**15.4**	**27.90**	**30.32**
PRO	-	43.1	45.6	35.8	**33.9**	16.6	**20.4**	24.0	24.1	17.3	14.9	27.37	30.23

Table 3.2: %BLEU on tune and test sets for all languages for phrase-based translation. For AR→EN, ZH→EN, and DE→EN, test BLEU scores are averages across three test sets. For UR→EN, test scores are averages over two test sets. For EN→MG, scores are given for a single test set. The test average in the final column is micro-averaged across test sets, not macro-averaged across language pairs. All results in this table used aggregation of k-best lists across iterations and a single initializer (the default Moses weights).

hierarchical phrase-based													
loss	α	AR→EN		ZH→EN		DE→EN		UR→EN		EN→MG		avg	
		tune	test	tune	test	tune	test	tune	test	tune	test	tune	test
$\text{loss}_{\text{ramp 1}}$	1	41.4	43.5	34.5	32.2	15.9	18.7	23.3	23.3	16.3	13.9	26.28	28.64
$\text{loss}_{\text{ramp 2}}$	1	42.8	44.9	35.8	33.9	15.8	18.6	24.9	25.9	17.2	15.0	27.32	29.89
$\text{loss}_{\text{ramp 3}}$	10	43.3	45.9	**37.3**	35.2	17.1	20.8	25.7	**26.6**	17.4	15.0	28.15	31.13
$\text{loss}_{\text{softramp 1}}$	1	42.5	45.4	35.3	33.4	16.8	20.4	24.4	24.5	16.5	14.0	27.09	30.06
$\text{loss}_{\text{softramp 2}}$	20	43.2	**46.1**	37.1	**35.4**	17.2	**21.0**	25.5	26.4	17.5	15.1	28.12	**31.29**
$\text{loss}_{\text{softramp 3}}$	10	43.3	45.6	36.9	35.0	16.8	20.7	24.7	25.1	17.3	15.0	27.79	30.76
$\text{loss}_{\text{Bayes risk}}$	20	43.3	45.6	36.7	34.8	16.9	20.7	24.6	25.0	17.2	15.0	27.75	30.69
MERT	-	**43.8**	45.9	37.1	35.1	**17.4**	**21.0**	**26.7**	26.4	**18.3**	**15.4**	**28.65**	31.19
PRO	-	43.1	**46.1**	37.0	35.1	17.2	20.9	25.2	25.9	17.6	15.0	28.01	31.08

Table 3.3: %BLEU on tune and test sets for all languages for hierarchical phrase-based translation. For AR→EN, ZH→EN, and DE→EN, test BLEU scores are averages across three test sets. For UR→EN, test scores are averages over two test sets. For EN→MG, scores are given for a single test set. The test average in the final column is micro-averaged across test sets, not macro-averaged across language pairs. All results in this table used aggregation of k-best lists across iterations and a single initializer (the default Moses weights).

3.5.3 Large-Feature Experiments

We compare the best-performing ramp losses to Bayes risk and PRO when using an extended feature set; MERT is excluded as it fails in such settings (Hopkins and May, 2011).

Features and Decoding Pipeline

For additional features, we added count features for common monolingual and bilingual lexical patterns from the parallel corpus for each language pair: the 1k most common bilingual word pairs from phrase extraction, 200 top unigrams, 1k top bigrams, 1k top trigrams, and 4k top trigger pairs extracted with the method of Rosenfeld (1996), ranked by mutual information. In all we added 7,200 features.

We integrated the features with our training procedure using the following pipeline. We first used Moses to generate phrase lattices instead of k-best lists. We pruned lattice edges using forward-backward pruning (Sixtus and Ortmanns, 1999), which has also been used by Tromble et al. (2008) for phrase lattice pruning. This pruning method computes the max-marginal for each lattice edge, which is the score of the best full path that uses that edge, then prunes edges whose max-marginal is below a certain fraction of the best path score in the lattice. We used a fraction of 0.01. This pruning strategy has the advantage that it preserves the best path in the lattice.

Given pruned lattices, we used cube pruning (Chiang, 2007) to incorporate the additional (potentially non-local) features while extracting k-best lists from the lattices. Each node's local M-best list used $M = 200$. Cube pruning is necessary because some of the bigram, trigram, and trigger pair features may be **non-local** in the phrase lattice, i.e., they might require looking at multiple lattice edges in order to be computed. We used lattices in order to avoid having to change the Moses decoder to add these features.

The k-best lists obtained by cube pruning are then passed to our training algorithms. After performing an iteration of training on the batch of k-best lists, we then call Moses with the new weights to generate new phrase lattices and the process repeats. We still merge k-best lists across iterations as before. We trained weights for the standard Moses features and the additional features simultaneously.

Results

Results are shown in Tables 3.4-3.7. In each table, we show the BLEU scores on tuning and test data with both the small and large feature sets. The "small" feature set includes only the default Moses phrase-based features; these numbers are the same as in Table 3.2. The "large" feature set adds the 7,200 features just described.

We find that PRO finds the highest BLEU scores on the tuning data but fails to generalize, leading to degradation on the held-out test sets. The ramp losses and Bayes risk are more stable by contrast, even improving in certain cases over the baseline small-feature model. Thus, if we only used PRO we might conclude that these features are harmful for these language pairs, but we might simply need to change the learning algorithm.

ZH→EN phrase-based with small and large feature sets

loss	features	tune	test 1	test 2	test 3	test avg.	Δ
$\text{loss}_{\text{ramp 3}}$	small	36.02	35.48	34.30	**31.27**	33.68	
	large	38.60	36.16	**34.65**	31.01	**33.94**	+0.26
$\text{loss}_{\text{softramp 2}}$	small	35.76	36.16	34.42	30.67	33.75	
	large	37.44	**36.59**	34.39	30.69	33.89	+0.14
$\text{loss}_{\text{Bayes risk}}$	small	35.44	36.02	34.26	30.87	33.72	
	large	36.82	36.24	34.58	31.01	**33.94**	+0.22
PRO	small	35.83	36.22	34.53	30.88	33.88	
	large	**39.90**	35.90	33.71	29.47	33.03	-0.85

Table 3.4: %BLEU on tune and test sets for ZH→EN phrase-based translation with small and large feature sets. The highest BLEU score in each column is bold. Final column shows difference in test average between small and large feature set for the given learning method.

However, it is still the case that none of the algorithms are able to find much improvement on test data using this feature set. Even when a small improvement is seen on average across test sets for a particular language pair, there is often one test set on which the small feature set performs better. For ZH→EN, the ramp losses find gains on average, but for the other language pairs the results are more mixed. Ramp loss 3 does not bring gains for DE→EN, UR→EN, or EN→MG, but it also does not hurt BLEU very much; as a result it is still better than the other algorithms for UR→EN and EN→MG translation. Soft ramp 2 yields improvements for UR→EN and EN→MG when adding features, but suffers a loss in BLEU for DE→EN. However, even when our algorithms suffer losses with the extra features (likely due to overfitting), the drops in BLEU are never as bad as those seen with PRO.

We suspect PRO's overfitting is worsened by training a binary classifier to convergence on each iteration.[4] As such it may require stronger regularization than our ramp loss optimization algorithms, which achieve a sort of regularization through early stopping (the setting of T' and T''). Our initial intent in limiting the number of optimization iterations in our algorithms was to avoid overfitting to a particular set of k-best lists, and this may also be preventing overfitting when using the larger feature set. While we note that Algorithms 1 and 2 do require choosing values for several hyperparameters, such as the number of optimization iterations, the learning rate, and the regularization strength, we used the same values of these hyperparameters across all language pairs and models, for both small- and large-feature settings. The only hyperparameter that differed across the loss functions was the cost function multiplier α, which we tuned for each

[4]We note that in the large-feature experiments we did not actually train PRO's binary classifier to convergence, but rather set a limit on the number of LBFGS iterations in training to 200. However, this does not appear sufficient to prevent overfitting.

DE→EN phrase-based with small and large feature sets

loss	features	tune	test 1	test 2	test 3	test avg.	Δ
$\text{loss}_{\text{ramp 3}}$	small	16.21	18.95	20.66	19.16	19.59	
	large	17.38	18.93	20.54	19.19	19.55	-0.04
$\text{loss}_{\text{softramp 2}}$	small	16.53	**19.87**	21.41	19.80	20.36	
	large	17.26	19.61	21.05	19.55	20.07	-0.29
$\text{loss}_{\text{Bayes risk}}$	small	16.45	19.10	21.00	19.51	19.87	
	large	16.63	19.20	21.11	19.68	20.00	+0.13
PRO	small	16.56	**19.80**	**21.42**	**19.89**	**20.37**	
	large	**18.88**	18.74	20.06	18.55	19.12	-1.25

Table 3.5: %BLEU on tune and test sets for DE→EN phrase-based translation with small and large feature sets.

loss using the UR→EN phrase-based task and then kept fixed for the remaining experiments.

This is a challenging learning task, as lexical features are prone to overfitting with small tuning sets. Hopkins and May (2011) and Cherry and Foster (2012) similarly found little gain on test data when using extended feature sets in phrase-based translation for ZH→EN and UR→EN translation.

UR→EN phrase-based with small and large feature sets

loss	features	tune	test 1	test 2	test avg.	Δ
$\text{loss}_{\text{ramp 3}}$	small	24.58	**24.61**	24.46	**24.54**	
	large	27.29	24.27	**24.75**	24.51	-0.03
$\text{loss}_{\text{softramp 2}}$	small	24.20	23.85	24.20	24.03	
	large	25.81	24.23	24.54	24.39	+0.36
$\text{loss}_{\text{Bayes risk}}$	small	23.84	23.44	23.70	23.57	
	large	24.60	24.02	23.96	23.99	+0.42
PRO	small	23.98	24.28	24.01	24.15	
	large	**28.28**	23.01	23.31	23.16	-0.99

Table 3.6: %BLEU on tune and test sets for UR→EN phrase-based translation with small and large feature sets.

EN→MG phrase-based with small and large feature sets

loss	features	tune	test	Δ
$\text{loss}_{\text{ramp 3}}$	small	17.56	**15.05**	
	large	20.21	15.02	-0.03
$\text{loss}_{\text{softramp 2}}$	small	17.30	14.72	
	large	18.42	14.82	+0.10
$\text{loss}_{\text{Bayes risk}}$	small	17.13	14.60	
	large	18.20	15.02	+0.42
PRO	small	17.33	14.88	
	large	**21.19**	13.98	-0.90

Table 3.7: %BLEU on tune and test sets for EN→MG phrase-based translation with small and large feature sets.

3.5.4 Additional Experimental Results

Although we presented results for a wide range of experimental conditions, spanning multiple models and feature sets, five language pairs, and twelve test sets, they still only cover a small fraction of the experimental space. In particular, we have not thoroughly explored the space of hyperparameter values for our learning algorithms. Therefore, our experiments do not definitively establish which loss function is optimal for learning in machine translation; rather, they identify particular points on the hyperparameter landscape that consistently lead to good performance.

Future work may discover better algorithms simply by doing a more thorough exploration of the experimental space. In this section, we present the results of some simple additional experiments conducted after obtaining the results in the preceding sections. We focus on the selection of the hyperparameters α (the cost function multiplier) and η (the fixed step size). The experiments in Section 3.5.1 showed the results of varying α for the different loss functions for the UR→EN phrase-based translation task and informed our selection of α in the subsequent experiments. However, we only considered $\alpha \in \{1, 10, 20\}$ for each loss, and for some losses the optimal value occurred at one of the extreme values. Most of the other hyperparameters, including the fixed step size η, were not thoroughly tuned in any of the preceding experiments. In order to conduct a fairer comparison and potentially discover better learning algorithms for machine translation, the effects of varying these hyperparameters should be investigated empirically in future work. In this section we report the results of some initial steps.

Additional α values

Table 3.8 shows results for UR→EN phrase-based translation when using additional values of α, namely 0.1, 0.5, 50, and 100. For $\text{loss}_{\text{ramp 1}}$, the rows for $\alpha = 50$ and 100 were identical to that for $\alpha = 20$ and are omitted. Similarly, for $\text{loss}_{\text{ramp 2}}$, the row for $\alpha = 100$ was identical to that for $\alpha = 50$ and, for $\text{loss}_{\text{softramp 1}}$, the rows for $\alpha > 10$ were equal to that for $\alpha = 10$. Assuming we again choose α based on tuning set BLEU score, the selection of α would only change for $\text{loss}_{\text{softramp 1}}$ and $\text{loss}_{\text{softramp 3}}$ when considering the enlarged set of α values. We note that, even though $\alpha = 20$ gives the best performance for $\text{loss}_{\text{softramp 2}}$ on the tuning set, setting $\alpha = 50$ gives better performance on the test sets. This is significant because $\text{loss}_{\text{softramp 2}}$ with $\alpha = 20$ performed well across conditions in our preceding experiments, so it is possible that using $\alpha = 50$ would lead to even better results.

Increasing η for large feature sets

We chose $\eta = 0.0001$ based on preliminary experiments with the small-feature UR→EN phrase-based translation task, but η may need to be tuned separately when using large numbers of sparse features. We report one additional experimental result along these lines: we increase η from 0.0001 to 0.001 for the large-feature ZH→EN phrase-based translation task. Table 3.9 shows the result. We find that tuning BLEU increases, but that test set BLEU decreases, suggesting overfitting. Fortunately, however, the drop in test BLEU is not as large as that suffered by PRO.

3.6 Summary

In this chapter, we presented new loss functions and simple algorithms for optimizing them for machine translation. Several of our loss functions were inspired by well-known MT learning algorithms, and we noted these connections, situating all in an empirical risk minimization framework. We performed a thorough experimental evaluation using 12 test sets across 5 language pairs for both phrase-based and hierarchical phrase-based models. Our experiments show that our learning algorithms are competitive with the state-of-the-art for the standard small-feature setting.

We also conducted experiments using a phrase-based model with an enlarged feature set consisting of 7,200 lexical monolingual and bilingual features. In this setting, MERT fails (Hopkins and May, 2011) so we used PRO as our baseline, which is becoming widely-used by MT researchers for large-feature experiments due to its simplicity. However, we found that PRO overfit with these features, finding very high BLEU scores on the tuning data but suffering losses in performance on held-out data of up to 1 BLEU point. By contrast, our algorithms maintained or exceeded their baseline performance when adding the additional features.

Our algorithms are: (1) based on loss functions with attractive properties, (2) conceptually straightforward, as they use a simple cost function and standard optimization subroutines, and (3) easy to implement and parallelize, as they use k-best lists, batch optimization, and simple parameter updates that require no external software libraries. Our hope is that the adoption of these algorithms can enable more research in the use of large feature sets for machine translation.

In Chapter 4 we introduce our feature-rich syntax-based translation model and we use the algorithms presented here to learn its parameters in the experiments in Chapter 5.

loss	α	tune	(iter.)	test 1	test 2
$\text{loss}_{\text{ramp 1}}$	0.1	22.5	(20)	22.5	22.5
	0.5	22.7	(20)	22.4	22.4
	1	22.8	(19)	22.3	22.5
	10	22.4	(4)	22.2	21.7
	20	22.3	(0)	22.3	22.4
$\text{loss}_{\text{ramp 2}}$	0.1	22.6	(19)	22.4	22.5
	0.5	23.4	(13)	22.3	22.7
	1	23.5	(20)	22.7	23.1
	10	23.2	(14)	22.5	23.0
	20	22.5	(7)	22.8	22.8
	50	22.3	(0)	22.3	22.4
$\text{loss}_{\text{ramp 3}}$	0.1	22.9	(20)	22.4	22.4
	0.5	23.7	(18)	22.8	23.0
	1	23.1	(18)	22.4	22.5
	10	**24.6**	(17)	**24.6**	24.5
	20	24.0	(12)	24.3	**24.6**
	50	24.0	(9)	24.4	24.3
	100	24.2	(9)	24.4	24.2
$\text{loss}_{\text{softramp 1}}$	**0.1**	22.7	(4)	22.6	22.6
	0.5	22.7	(1)	22.6	22.5
	1	22.6	(2)	22.9	22.5
	10	22.3	(0)	22.3	22.4
$\text{loss}_{\text{softramp 2}}$	0.1	22.7	(5)	22.6	22.5
	0.5	22.7	(1)	22.6	22.4
	1	22.7	(2)	22.9	22.5
	10	24.2	(13)	23.7	24.1
	20	24.2	(3)	23.9	24.2
	50	24.0	(2)	24.3	24.5
	100	23.6	(2)	24.0	23.9
$\text{loss}_{\text{softramp 3}}$	0.1	22.7	(4)	22.6	22.6
	0.5	22.6	(2)	22.9	22.4
	1	23.2	(20)	23.2	23.1
	10	23.8	(11)	24.0	23.7
	20	23.2	(4)	23.6	23.3
	50	23.7	(4)	24.1	23.6
	100	23.9	(4)	23.8	23.7
$\text{loss}_{\text{Bayes risk}}$	0.1	22.7	(5)	22.6	22.5
	0.5	22.6	(1)	22.6	22.4
	1	22.6	(2)	22.9	22.4
	10	23.82	(20)	23.5	23.6
	20	23.84	(18)	23.4	23.7
	50	23.83	(10)	23.4	23.6
	100	23.78	(4)	23.4	23.5

Table 3.8: Results when using $\alpha \in \{0.1, 0.5, 1, 10, 20, 50, 100\}$ for each loss for UR→EN phrase-based translation. Omitted rows for a particular loss were identical to the largest α value shown for that loss.

ZH→EN phrase-based with small and large feature sets

		features	tune	test 1	test 2	test 3	test avg.	Δ
$\text{loss}_{\text{ramp 3}}$	$\eta = 0.0001$	small	36.02	35.48	34.30	**31.27**	33.68	
	$\eta = 0.0001$	large	38.60	36.16	**34.65**	31.01	**33.94**	+0.26
	$\eta = 0.001$	large	39.57	35.99	34.48	30.12	33.53	-0.15
PRO		small	35.83	**36.22**	34.53	30.88	33.88	
		large	**39.90**	35.90	33.71	29.47	33.03	-0.85

Table 3.9: Showing the effect of increasing η for ZH→EN phrase-based translation with the large feature set. The highest BLEU score in each column is bold. Increasing η by an order of magnitude results in higher tuning BLEU but lower test BLEU, although the drop in test BLEU when adding features is still smaller than that seen with PRO.

Chapter 4

Quasi-Synchronous Phrase Dependency Grammars

In this chapter, we introduce our translation model. It is based on a novel formalism for translation—**phrase dependency grammar**—and combines structural units of phrase-based and syntax-based translation in a single model. Our model resides in the framework of **quasi-synchronous grammar** (QG; Smith and Eisner, 2006a), a flexible framework for tree-to-tree translation that does not require source and target trees to be isomorphic. In this chapter, we define phrase dependency grammar and our model, including its rules and features, and give a decoding algorithm. The next chapter presents experiments with the model and analysis of the results.

This chapter is laid out as follows. In Section 4.1 we present a simpler model—Model W—that serves as a starting point for our exposition. In presenting it, we review quasi-synchronous grammar and include an empirical study of syntactic divergence in a German-English parallel corpus that serves to motivate our approach. We then present our core model—Model P—in Section 4.2, along with a description of its formalism, rules (Section 4.3), and features (Section 4.4). We discuss decoding in Section 4.5 and close with a summary in Section 4.6. This chapter contains material originally published in Gimpel and Smith (2009b, 2011a), as well as additional features, examples, and background material.

4.1 Quasi-Synchronous Grammar

To build up to our model, we present a simpler model first. It is based on quasi-synchronous dependency grammar and is the model originally presented in Gimpel and Smith (2009b). It is not competitive with the current state of the art, so we do not include it in our experiments, but it provides a stepping stone to the model we present in Section 4.2. We call it Model W (for "word"). While describing Model W, we also provide background on quasi-synchronous grammar and conduct an empirical study of syntactic divergence in a real parallel corpus to motivate our choices in developing Model P.

In Section 2.2.2, we discussed synchronous grammars. Synchronous grammars generate

strings in two languages simultaneously. As such they appear to be a natural fit for modeling translation, but practical instantiations are constrained by isomorphism in the derivations of the two strings (Ding and Palmer, 2005). Smith and Eisner (2006a) noted that such limitations result from an emphasis on *generating* the two strings. However, for many real-world applications, such as translation, one of the sentences is provided. The model only needs to score translations of the given sentence, not provide a generative story for entire sentence pairs. Smith and Eisner proposed an alternative to synchronous grammars—which they call **quasi-synchronous grammar** (QG)—that exploits this fact for increased flexibility in translation modeling. A QG assumes the source sentence and its parse are given, and scores possible translations of the source sentence along with their parses. That is, a quasi-synchronous grammar is a *monolingual* grammar over (derivations of) strings in the target language. The strings are scored based on an alignment from nodes in the target tree to nodes in the source tree.

Quasi-synchronous grammar makes no restrictions on the form of the target monolingual grammar, though dependency grammars have been used in most previous applications of QG (Wang et al., 2007; Das and Smith, 2009; Smith and Eisner, 2009), including word alignment for MT (Smith and Eisner, 2006a). QG has been used for a variety of other applications involving relationships among sentences, such as question answering (Wang et al., 2007), paraphrase identification (Das and Smith, 2009), parser projection and adaptation (Smith and Eisner, 2009), title generation (Woodsend et al., 2010), sentence simplification (Woodsend and Lapata, 2011), information retrieval (Park et al., 2011), and supervised parsing from multiple treebanks with different annotation conventions (Li et al., 2012). This thesis represents its first application to machine translation. In the next section we present an initial translation model based on QG.

4.1.1 Model W

Model W is a **quasi-synchronous dependency grammar** (QDG). A QDG is a QG in which the target-side monolingual grammar is a dependency grammar (defined in Section 2.2.3).

Given a source sentence $\boldsymbol{x}$ and its dependency parse $\tau_{\boldsymbol{x}}$, the derivation variable $\boldsymbol{h}$ in Model W contains a dependency tree $\tau_{\boldsymbol{y}}$ for the translation $\boldsymbol{y}$ and an alignment $\boldsymbol{c}$ from nodes in $\tau_{\boldsymbol{y}}$ to nodes in $\tau_{\boldsymbol{x}}$ or, equivalently, words in $\boldsymbol{y}$ to words in $\boldsymbol{x}$.[1] In principle, any portion of $\tau_{\boldsymbol{y}}$ may align to any portion of $\tau_{\boldsymbol{x}}$, but in practice we make restrictions on the alignments to simplify computation. Smith and Eisner, for example, required each target word to align to at most one source word. We also do so here: we define $\boldsymbol{c}$ to be a vector of length $m = |\boldsymbol{y}|$ where, if $c_j = i$, y_j is aligned to x_i. If $c_j = 0$, y_j is aligned to NULL.[2] Given these definitions, $\boldsymbol{h} \stackrel{\text{def}}{=} \langle \tau_{\boldsymbol{y}}, \boldsymbol{c} \rangle$. Given a sentence $\boldsymbol{x}$ and its dependency tree $\tau_{\boldsymbol{x}}$, decoding searches over sentences $\boldsymbol{y}$, dependency trees $\tau_{\boldsymbol{y}}$, and alignments $\boldsymbol{c}$. We refer to this as Model W in the remainder of this thesis.

[1] We note that while this equivalence exists when using dependency grammars, it does not hold for phrase structure grammars.

[2] Note that we introduce new notation here because $\boldsymbol{c}$ holds alignments from target words to source words, allowing multiple target words to align to the same source word; we defined $\boldsymbol{a}$ in Chapter 2 to hold alignments from source words to target words, allowing multiple *source* words to align to the same target word.

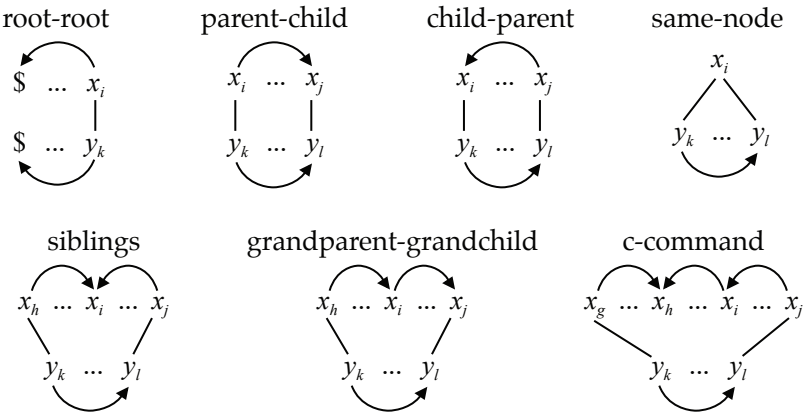


Figure 4.1: Quasi-synchronous configurations from Smith and Eisner (2006a). There are additional configurations involving NULL alignments and an "other" category for those that do not fit into any of the named categories.

Which translations are possible depends on the **configurations** that the QDG permits. Formally, for a child-parent pair $\langle y_j, y_{\tau_{\boldsymbol{y}}(j)} \rangle$ in $\tau_{\boldsymbol{y}}$, we consider the relationship between source words $\boldsymbol{c}(j)$ and $\boldsymbol{c}(\tau_{\boldsymbol{y}}(j))$, the words to which y_j and $y_{\tau_{\boldsymbol{y}}(j)}$ align. If, for example, we require that, for all $j \in [m]$, $\boldsymbol{c}(\tau_{\boldsymbol{y}}(j)) = \tau_{\boldsymbol{x}}(\boldsymbol{c}(j))$ or $\boldsymbol{c}(j) = 0$, and that the root of $\tau_{\boldsymbol{y}}$ must align to the root of $\tau_{\boldsymbol{x}}$ or to NULL, then strict isomorphism holds between the source and target trees, and we have implemented a synchronous context-free dependency grammar (Alshawi et al., 2000).

Smith and Eisner (2006a) grouped the configurations into eight classes and explored the effects of permitting different sets of classes in word alignment. This first configuration, "$\boldsymbol{c}(\tau_{\boldsymbol{y}}(j)) = \tau_{\boldsymbol{x}}(\boldsymbol{c}(j))$," corresponds to their "parent-child" configuration. Figure 4.1 shows the rest; we give examples in the next section.

4.1.2 Configuration Analysis in Data

The configurations are fundamental to QG. They allow the model to discourage or constrain various types of non-isomorphic structure. But when defining and discussing these configurations, several questions arise. How frequent are these configurations in real data? What types of phenomena do they capture? Are these true instances of syntactic divergence or are they caused by errors of automatic annotation tools or different annotation conventions?

Some of these questions were addressed in studies of syntactic divergence in parallel corpora (Wu, 1997; Fox, 2002; Zens and Ney, 2003; Wellington et al., 2006; Zhang et al., 2006; Søgaard and Kuhn, 2009). However, most of this prior work used manual word alignments. We use automatic word alignment and automatic parsing, so our analysis will be affected by the noise in

root configuration	count	percentage
root — root	85,878	58.2%
root — non-root	58,961	39.9%
root — NULL	2,756	1.9%
total	147,595	100%

Table 4.1: Counts of root configurations.

real-world tools. This lets us understand the types and amounts of syntactic divergence that we need to handle when building real-world systems, including errors made by word alignment systems and parsers and problems stemming from differences in treebank conventions.

In this section, we contribute an analysis of the QDG configuration categories in parallel German-English data. We aligned the DE→EN data described in Appendix C using GIZA++ (Och and Ney, 2003), then focused on the news commentary portion which consists of 150,697 sentence pairs. We parsed the German side using the factored model in the Stanford parser (Rafferty and Manning, 2008), which can output dependencies using head rules based on the phrase structure parses. We parsed the English side using TurboParser (Martins et al., 2009), a state-of-the-art dependency parser. We discarded sentence pairs which contained multiple roots; TurboParser occasionally returns structures with multiple roots, and it is not surprising that this would happen in parallel text as it frequently contains many-to-one sentence alignments. There were 3,102 sentence pairs in which the English sentence had multiple roots, leaving 147,595 sentence pairs for the analysis. We ran GIZA++ in the target-to-source direction, giving us alignments with at most one source word aligned to each target word, to match the definition of $\boldsymbol{c}$ above.

Given this parsed, word-aligned parallel corpus, we counted occurrences of the various tree-to-tree configurations. Table 4.1 shows the counts of configurations involving alignment of root words. The root word in the English tree is aligned to the root word in the German tree in 58.2% of the sentence pairs. The English root is aligned to a word other than the German root a surprisingly large percentage of the time. When we looked at these cases, we found that they typically occurred in sentences with multiple clauses, each with its own verb, and the German and English parsers chose different verbs as the roots of their dependency trees. Sometimes they are different verbs in the same verb phrase (auxiliary vs. infinitive) and other times they are in different verb phrases.

Table 4.2 shows configurations involving non-root dependencies. While parent-child configurations are the most frequent, they only occur a third of the time. These numbers suggest a surprising amount of non-isomorphism between German and English, at least when using off-the-shelf word aligners and parsers.

Inspection reveals that much of the divergence is due to differences in annotation conventions and head-finding rules. For example, 13.8% of the configurations are sibling relationships, meaning that the source words aligned to the parent and child in the English sentence have the same

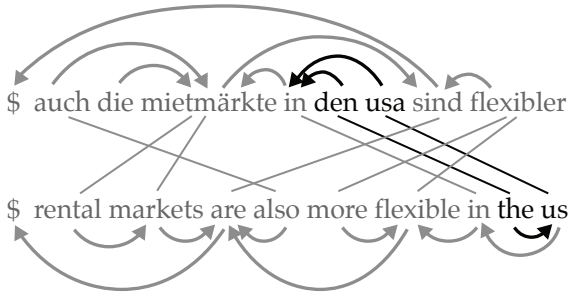


Figure 4.2: Example of a sentence pair containing a frequently-observed sibling configuration in DE→EN data: in the *the→us* dependency, the aligned German words are siblings in the source dependency tree. This occurs due to differences in treebank and head rule conventions between the two languages. The German parser produces flat PPs with little internal structure, so when the dependency tree is generated, each word in the PP attaches to the P, the head of the phrase. Many sibling configurations in the data follow this pattern.

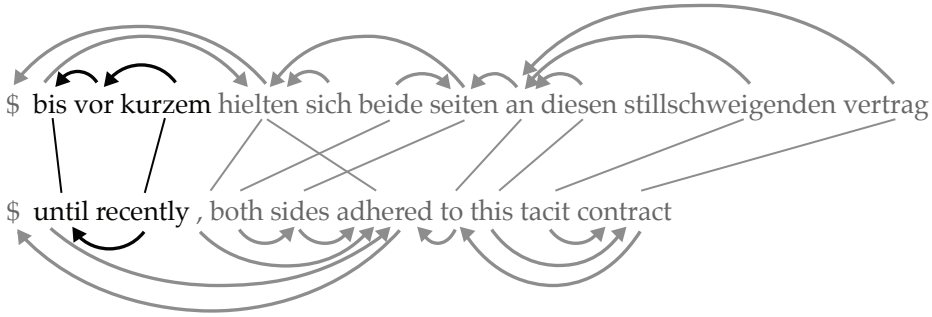


Figure 4.3: Example of a sentence pair containing a frequently-observed grandparent-grandchild configuration in DE→EN data.

configuration	count	percentage
parent-child	1,080,736	33.7%
child-parent	193,407	6.0%
same node	349,502	10.9%
siblings	444,160	13.8%
grandparent-grandchild	212,363	6.6%
c-command	472,120	14.7%
child — NULL, parent — !NULL	91,639	2.9%
child — !NULL, parent — NULL	59,649	1.9%
child — NULL, parent — NULL	1,031	<0.1%
other	302,952	9.4%
total	3,207,559	100%

Table 4.2: Counts of configurations involving non-root dependencies.

parent on the German side. Figure 4.2 shows a common example. Many sibling configurations appear when the English dependency is DET→N within a PP. The Stanford factored German parser tends to generate flat PPs of the form "P DET N" with the P marked as the head. When the parser converts this to a dependency tree, however, the DET and N are made children of the P. In English dependency parsing, the DET would typically be a child of the N, which would be a child of the P. Hence, some of the most common sibling configurations occur on dependencies that frequently lie within PPs, like *the→us* and *recent→years*.

Differences in annotation and parser conventions are one reason for using a formalism like QG, but we also found evidence of genuine syntactic divergence. Figure 4.3 shows an example of a grandparent-grandchild configuration. In the English dependency *until←recently*, the aligned source words are in a grandparent relationship in the source-sentence dependency tree. The German phrase *vor kurzem* translates into the English *recently*, but this is not reflected in the word alignments due to the limit of one source word aligned to each target word. One approach would be to permit many-to-many word alignments and define more complex configurations. The configurations proposed by Smith and Eisner are simple due to the many-to-one constraints of $\boldsymbol{c}$. Generating configurations on many-to-many alignments quickly becomes unwieldy. We note, however, that if we simply group *vor kurzem* as a phrase, and let the entire phrase be the child of *bis*, then this becomes a parent-child configuration.

Similarly, in Figure 4.2, if we grouped together *the us* as a phrase and also *den usa*, then both would be children of the preposition and we would have a parent-child configuration instead of a sibling configuration. So by considering phrasal structure and dependencies among phrases, we hope to reduce some of the syntactic divergence in real-world data. The model we develop in the next section is based on this idea.

4.2 Quasi-Synchronous Phrase Dependency Grammars

In the previous section we showed how a parsed, word-aligned parallel corpus can exhibit a surprising amount of non-isomorphic structure. We also noted two examples in which flattening the tree structure into phrasal dependencies could eliminate some cases of non-isomorphism. This idea is compatible with a principle well-known to MT researchers that translation quality is improved when units larger than words are modeled. For example, phrase-based models (Section 2.2.1) excel at capturing local reordering phenomena and memorizing multi-word translations.

But models that employ syntax or syntax-like representations (Sections 2.2.2 and 2.2.3) handle long-distance reordering better than phrase-based systems (Birch et al., 2009), although they often require constraints on the formalism or rule extraction method in order to achieve computational tractability. As a result, certain instances of syntactic divergence are more naturally handled by phrase-based systems (DeNeefe et al., 2007). We want a way to combine structural components of these two approaches to MT in a single model, in order to better handle syntactic divergence and obtain the benefits of phrase-based and syntax-based models.

In this section we formalize these ideas in a new model for syntax-based machine translation. The core of our model is the use of a dependency grammar in which the leaves are *phrases* rather than words, which we term a **phrase dependency grammar**. The model we define is inspired by Model W from the previous section but adds phrasal structure. We call the result Model P (for "phrase"). This model was originally presented in Gimpel and Smith (2011a).

In Section 4.2.1, we describe the formalism and in Section 4.2.2 we define our model's derivation variable. In Section 4.3 we describe how rules are extracted and how the output space is populated. We include several examples of frequently-extracted rules from our DE→EN corpus. We then discuss the features in our model in Section 4.4. These include several categories, including those that only look at target-side syntax, those that also look at the source sentence, and finally those that additionally look at source-side syntax. Decoding is discussed in Section 4.5.

4.2.1 Phrase Dependency Grammars

In Section 2.2.3 we defined dependency trees and dependency grammar. Now we provide similar definitions for phrase dependency trees and phrase dependency grammar. Given a sentence $\boldsymbol{y}$ and its segmentation into n' phrases $\boldsymbol{\phi}$ (Section 2.2.1), we define a **phrase dependency tree** as a function $\tau_{\boldsymbol{\phi}} : \{1, \ldots, n'\} \rightarrow \{0, \ldots, n'\}$ where $\tau_{\boldsymbol{\phi}}(i)$ is the index of the parent of phrase ϕ_i. If $\tau_{\boldsymbol{\phi}}(i) = 0$, we say phrase ϕ_i is the **root** of the tree. As with lexical dependency trees, $\tau_{\boldsymbol{\phi}}$ cannot have cycles. We restrict our attention to projective phrase dependency trees in this thesis, but the generalization to non-projective trees is easily made.

We define a **phrase dependency grammar** as a model over the joint space of sentences, their segmentations into phrases, and projective phrase dependency trees on the phrases. We assume a

linear model parameterized by $\boldsymbol{\theta}$ and with feature vector $\boldsymbol{f}$:

$$\langle \boldsymbol{y}^*, \boldsymbol{\phi}^*, \tau_{\boldsymbol{\phi}}^* \rangle = \operatorname*{argmax}_{\boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}} \sum_{i=1}^{n'} \boldsymbol{\theta}^\top \boldsymbol{f}(\phi_i, \phi_{\tau_{\boldsymbol{\phi}}(i)}) \tag{4.1}$$

We assume that $\boldsymbol{f}$ is arc-factored, as in Eq. 2.10.

Phrase dependency grammars have also been used by Wu et al. (2009) for feature extraction for opinion mining. When used for translation modeling, they allow us to capture phenomena like local reordering and idiomatic translations within each phrase as well as longer-distance relationships among the phrases in a sentence.

4.2.2 Model P

To define the latent derivation variable for Model P, we build on the phrase-based derivation variable described in Section 2.2.1. To review, in phrase-based models, the latent variable $\boldsymbol{h}$ contains a segmentation of $\boldsymbol{x}$ into n' phrases, which we denote $\boldsymbol{\pi}$, a segmentation of $\boldsymbol{y}$ into the same number of phrases, denoted $\boldsymbol{\phi}$, and a one-to-one alignment $\boldsymbol{b} : \{1, \ldots, n'\} \rightarrow \{1, \ldots, n'\}$ from phrases in $\boldsymbol{\phi}$ to phrases in $\boldsymbol{\pi}$. Model P additionally contains a phrase dependency tree on the phrases in the target sentence, which we denote $\tau_{\boldsymbol{\phi}}$. The alignment variable still aligns target phrases to source phrases.

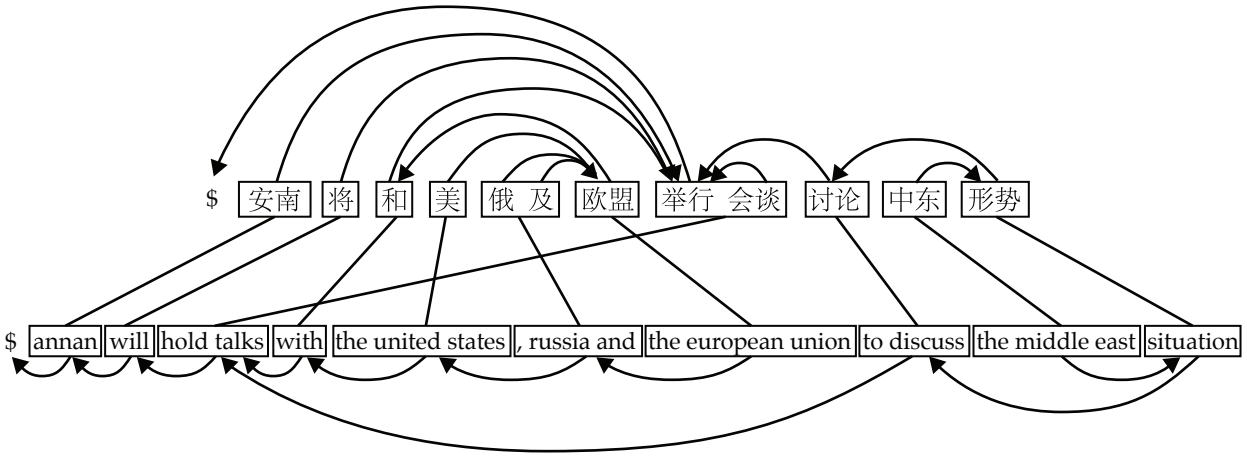
If a source-language parser is available, we also use a dependency tree on the source sentence, denoted $\tau_{\boldsymbol{x}}$. This is not part of the derivation variable, but is more accurately considered as part of the input $\boldsymbol{x}$, since it is fully observed at translation time. We include features that consider $\tau_{\boldsymbol{x}}$ if it is available, but the presence or absence of a source-side parse does not alter the basic decoding algorithm.

Given these definitions, $\boldsymbol{h} \stackrel{\text{def}}{=} \langle \boldsymbol{\pi}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b} \rangle$. That is, given a sentence $\boldsymbol{x}$ and its dependency tree $\tau_{\boldsymbol{x}}$, we formulate the translation problem as finding the target sentence $\boldsymbol{y}^*$, the segmentation $\boldsymbol{\pi}^*$ of $\boldsymbol{x}$ into phrases, the segmentation $\boldsymbol{\phi}^*$ of $\boldsymbol{y}^*$ into phrases, the dependency tree $\tau_{\boldsymbol{\phi}}^*$ on the target phrases $\boldsymbol{\phi}^*$, and the one-to-one phrase alignment $\boldsymbol{b}^*$ such that

$$\langle \boldsymbol{y}^*, \boldsymbol{\pi}^*, \boldsymbol{\phi}^*, \tau_{\boldsymbol{\phi}}^*, \boldsymbol{b}^* \rangle = \operatorname*{argmax}_{\langle \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b} \rangle \in \mathcal{T}_{\boldsymbol{x}}} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b}) \tag{4.2}$$

This model is formally a quasi-synchronous grammar that uses a phrase dependency grammar as the target-side monolingual grammar. So we call the model a **quasi-synchronous phrase dependency grammar**. We refer to it as Model P in this thesis.

Figure 4.4 shows an example. The inputs to the model are a sentence and its *lexical* dependency tree. We used the Stanford parser (Levy and Manning, 2003) to get dependency trees for Chinese. The outputs of the model include the English translation, its segmentation into phrases, a projective dependency tree on the English *phrases*, a segmentation of the Chinese sentence into phrases, and a one-to-one alignment between the English phrases and Chinese phrases. Four reference translations are also shown. In this example, the model correctly moved the phrase *hold talks* and also noted its connection to *to discuss* by making the latter a phrasal dependent.



references: annan to hold talks with us , russia and eu over situation in middle east
annan will discuss middle east situation with u.s. , russia and european union
annan to discuss mideast situation with us , russia and eu
annan to meeting the us , russia and eu to discuss middle east crisis

Figure 4.4: Example output of Model P for Chinese-to-English translation. Chinese sentence and lexical dependency tree are inputs and the Model P derivation includes the English translation, its phrase segmentation, a projective dependency tree on the English phrases, a phrase segmentation of the Chinese sentence, and a one-to-one alignment between the English phrases and Chinese phrases.

For phrase-based models, $\mathcal{T}_{\boldsymbol{x}}$ is determined by the phrase table as described in Section 2.2.1. In our model, even though we have an additional variable in our derivation, we do not wish to enforce any additional constraints on $\mathcal{Y}_{\boldsymbol{x}}$. That is, the translation space $\mathcal{Y}_{\boldsymbol{x}}$ is still determined by the phrase table alone; we allow $\tau_{\boldsymbol{\phi}}$ to be *any* projective phrase dependency tree on $\boldsymbol{\phi}$, so the structure of $\tau_{\boldsymbol{\phi}}$ merely affects how translations in $\mathcal{Y}_{\boldsymbol{x}}$ are scored, not what translations are permitted. We made this decision because we did not want to reduce the coverage of phrase-based models, which is one of their strengths. Rather, we wanted to better score their translations.

This also gives rise to certain limitations of our approach. Since we do not expand the search space beyond what is licensed by the phrase table, or indeed what is actually explored during decoding (see Section 4.5), we are limited by the ability of the underlying phrase-based model to provide us with a good search space.

Finally, we note that Model P actually departs from the original definition of QG (Smith and Eisner, 2006a). The alignment variable in QG links target tree nodes to source tree nodes. However, we never commit to a source phrase dependency tree, instead using a source *lexical* dependency tree from a dependency parser, so our alignment variable $\boldsymbol{b}$ is a function from target tree

nodes (phrases in ϕ) to source phrases in $\boldsymbol{\pi}$, which might not be source tree nodes. The features in our model consider many source phrase dependency trees that are compatible with $\tau_{\boldsymbol{x}}$.

4.3 Rule Extraction

Though all of the rules used to populate $\mathcal{Y}_{\boldsymbol{x}}$ come from the phrase table, we need additional rules to score the target tree τ_ϕ for elements of $\mathcal{T}_{\boldsymbol{x}}$. The rules we discuss in this section are used to compute relative frequency estimates for the features presented in Section 4.4. We first discuss rules that only look at target-side words and syntax; later in the section we extract rules that also look at the source sentence. We never, however, extract rules that require source-side syntax. All of the features that use source syntax are unlexicalized. This has the advantage of avoiding the computational expense of parsing the source side of the parallel corpus.

4.3.1 Target-Tree Rules

We first extract rules that only consider $\boldsymbol{y}$, ϕ, and τ_ϕ. These rules can be used as the basis for "dependency language model" features (Shen et al., 2008; Galley and Manning, 2009), though unlike this previous work, our features model both the phrase segmentation and dependency structure. Typically, these sorts of features are relative frequencies from a corpus parsed using a supervised parser. However, there do not currently exist treebanks with annotated phrase dependency trees. Our solution is to use a standard lexical dependency parser and extract phrase dependencies using bilingual information.[3] Essentially, we combine phrases from the standard phrase extraction pipeline with certain lexical dependencies from the output of a dependency parser.

We begin by obtaining word alignments and extracting phrase pairs that are **p-consistent** with them (as described in Section 2.2.1). We parse the target sentence with a projective dependency parser to obtain a projective dependency tree $\tau_{\boldsymbol{y}}$ for a sentence $\boldsymbol{y}$. Note that $\tau_{\boldsymbol{y}}$ is a tree on *words*, not phrases. For each pair of target-side phrases in the phrase pairs from phrase extraction, we extract a phrase dependency (along with its direction) if the phrases do not overlap and there is at least one lexical dependency between them. If there is only a dependency in one direction, we extract a single phrase dependency with that direction. If there are lexical dependencies in both directions, we extract a phrase dependency only for the single longest lexical dependency, and in its direction. Since we use a projective dependency parser, the longest lexical dependency between two phrases is guaranteed to be unique. If a phrase contains the root word in $\tau_{\boldsymbol{y}}$, we extract a phrase dependency with the wall symbol as its head.

We now present the procedure more formally. Since it is similar to phrase extraction in phrase-based translation, we restate some definitions from Section 2.2.1. First, word alignments (possibly many-to-many) are obtained for each sentence pair in the corpus using a word alignment system. Let R denote the alignment relation between the two sets $[n]$ and $[m]$, where $n = |\boldsymbol{x}|$ and $m = |\boldsymbol{y}|$. If a pair $(i, j) \in R$, for some $i \in [n]$ and $j \in [m]$, then we say that x_i is aligned to y_j. We say a

[3] Other ways of getting phrase dependencies are possible. E.g., for a monolingual task, Wu et al. (2009) used a shallow parser to convert lexical dependencies from a dependency parser into phrase dependencies.

phrase pair $\langle \boldsymbol{x}_i^j, \boldsymbol{y}_k^l \rangle$ is **p-consistent** with R if, $\forall u$ such that $i \leq u \leq j$, and all v such that $(u, v) \in R$, it is the case that $k \leq v \leq l$.

Our goal is to extract target-side **phrase dependencies**. We say a phrase dependency $\langle \boldsymbol{y}_i^j, \boldsymbol{y}_k^l \rangle$ with $\boldsymbol{y}_k^l$ as the parent phrase is **d-consistent** with $\tau_{\boldsymbol{y}}$ and R if:

1. $\exists \boldsymbol{x}_{i'}^{j'}, \boldsymbol{x}_{k'}^{l'}$ such that $\langle \boldsymbol{x}_{i'}^{j'}, \boldsymbol{y}_i^j \rangle$ and $\langle \boldsymbol{x}_{k'}^{l'}, \boldsymbol{y}_k^l \rangle$ are p-consistent with R
2. $\boldsymbol{y}_i^j$ and $\boldsymbol{y}_k^l$ do not overlap: $(1 \leq i \leq j < k \leq l \leq m) \vee (1 \leq k \leq l < i \leq j \leq m)$
3. the longest lexical dependency from $\boldsymbol{y}_i^j$ to $\boldsymbol{y}_k^l$ is longer than the longest from $\boldsymbol{y}_k^l$ to $\boldsymbol{y}_i^j$: $\max_{i \leq u \leq j} \mathbb{I}\left[k \leq \tau_{\boldsymbol{y}}(u) \leq l\right] |\tau_{\boldsymbol{y}}(u) - u| > \max_{k \leq v \leq l} \mathbb{I}\left[i \leq \tau_{\boldsymbol{y}}(v) \leq j\right] |\tau_{\boldsymbol{y}}(v) - v|$

where $\mathbb{I}[P]$ is the indicator function that returns 1 if P evaluates to true and 0 otherwise. The indicator functions in the final condition restrict attention to dependencies ending in the other phrase. The final condition also implies that there is a lexical dependency from a word in $\boldsymbol{y}_i^j$ to a word in $\boldsymbol{y}_k^l$: $\exists u, i \leq u \leq j$, such that $k \leq \tau_{\boldsymbol{y}}(u) \leq l$.

We also need to extract root phrase dependencies. We say a root phrase dependency $\langle \boldsymbol{y}_i^j, \$ \rangle$ is d-consistent with $\tau_{\boldsymbol{y}}$ and R if:

1. $\exists \boldsymbol{x}_{i'}^{j'}$ such that $\langle \pi, \boldsymbol{y}_i^j \rangle$ is p-consistent with R
2. $\exists u, i \leq u \leq j$, such that $\tau_{\boldsymbol{y}}(u) = 0$

We extract all phrase dependencies that are d-consistent with the word alignments and target-side lexical dependency trees. We note that while extracting phrase dependencies we never explicitly commit to any single phrase dependency tree for a target sentence. Rather, we extract phrase dependencies from all phrase dependency trees compatible with the word alignments and the lexical dependency tree. Thus we treat phrase dependency trees analogously to phrase segmentations in standard phrase extraction.

Our rule extraction algorithm does not look at any source parse tree. The only features that use source syntax are unlexicalized, considering only the derivation structure and source dependency tree and not at particular words. This has the advantage of avoiding the need to parse the source side of the parallel corpus.

When actually extracting phrase dependencies, we take additional information from the sentence pairs in which they are found. Specifically, for d-consistent phrase dependencies $\langle \boldsymbol{y}_i^j, \boldsymbol{y}_k^l \rangle$ (where $\boldsymbol{y}_k^l$ is the head), we extract tuples of the following form:

$$\langle \boldsymbol{y}_i^j, \boldsymbol{y}_k^l, y_{u^*}, y_{\tau_{\boldsymbol{y}}(u^*)}, \mathbb{I}\left[j < k\right] \rangle \tag{4.3}$$

where u^* is chosen as

$$u^* = \operatorname*{argmax}_{i \leq u \leq j} \; \mathbb{I}\left[k \leq \tau_{\boldsymbol{y}}(u) \leq l\right] |\tau_{\boldsymbol{y}}(u) - u|$$

making $\langle y_{u^*}, y_{\tau_{\boldsymbol{y}}(u^*)} \rangle$ the longest lexical dependency within the phrase dependency. This lexical dependency is recorded for use in back-off features, analogous to the lexical weighting in phrase-based models. The fifth field holds the direction of the phrase dependency, which is also the

35265	it **is**	6210	i **think**	4843	**would** be	2918	**thank** you
13751	this **is**	6115	**is** that	4289	we **will**	2816	it **will**
12763	**is** a	6105	**is** not	4262	i **believe** that	2788	**is** to
11831	we **have**	6019	, it **is**	4018	**is** also	2737	it **is** a
11551	**would** like	5975	**believe** that	3910	that **is** why	2736	it **has**
11245	we **must**	5818	**will** be	3838	i **would** like to	2730	they **are**
11243	**is** the	5706	we **need**	3775	**would** like to	2611	we **can**
11015	i **would** like	5628	there **are**	3505	**hope** that	2580	i **think** that
10008	there **is**	5495	**should** like	3427	**is** an	2551	i **will**
8983	i **am**	5453	i **should** like	3239	, i **would** like	2483	**does** not
8019	we **are**	5227	i **hope**	3130	i **hope** that	2482	debate **is**
7722	that **is**	5150	, **is**	3101	**need** to	2445	i **can**
6883	i **would**	5110	we **should**	3059	it **was**	2438	**want** to
6443	i **have**	5010	**has** been	3021	**have** been	2416	**must** be
6328	i **believe**	4917	**do** not	2937	**think** that	2405	this **is** a

Table 4.3: Top 60 most frequent root phrases in DE→EN data with at least 2 words, shown with their counts. Shown in bold are the actual sentence roots in the lexical dependency trees from which these phrases were extracted; these are extracted along with the phrases and used for back-off features.

direction of the longest lexical dependency. Root phrase dependencies use $k = l = 0$ in the parent phrase and designate \$ as y_0. The direction of root phrase dependencies is immaterial and can remain as $\mathbb{I}[j < k]$.

4.3.2 Examples

What do typical phrase dependencies look like? Tables 4.3 and 4.4 show some of the most frequent examples of root phrases and parent-child phrase dependencies extracted by this technique on our DE→EN corpus. The English side of the parallel corpus was parsed using TurboParser (Martins et al., 2009). Naturally, there are many phrase dependencies with a single word in each phrase, but since these are very similar to lists of frequent lexical dependencies in a parsed corpus, we have only shown dependencies with phrases containing more than one word.

Root phrases (Table 4.3) frequently contain a subject along with a verb (*it is*, *i would like*, etc.), though the lexical root is typically a verb or auxiliary. These are examples of how we can get syntactic information for phrases that typically would not correspond to constituents in phrase structure trees.

Table 4.4 shows frequent phrase dependencies from the same corpus. Since this corpus is mostly European Parliamentary proceedings, certain formulaic and domain-specific phrases appear with large counts. When phrases attach to each other, they typically behave like their heads.

30064	**mr** → **president** ,	3347	**,** but → **is**
19931	**the** → european **union**	3332	**in** the ← european **union**
18819	**the** european → **union**	3326	**it** → **is** a
12318	**i** → **would** like	3301	the **commission** → **has**
11990	**the** → member **states**	3242	**in** ← european **union**
8169	it **is** ← **that**	3213	**this** → **is** a
7779	**the** → european **parliament**	3172	**not** ← , **but**
7762	**madam** → **president** ,	3171	to **ensure** ← **that**
7448	**the** european → **parliament**	3168	**that** ← **should** be
6897	**of** the ← **union**	3158	**that** it ← **is**
6196	**mr** → **president** , i	3154	, **but** ← **also**
6188	**i** → **should** like	3145	**in** the european ← **union**
6087	**that** the ← **is**	3112	**in** ← the european **union**
5478	**i** → **believe** that	3102	**a** → **number** of
5422	**)** → **president** ,	3062	**i** → **hope** that
5283	**of** the ← european **union**	3019	**the** → european **commission**
5268	**that** → and **that**	3017	**president** , → **commissioner**
4956	**of** the european ← **union**	2988	**president** , → **commissioner** ,
4902	**,** and → **is**	2979	**the** united → **states**
4798	**the** → united **states**	2964	**the** european → **commission**
4607	**)** mr → **president** ,	2951	**president** → **commissioner** ,
4592	**,** it → **is**	2931	**i** → **think** that
4582	i **believe** ← **that**	2921	**that** the ← **has**
4516	, **which** ← **is**	2885	**president** → , **commissioner**
4347	**that** ← **will** be	2860	mr **president** → ladies and **gentlemen** ,
4297	the **fact** ← **that**	2857	**by** the ← **commission**
4289	it **is** ← **important**	2838	**,** i → **would** like
4232	**one** ← **of** the	2832	**we** → **do** not
4215	**of** the ← **commission**	2826	mr **president** , → ladies and **gentlemen** ,
3932	it **is** ← **not**	2775	**president** → , **commissioner** ,
3793	**i** → **would** like to	2764	**not** only ← **but**
3761	**in** the ← **union**	2751	**mr** → **president** , the
3752	**in** ← member **states**	2697	**of** the ← **parliament**
3713	**)** → mr **president** ,	2693	**that** this ← **is**
3673	**president** → ladies and **gentlemen** ,	2645	**that** we ← **have**
3673	**is** ← **that** the	2638	**and** → it **is**
3667	**president** , → ladies and **gentlemen** ,	2534	**not** ← **but** also
3602	i **hope** ← **that**	2515	**of** the ← member **states**
3531	**we** → **need** to	2509	**it** → **is** not
3495	**the** → **fact** that	2506	**of** the ← european **parliament**
3494	that **the** → **commission**	2489	**,** we → **have**
3462	**i** → **do** not	2487	**the** → **commission** and
3446	, **the** → **commission**	2480	to **thank** ← **for**
3421	**that** the ← **will**	2471	**mr** → **president** , commissioner
3353	**,** the → **is**	2420	**the** → **use** of

Table 4.4: Most frequent phrase dependencies in DE→EN data, shown with their counts and attachment directions. Child phrases point to their parents. To focus on interesting phrase dependencies, we only show those in which one phrase has at least 2 tokens and neither phrase is entirely punctuation. The words forming the longest lexical dependency in each extracted phrase dependency are shown in bold; these are used for back-off features.

176937	is	14122	{we, they} {should, must, cannot, shall}
49725	{are, were}	12757	is the
43651	{will, can, could, may}	12628	{we, they} {are, were}
42203	{should, must, cannot, shall}	12401	would {like, according, relating}
41324	have	12150	i would {like, according, relating}
40403	it is	11480	there is
38671	has	10316	i {am, voted, 'm}
28467	{should, must, cannot, shall}	10203	{need, want, needs, wish, wanted}
23672	{was, took, makes, comes, seems}	10193	{say, believe, think, know, hope}
21847	would	9659	{report, committee, issue, agreement, debate} is
21470	{will, can, could, may}	9447	{am, voted, 'm}
17204	is {important, clear, necessary, concerned, proposed}	9367	{say, believe, think, know, hope}
		9213	it is {important, clear, necessary, concerned, proposed}
16016	this is		
14944	{we, they} have	8858	that is
14365	is a	8723	is {also, still, really, always, never}

Table 4.5: Top 30 most frequent Brown cluster root phrases from DE→EN data, shown with their counts. Each cluster is shown as a set of words large enough to cover 95% of the token counts in the cluster, up to a maximum of 5 words. It is characteristic of Brown clustering that very frequent words often receive their own clusters, such as *is*, *have*, *it*, etc.

For example, in the phrase dependency *of the*←*union*, the word *union* is the child phrase because *of the* is behaving like *of*. There is likely a dependency from *the* to *union* whenever this phrase dependency is extracted, but due to our practice of following the longest lexical dependency in deciding the direction, *of*←*union* is favored over *the*→*union*.

We note that even though these phrase dependencies only contain words from the target language (English), the presence and counts of the phrase dependencies will depend on the source language through the word alignments. For example, when *of the union* is expressed in German, the preposition will often be dropped and the definite article chosen to express genitive case. In our corpus, the most common translation of the English *union* is in fact the German noun *union*, which is feminine. The genitive feminine definite article is *der* and, indeed, we find in the phrase table that the translation of *of the union* with highest probability is *der union*.[4] Thus the dominance of the phrase dependency *of the*←*union* (6897 occurrences) as compared with *of*←*the union* (142 occurrences) is caused by the German translation.

We also extract tuples in which we replace each word by its hard cluster ID obtained through Brown clustering (Brown et al., 1992). In the following, $\mathrm{clust}()$ is a function that takes a sequence

[4]The phrase table probability of the German *der union* given the English *of the union* is 0.64. The next most-probable German phrase is *der europäischen union*, with probability 0.03.

of words and replaces each by its Brown cluster:

$$\langle \text{clust}(\boldsymbol{y}_i^j), \text{clust}(\boldsymbol{y}_k^l), \text{clust}(y_{u^*}), \text{clust}(y_{\tau_{\boldsymbol{y}}(u^*)}), \mathbb{I}\left[j < k\right] \rangle \tag{4.4}$$

Examples of frequent Brown cluster phrase dependencies are shown in Table 4.5. The purpose of this table is also to show examples of Brown clusters, so we have not removed dependencies with only one word in each phrase as we did in the earlier tables.

4.3.3 String-to-Tree Rules

Our simplest features use the information in the above tuples, but we also extract tuples with more information to support richer features. In particular, we record aligned source phrases and details about reordering and the presence of gaps between phrases. That is, for d-consistent phrase dependencies $\langle \boldsymbol{y}_i^j, \boldsymbol{y}_k^l \rangle$, we extract tuples

$$\begin{aligned}
&\langle \boldsymbol{y}_i^j, \boldsymbol{y}_k^l, \boldsymbol{x}_{i'}^{j'}, \boldsymbol{x}_{k'}^{l'}, \\
&\mathbb{I}\left[j < k\right], \\
&\mathbb{I}\left[\mathbb{I}\left[j < k\right)\right] = \mathbb{I}\left[j' < k'\right]\Big], \\
&\mathbb{I}\left[(j+1 = k) \vee (l+1 = i)\right], \\
&\mathbb{I}\left[(j'+1 = k') \vee (l'+1 = i')\right]\rangle
\end{aligned}$$

for all i', j', k', and l' such that the phrase pairs $\langle \boldsymbol{x}_{i'}^{j'}, \boldsymbol{y}_i^j \rangle$ and $\langle \boldsymbol{x}_{k'}^{l'}, \boldsymbol{y}_k^l \rangle$ are p-consistent with R, and such that $\boldsymbol{x}_{i'}^{j'}$ does not overlap with $\boldsymbol{x}_{k'}^{l'}$.[5] We include here the two target phrases, their aligned source phrases, the direction of the target attachment, the **orientation** between the source and target phrases (whether the two target phrases are in the same order as their aligned source phrases or swapped), whether a gap is present between the two target phrases, and finally whether a gap is present between the two source phrases. When $\boldsymbol{y}_k^l = \$$, all of the additional fields are irrelevant except the aligned source phrase $\boldsymbol{x}_{i'}^{j'}$.

We now note some examples of the phenomena that we can model with these richer tuples. A common cause of reordering in German-to-English translation relates to verbs. Figure 4.5 shows two examples of frequently-extracted phrase dependencies that model verb movement. Figure 4.5(a) gives an example of how German reorders the finite verb to the end of a dependent clause, while English keeps it next to the subject. The extracted rule, shown below the sentence pair, only applies when intervening words appear on the German side and no intervening words appear on the English side. This is indicated by the presence of an ellipsis on the German side of the rule.

Figure 4.5(b) shows an example of how German moves an infinitive (*danken*, "to thank") to the end of an independent clause when a modal verb (*möchte*, "would like") is present. The ellipses on both sides indicate that other words must be present between both the source and target phrases.

[5] This non-overlapping constraint is what differentiates these tuples from the target-tree rule tuples from the previous section, which are extracted even when the source phrases overlap.

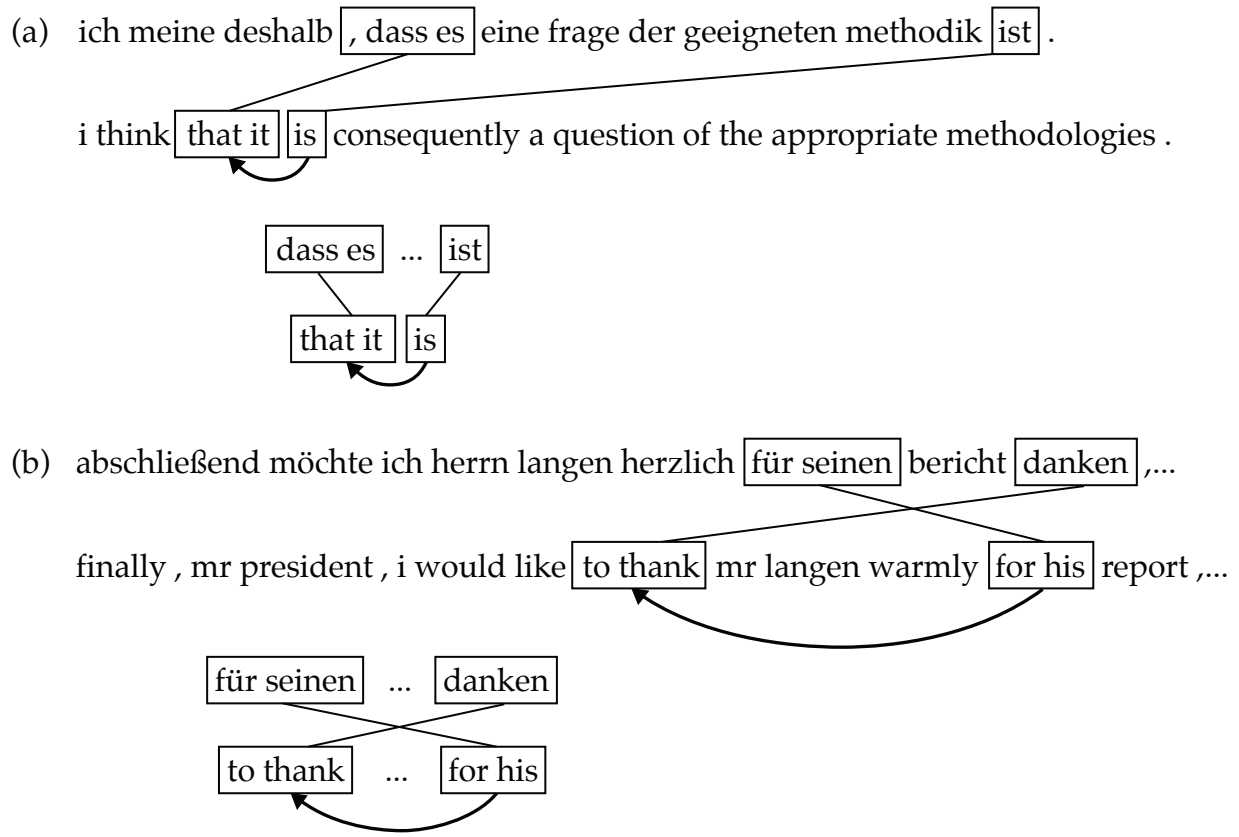


Figure 4.5: Examples of frequently-extracted rules that model verb movement between German and English. An ellipsis indicates that there must be material between the two phrases for the rule to apply. (a) Example of movement of the finite verb to the end of a dependent clause. (b) Example of movement of an infinitive to the end of an independent clause following a modal verb (*möchte*, "would like").

We note that this rule says nothing about what fills the gap. In particular, the gap-filling material does *not* have to be translationally equivalent, and indeed in the given sentence pair it is not. As opposed to hierarchical rules with gaps, which typically specify translationally-equivalent substructures, this rule simply models the reordering and long-distance movement of the infinitive.

Many researchers have shown phrase pairs with gaps to be useful for machine translation (Simard et al., 2005; Chiang, 2005; Crego and Yvon, 2009; Galley and Manning, 2010), and we extract these tuples so that we can model these types of structure. However, the gappy phrase pairs we model are confined to pairs of p-consistent phrase pairs. We do not extend the rule set beyond that obtained during standard phrase extraction, but rather simply include additional features in our model that consider the two phrase pairs connected by each (target-side) phrase dependency.

The tuples we described account for all of the lexicalized phrase dependency features in Model P. We extract each tuple with a count of 1 each time it is observed, aggregate the counts across all sentence pairs in the parallel corpus, and use the counts to compute the features we present in the next section. We also have structural features that consider string-to-tree and tree-to-tree configurations, but these do not require any rule extraction. In the next section we present all of the features in our model.

4.4 Features

Our model contains all the features in the Moses phrase-based model described in Section 2.2.1, including four phrase table probability features, a phrase penalty feature, an N-gram language model, a distortion cost, six lexicalized reordering features, and a word penalty feature.

We now describe in detail the additional features in our model that are used to score phrase dependency trees. We shall refer to these as QPDG features and will find it useful later to notationally distinguish their feature weights from those of the phrase-based model. We use $\boldsymbol{\lambda}$ for weights of the standard phrase-based model features and $\boldsymbol{\psi}$ for weights of the QPDG features. The full set of model parameters is denoted $\boldsymbol{\theta} = \langle \boldsymbol{\lambda}, \boldsymbol{\psi} \rangle$. We include several categories of features, differentiated by what pieces of structure they consider.

4.4.1 Target-Tree Features

We include a feature in the model for the sum of the scaled log-probabilities of each attachment in $\tau_{\boldsymbol{\phi}}$:

$$f_{\text{pdep}}(\boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}) = \sum_{i=1}^{n'} \max\left(0, C + \log \tilde{p}(\phi_i \mid \phi_{\tau_{\boldsymbol{\phi}}(i)}, \text{dir}(\tau_{\boldsymbol{\phi}}, i))\right)$$

where $\text{dir}(\tau_{\boldsymbol{\phi}}, i)$ is defined

$$\text{dir}(\tau_{\boldsymbol{\phi}}, i) = \begin{cases} \textsf{root} & \text{if } \tau_{\boldsymbol{\phi}}(i) = 0 \\ \textsf{left} & \text{if } \tau_{\boldsymbol{\phi}}(i) > i \\ \textsf{right} & \text{otherwise} \end{cases}$$

and returns the direction of the phrase dependency attachment of phrase ϕ_i, i.e., the direction in which the child resides; root indicates that ϕ_i is the root phrase in the tree. We use $\tilde{p}$ to indicate that the probability is estimated using relative frequency estimation from the tuples extracted in the previous section, as in the phrase table features from Section 2.2.1.

Although we use log-probabilities in this feature function, we first add a constant C which is chosen to ensure the feature is never negative. The reasoning here is that whenever we use a phrase dependency that we have observed in the training data, we want to boost the score of the translation. If we used log-probabilities, each observed dependency would incur a penalty. The $\max$ expression protects unseen parent-child phrase dependencies from causing the score to be negative infinity. Our motivation is a desire for the features to be used to prefer one derivation over another but not to rule out a derivation completely if it merely happens to contain a dependency unobserved in the training data.

Since we will use this same practice for all of the other probability features that we introduce next, we now introduce some shorthand for simplicity of presentation. We first redefine the feature above:

$$f_{\text{pdep}}(\boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}) = \sum_{i=1}^{m'} \max\left(0, C_{\text{pdep}} + \log g_{\text{pdep}}(\boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i)\right) \tag{4.5}$$

where

$$g_{\text{pdep}}(\boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i \mid \phi_{\tau_{\boldsymbol{\phi}}(i)}, \text{dir}(\tau_{\boldsymbol{\phi}}, i))$$

In what follows, we will restrict our attention to defining the g functions for probability features, and assume that there is always a corresponding f that has the same subscript and takes the same inputs (except i), as in Eq. 4.5, and that C is chosen appropriately for each g based on the minimum log-probability for the feature. For example,

$$C_{\text{pdep}} = 0.01 - \min_{\phi, \phi', d} \log \tilde{p}(\phi \mid \phi', d) \tag{4.6}$$

i.e., the minimum log-probability is found, negated, and a small positive value (0.01) is added to ensure the feature is greater than zero. This ensures that, if a phrase dependency has been seen, its contribution to f_{pdep} is at least 0.01.

To counteract data sparseness, we include other features that are less specific than g_{pdep}. First, we include a version of this feature with words replaced by Brown clusters:

$$g_{\substack{\text{pdep} \\ \text{clust}}}(\boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\text{clust}(\phi_i) \mid \text{clust}(\phi_{\tau_{\boldsymbol{\phi}}(i)}), \text{dir}(\tau_{\boldsymbol{\phi}}, i))$$

We also include lexical weighting features similar to those used in phrase-based MT (Koehn et al., 2003). These use the longest lexical dependencies extracted during rule extraction (described in Section 4.3). First, for all ⟨child, parent, direction⟩ *lexical* dependency tuples $\langle y, y', d\rangle$ in the parsed target side of the parallel corpus, we estimate conditional probabilities $\tilde{p}_{\text{lex}}(y \mid y', d)$ using relative frequency estimation.

Then, assuming phrase dependency i has longest child-parent lexical dependency $\langle y, y' \rangle$ for direction $\text{dir}(\tau_{\boldsymbol{\phi}}, i)$, we include the feature:

$$g_{\text{ldep}}(\boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}_{\text{lex}}(y \mid y', \text{dir}(\tau_{\boldsymbol{\phi}}, i)) \tag{4.7}$$

We include an analogous feature with words replaced by Brown clusters. Different instances of a phrase dependency may have different lexical dependencies extracted with them. We only use the lexical weight for the most frequent, breaking ties by choosing the lexical dependency that maximizes $\tilde{p}_{\text{lex}}(y \mid y', d)$, as was done similarly by Koehn et al. (2003).

So far we described four features that consider $\boldsymbol{y}$, $\boldsymbol{\phi}$, and $\tau_{\boldsymbol{\phi}}$: one for phrase dependencies, one for lexical dependencies, and the same features computed on a transformed version of the corpus in which each word is replaced by its Brown cluster.

4.4.2 String-to-Tree Features

We next discuss features that consider properties of $\boldsymbol{x}$, $\boldsymbol{\pi}$, and $\boldsymbol{b}$ in addition to $\boldsymbol{y}$, $\boldsymbol{\phi}$, and $\tau_{\boldsymbol{\phi}}$. That is, these features look at the source sentence, its phrase segmentation, and the phrase alignment. However, they still do not look at the source tree $\tau_{\boldsymbol{x}}$, so they can be included even when a parser for the source language is not available. We will discuss features that use $\tau_{\boldsymbol{x}}$ later in this chapter.

We include features similar to g_{pdep} above, but which condition on additional pieces of structure. All features condition on direction. The first pair of features condition on the source phrase ($\pi_{\boldsymbol{b}(i)}$) aligned to the child phrase (ϕ_i) in the target phrase dependency ($\langle \phi_i, \phi_{\tau_{\boldsymbol{\phi}}(i)} \rangle$):

$$g_{\substack{\text{pdep} \\ \text{child}}}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i \mid \phi_{\tau_{\boldsymbol{\phi}}(i)}, \text{dir}(\tau_{\boldsymbol{\phi}}, i), \pi_{\boldsymbol{b}(i)})$$

$$g_{\substack{\text{pdep} \\ \text{child} \\ \text{clust}}}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i \mid \text{clust}(\phi_{\tau_{\boldsymbol{\phi}}(i)}), \text{dir}(\tau_{\boldsymbol{\phi}}, i), \pi_{\boldsymbol{b}(i)})$$

In the second feature, we condition on word clusters for the parent phrase, but on words in the aligned source phrase $\pi_{\boldsymbol{b}(i)}$. In general, we ensure that if we use clusters on the left-hand side of the conditioning bar, we use clusters for the aligned material that is being conditioned on, and if we use word on the left-hand side, we use words for the aligned phrases that are conditioned on. Since Brown clusters often correspond to syntactic clusters, even at times resembling part-of-speech tags (Christodoulopoulos et al., 2010), it did not seem logical to model translation probabilities between source- and target-language word clusters. This is why we did not include a feature like the above with word clusters for ϕ_i and $\pi_{\boldsymbol{b}(i)}$.

The next set of features includes those that condition on the orientation between the source- and target-side phrases. The $\text{ori}()$ function returns the orientation of the aligned source phrases in a target phrase dependency attachment, namely whether the aligned source phrases are in the same order as the target phrases ("same") or if they are in the opposite order ("swap"):

$$\text{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}) = \begin{cases} \textsf{root} & \text{if } \tau_{\boldsymbol{\phi}}(i) = 0 \\ \textsf{same} & \text{if } (\text{dir}(\tau_{\boldsymbol{\phi}}, i) = \textsf{left} \wedge \boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i)) > \boldsymbol{b}(i)) \vee (\text{dir}(\tau_{\boldsymbol{\phi}}, i) = \textsf{right} \wedge \boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i)) < \boldsymbol{b}(i)) \\ \textsf{swap} & \text{otherwise} \end{cases}$$

Given this definition of ori, we define the following features that condition on orientation (in addition to other fields):

$$g_{\substack{\text{pdep}\\\text{orient}}}(\boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i \mid \phi_{\tau_{\boldsymbol{\phi}}(i)}, \mathrm{dir}(\tau_{\boldsymbol{\phi}}, i), \mathrm{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}))$$

$$g_{\substack{\text{pdep}\\\text{orient}\\\text{clust}}}(\boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\mathrm{clust}(\phi_i) \mid \mathrm{clust}(\phi_{\tau_{\boldsymbol{\phi}}(i)}), \mathrm{dir}(\tau_{\boldsymbol{\phi}}, i), \mathrm{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}))$$

$$g_{\substack{\text{pdep}\\\text{child}\\\text{orient}}}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i \mid \phi_{\tau_{\boldsymbol{\phi}}(i)}, \mathrm{dir}(\tau_{\boldsymbol{\phi}}, i), \pi_{\boldsymbol{b}(i)}, \mathrm{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}))$$

$$g_{\substack{\text{pdep}\\\text{child}\\\text{orient}\\\text{clust}}}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i \mid \mathrm{clust}(\phi_{\tau_{\boldsymbol{\phi}}(i)}), \mathrm{dir}(\tau_{\boldsymbol{\phi}}, i), \pi_{\boldsymbol{b}(i)}, \mathrm{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}))$$

where the last two features condition on the aligned child phrase in addition to the direction and orientation.

We next give features that condition on the presence of gaps between the child and parent target phrases and gaps between the aligned phrases on the source side. The $\mathrm{tgap}()$ function indicates whether there is a gap between the target phrases in the phrase dependency in which y_i is the child phrase:

$$\mathrm{tgap}(\tau_{\boldsymbol{\phi}}, i) = \begin{cases} \mathsf{root} & \text{if } \tau_{\boldsymbol{\phi}}(i) = 0 \\ \mathsf{yes} & \text{if } |\tau_{\boldsymbol{\phi}}(i) - i| \geq 1 \\ \mathsf{no} & \text{otherwise} \end{cases}$$

The $\mathrm{sgap}()$ function indicates whether there is a gap between the aligned source phrases:

$$\mathrm{sgap}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}) = \begin{cases} \mathsf{root} & \text{if } \tau_{\boldsymbol{\phi}}(i) = 0 \\ \mathsf{yes} & \text{if } |b(\tau_{\boldsymbol{\phi}}(i)) - b(i)| \geq 1 \\ \mathsf{no} & \text{otherwise} \end{cases}$$

Given these gap functions, we define the following features:

$$g_{\substack{\text{pdep}\\\text{orient}\\\text{gap}}}(\boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i \mid \phi_{\tau_{\boldsymbol{\phi}}(i)}, \mathrm{dir}(\tau_{\boldsymbol{\phi}}, i), \mathrm{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}), \mathrm{sgap}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}), \mathrm{tgap}(\tau_{\boldsymbol{\phi}}, i))$$

$$g_{\substack{\text{pdep}\\\text{orient}\\\text{gap}\\\text{clust}}}(\boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\mathrm{clust}(\phi_i) \mid \mathrm{clust}(\phi_{\tau_{\boldsymbol{\phi}}(i)}), \mathrm{dir}(\tau_{\boldsymbol{\phi}}, i), \mathrm{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}), \mathrm{sgap}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}), \mathrm{tgap}(\tau_{\boldsymbol{\phi}}, i))$$

All the features mentioned so far have the child phrase on the left-hand side. We now present features that have both the child and parent phrases on the left-hand side. These features can model phrase pairs with gaps in them that are composed of phrase pairs extracted by a phrase-

based model:

$$g_{\substack{\text{pdep}\\\text{pc}}}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i, \phi_{\tau_{\boldsymbol{\phi}}(i)}, \text{dir}(\tau_{\boldsymbol{\phi}}, i) \mid \pi_{\boldsymbol{b}(i)}, \pi_{\boldsymbol{b}(\tau_{\boldsymbol{\phi}} i)})$$

$$g_{\substack{\text{pdep}\\\text{pc}\\\text{orient}}}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i, \phi_{\tau_{\boldsymbol{\phi}}(i)}, \text{dir}(\tau_{\boldsymbol{\phi}}, i) \mid \pi_{\boldsymbol{b}(i)}, \pi_{\boldsymbol{b}(\tau_{\boldsymbol{\phi}} i)}, \text{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}))$$

$$g_{\substack{\text{pdep}\\\text{pc}\\\text{orient}\\\text{gap}}}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, i) = \tilde{p}(\phi_i, \phi_{\tau_{\boldsymbol{\phi}}(i)}, \text{dir}(\tau_{\boldsymbol{\phi}}, i), \text{tgap}(\tau_{\boldsymbol{\phi}}, i) \mid \pi_{\boldsymbol{b}(i)}, \pi_{\boldsymbol{b}(\tau_{\boldsymbol{\phi}} i)}, \text{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}), \text{sgap}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}))$$

These last features score larger rules composed of two phrase pairs from the phrase table. Including direction, orientation, and gaps enables us to model longer-distance reorderings; we showed some examples of such frequently-extracted phrase dependencies in the previous section.

In all, we introduced 11 features in this section, giving us a total of 15 features so far.

4.4.3 Dependency Length Features

We include features that compute lengths of phrase dependencies (i.e., number of words crossed). We can also enforce maximum dependency lengths through hard constraints to speed up inference. We will return to this in Section 4.5. These features and constraints are similar to those used in **vine grammar** (Eisner and Smith, 2005).

We first include a feature that counts the number of source-side words between the aligned source phrases in each attachment in $\tau_{\boldsymbol{\phi}}$:

$$f^{\text{src}}_{\text{vine}}(\boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}) = \sum_{i=1}^{n'} \sum_{j=\min(b(i), b(\tau_{\boldsymbol{\phi}}(i)))+1}^{\max(b(i), b(\tau_{\boldsymbol{\phi}}(i)))-1} |\pi_j|$$

where the verbose bounds in the inner sum handle both possible directions and orientations of the target phrases. While this feature requires the segmentation of the source sentence in order to determine the number of source words crossed, the actual identities of those words are not needed, so the feature does not depend on $\boldsymbol{x}$. We would expect this feature weight to be negative for most language pairs, encouraging closeness in the source sentence of phrases aligned to each phrase dependency in the target.

We would like to use a similar feature for dependency lengths in phrase dependencies on the target side, e.g.:

$$\sum_{i=1}^{n'} \sum_{j=\min(i, \tau_{\boldsymbol{\phi}}(i))+1}^{\max(i, \tau_{\boldsymbol{\phi}}(i))-1} |\phi_j|$$

However, such a feature could require looking at the entire translation being generated to score a single phrase dependency (e.g., if $\tau_{\boldsymbol{\phi}}(1) = m'$). Using this feature would prevent us from being able to use dynamic programming for decoding (we discuss our approach to decoding in Section 4.5). Instead, we use a feature that considers *bounds* on the number of target words crossed

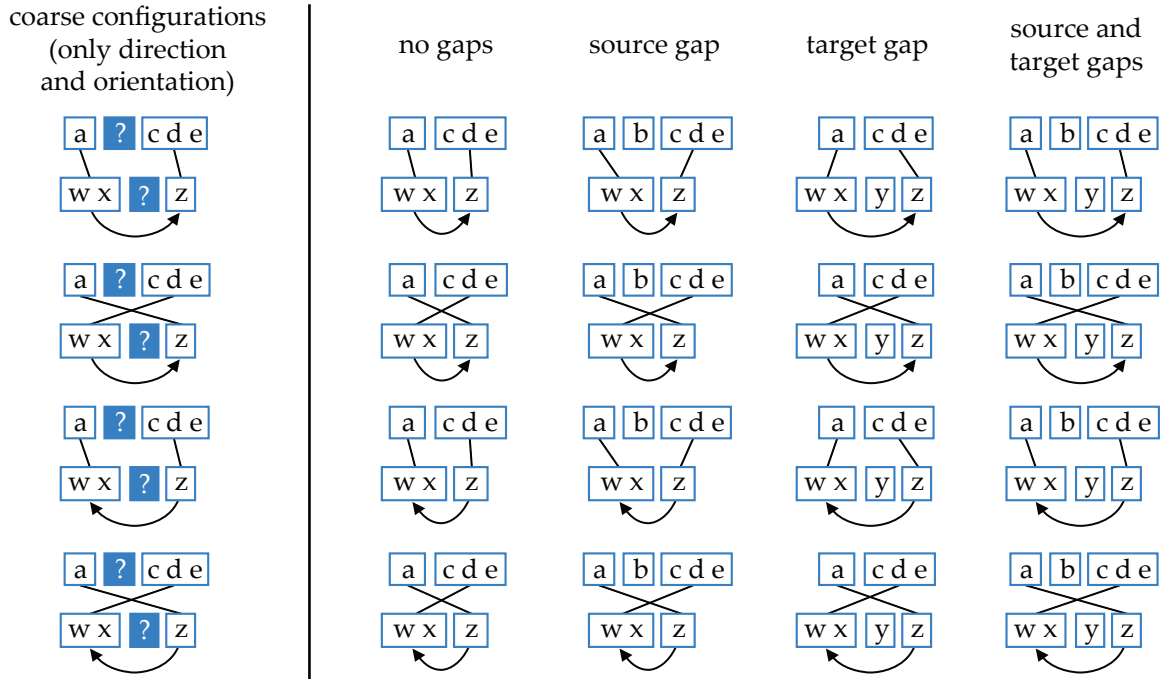


Figure 4.6: String-to-tree configurations; each is associated with a feature that counts its occurrences in a derivation.

by each phrase dependency. In particular, the feature sums the maximum number of target words that could be crossed by a particular phrase dependency. We will discuss how this feature is computed when we discuss decoding in Section 4.5.

4.4.4 String-to-Tree Configurations

We now present features that count instances of local reordering configurations involving phrase dependencies. These features consider $\boldsymbol{\phi}$, $\tau_{\boldsymbol{\phi}}$, and $\boldsymbol{b}$, though not $\boldsymbol{x}$, $\boldsymbol{\pi}$, $\tau_{\boldsymbol{x}}$, or $\boldsymbol{y}$.

Our first set of features only looks at configurations of direction and orientation. The first feature value is incremented if the child is to the left and the aligned source-side phrases are in the same order:

$$f_{\substack{\text{left}\\\text{same}}}(\boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b}) = \sum_{i=1}^{n'} \mathbb{I}\left[\text{dir}(\tau_{\boldsymbol{\phi}}, i) = \textsf{left} \wedge \text{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}) = \textsf{same}\right]$$

Another feature fires if the aligned source phrases are in the opposite order:

$$f_{\substack{\text{left}\\\text{swap}}}(\boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b}) = \sum_{i=1}^{n'} \mathbb{I}\left[\text{dir}(\tau_{\boldsymbol{\phi}}, i) = \textsf{left} \wedge \text{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}) = \textsf{swap}\right]$$

Analogous features are used when the child is to the right of the parent:

$$f_{\substack{\text{right}\\\text{same}}}(\boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b}) = \sum_{i=1}^{n'} \mathbb{I}\left[\text{dir}(\tau_{\boldsymbol{\phi}}, i) = \textsf{right} \wedge \text{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}) = \textsf{same}\right]$$

$$f_{\substack{\text{right}\\\text{swap}}}(\boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b}) = \sum_{i=1}^{n'} \mathbb{I}\left[\text{dir}(\tau_{\boldsymbol{\phi}}, i) = \textsf{right} \wedge \text{ori}(\tau_{\boldsymbol{\phi}}, i, \boldsymbol{b}) = \textsf{swap}\right]$$

These four configuration features are shown in order in the leftmost column in Figure 4.6. They are agnostic as to the presence of gaps between the two target phrases and between the two source phrases. We add 16 additional features that add gap information to these four coarse configurations, as shown in the remainder of the table. Four possible gap configurations are possible: {source gaps absent, source gaps present} × {target gaps absent, target gaps present}. We replicate the four coarse features above for each gap configuration, giving us a total of 20 string-to-tree configuration features, all shown in Figure 4.6.

4.4.5 Tree-to-Tree Configurations

The last two sets of features consider the source lexical dependency tree $\tau_{\boldsymbol{x}}$ in addition to $\boldsymbol{x}$, $\boldsymbol{\pi}$, $\boldsymbol{b}$, $\boldsymbol{y}$, $\boldsymbol{\phi}$, and $\tau_{\boldsymbol{\phi}}$. We begin with features for each of the quasi-synchronous configurations from Smith and Eisner (2006a), adapted to phrase dependency grammars. That is, for a child-parent pair $\langle \phi_i, \phi_{\tau_{\boldsymbol{\phi}}(i)} \rangle$ in $\tau_{\boldsymbol{\phi}}$, we consider the relationship between $\pi_{\boldsymbol{b}(i)}$ and $\pi_{\boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i))}$, the source-side phrases to which ϕ_i and $\phi_{\tau_{\boldsymbol{\phi}}(i)}$ align. We use the following named configurations from Smith and Eisner: root-root, parent-child, child-parent, grandparent-grandchild, sibling, and c-command. We define a feature to count instances of each of these configurations, including an additional feature for "other" configurations that do not fit into these categories.

There are several ways to compute tree-to-tree configuration features for Model P, since we use a phrase dependency tree for the target sentence, a lexical dependency tree for the source sentence, and a phrase alignment. We use a heuristic approach. First we find the full set of configurations that are present between any word in one source phrase and any word in the other source phrase. That is, given a pair of source words, one with index j in source phrase $\boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i))$ and the other with index k in source phrase $\boldsymbol{b}(i)$, we have a parent-child configuration if $\tau_{\boldsymbol{x}}(k) = j$; if $\tau_{\boldsymbol{x}}(j) = k$, a child-parent configuration is present. In order for the grandparent-grandchild configuration to be present, the intervening parent word must be outside both phrases. For sibling and other c-command configurations, the shared parent or ancestor must also be outside both phrases.

After obtaining a list of all configurations present for each pair of words $\langle j, k \rangle$, we fire the feature for the single configuration corresponding to the maximum distance $|j - k|$. If no con-

figurations are present between any pair of words, the "other" feature fires. Therefore, only one configuration feature fires for each phrase dependency attachment.

For the six configurations other than "root-root," we actually include multiple instances of each configuration feature: one set includes direction ($6 \times 2 = 12$ features), another set includes orientation (12 features), and the final set includes both source- and target-side gap information (24 features). So there are 49 features in this category (including the single "root-root" feature), giving us a total of 86 features thus far.

4.4.6 Tree-to-Tree Dependency Path Length Features

Finally, we include features that consider the **dependency path length** between the source phrases aligned to the target phrases in each phrase dependency. The features in Section 4.4.3 considered distance along the source *sentence* (the number of words crossed). Now we add features that consider distance along the source *tree* (the number of dependency arcs crossed). We expect the learned weights for these features to be positive in order to encourage short dependency path lengths on the source side.

We first include a feature that sums, for each target phrase i, the inverse of the minimum undirected path length between each word in $\boldsymbol{b}(i)$ and each word in $\boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i))$:

$$f_{\substack{\text{undir}\\\text{path}}}(\boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}) = \sum_{i=1}^{m'} \sum_{j=i'}^{j'} \sum_{k=k'}^{l'} \frac{1}{\text{minUndirPathLen}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)}$$

where $\boldsymbol{b}(i) = \boldsymbol{x}_{i'}^{j'}$, $\boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i)) = \boldsymbol{x}_{k'}^{l'}$, and $\text{minUndirPathLen}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)$ returns the shortest undirected dependency path length from x_j to x_k in $\tau_{\boldsymbol{x}}$. The shortest undirected path length is defined as the number of dependency arcs that must be crossed to travel from one word to the other along the arcs in $\tau_{\boldsymbol{x}}$.

Assuming an analogous function $\text{minDirPathLen}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)$ that computes the minimum *directed* dependency path length, we also include the following feature:

$$f_{\substack{\text{dir}\\\text{path}}}(\boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}) = \sum_{i=1}^{m'} \sum_{j=i'}^{j'} \sum_{k=k'}^{l'} \frac{1}{\text{minDirPathLen}(\boldsymbol{x}, \tau_{\boldsymbol{x}}, j, k)}$$

where again $\boldsymbol{b}(i) = \boldsymbol{x}_{i'}^{j'}$ and $\boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i)) = \boldsymbol{x}_{k'}^{l'}$. If there is no directed path from a word in $\boldsymbol{b}(i)$ to a word in $\boldsymbol{b}(\tau_{\boldsymbol{\phi}}(i))$, minDirPathLen returns ∞.

Adding the two features above gives us a total of 88 QPDG features. Along with the 14 phrase-based features there are a total of 102 features in our model.

4.5 Decoding

For Model P, decoding consists of solving Eq. 4.2, i.e., finding the highest-scoring tuple $\langle \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b} \rangle$ for an input sentence $\boldsymbol{x}$ and its parse $\tau_{\boldsymbol{x}}$. This is a challenging search problem,

since it is at least as hard as the search problem for phrase-based models, which is intractable to solve exactly (see Section 2.2.1). So we use a **coarse-to-fine** strategy for decoding (Charniak and Johnson, 2005; Petrov, 2009). Coarse-to-fine inference is a general term for procedures that make multiple passes over the search space, pruning the space with each pass. Typically, feature complexity is increased in each pass, as richer features can often be computed more easily in the smaller search space.

One simple coarse-to-fine procedure for Model P would start by generating a k-best list of derivations using a phrase-based decoder. This would account for all of the phrase-based features (those with weights $\boldsymbol{\lambda}$). Then we could parse each derivation to incorporate the QPDG features and rerank the k-best list with the modified scores. The advantage of this approach is its simplicity, but other research has shown that k-best lists for structured prediction tend to have very little diversity (Huang, 2008), and we expect even less diversity in cases like machine translation where a latent derivation variable is always present. Instead, we generate a phrase lattice (Section 2.2.1; Ueffing et al., 2002) in a coarse pass and perform **lattice dependency parsing**.

The remainder of this section is laid out as follows. In Section 4.5.1 we present our basic lattice dependency parsing algorithm. We give three ways to speed it up in Section 4.5.2; one enables a more judicious search without affecting the search space, and the other two prune the search space in different ways. In Section 4.5.3, we discuss how our decoder interacts with learning, including k-best list generation and learning separate feature weights for coarse and fine models. We close in Section 4.5.4 with a brief discussion of how this decoder differs from earlier versions published in Gimpel and Smith (2009b, 2011a).

4.5.1 Lattice Dependency Parsing

As we discussed in Section 2.2.1, phrase lattices concisely encode the pruned search space of a phrase-based model. Each edge in a phrase lattice corresponds to a phrase pair and each path from a designated start node to any of the designated final nodes corresponds to a tuple $\langle \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b} \rangle$ for the input $\boldsymbol{x}$.[6] Decoding for a phrase lattice consists of finding the highest-scoring path, which is done using dynamic programming. To also maximize over $\tau_{\boldsymbol{\phi}}$, we perform **lattice dependency parsing**, which allows us to search over the space of tuples $\langle \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{b}, \tau_{\boldsymbol{\phi}} \rangle$. Lattice parsing jointly maximizes over paths through a lattice and syntactic trees on those paths.

Since we use an arc-factored phrase dependency grammar in Model P, the lattice dependency parsing algorithm we use is a straightforward generalization of the arc-factored dynamic programming algorithm from Eisner (1996). The algorithm is shown in Figure 4.7. It is shown as a set of recursive equations in which shapes are used in place of function names and shape indices are used in place of function arguments. A semiring-generic format is used; for decoding, the semiring "plus" operator ($\oplus$) would be defined as $\max$ and the semiring "times" operator ($\otimes$) would be defined as $+$. The entry point when executing the algorithm is to build GOAL, which in turn requires building the other structures.

[6] To prevent confusion, we use the term **edge** to refer to a phrase lattice edge and **arc** to refer to a dependency attachment in a dependency tree.

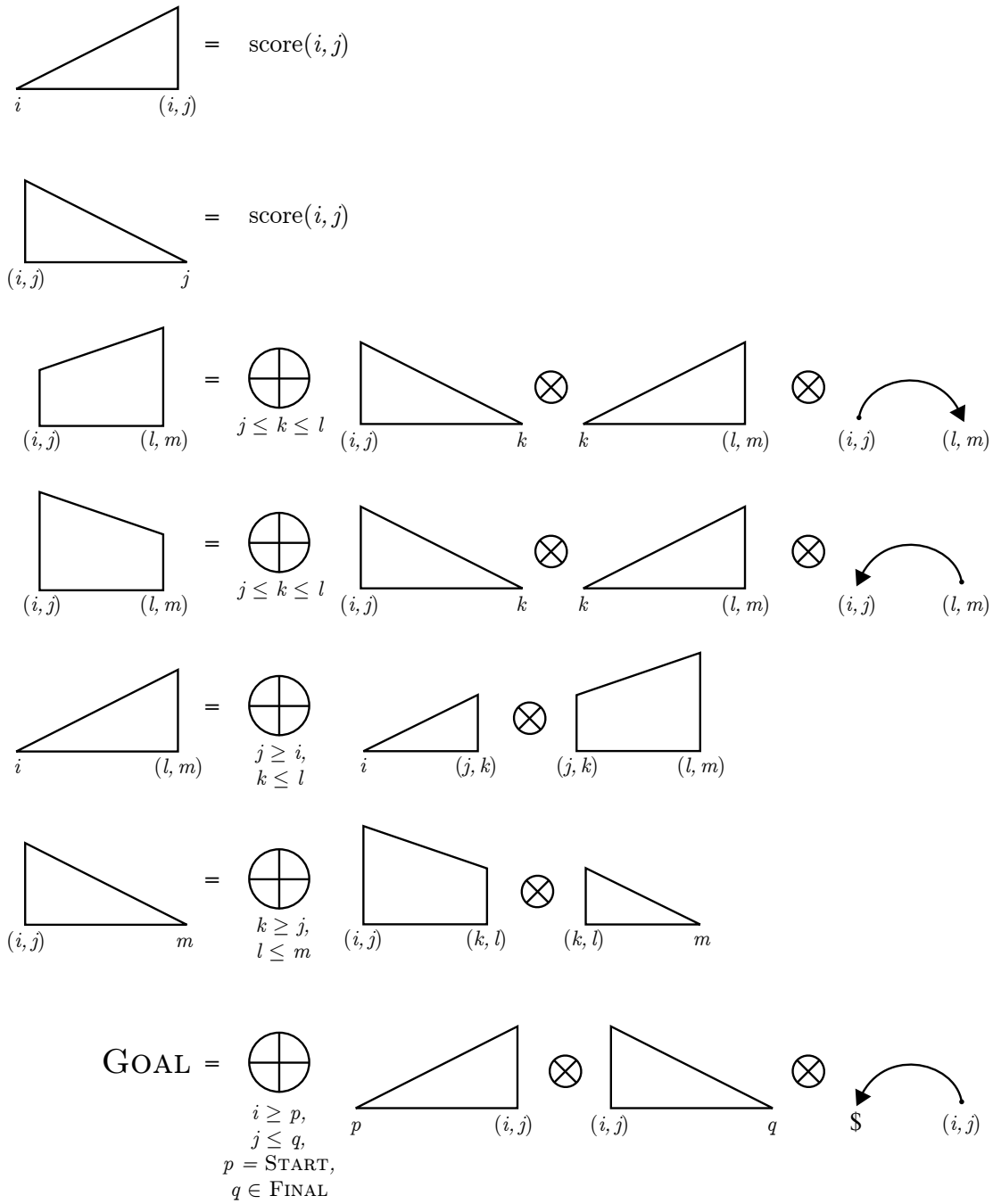


Figure 4.7: Lattice dependency parsing using an arc-factored dependency model. Lone indices like p and i denote nodes in the lattice while an ordered pair like (i, j) denotes the lattice edge from node i to node j. START is the single start node in the lattice and FINAL is a set of final nodes. We use $\text{score}(i, j)$ to denote the model score of crossing lattice edge (i, j), which only includes the phrase-based features. This accounts for the edge scores, so the model score of a dependency arc only uses the QPDG features.

We use a simple top-down implementation with memoization. Our style of specifying dynamic programming algorithms is similar to weighted deduction, but additionally specifies indices and ranges of iteration, which are useful for a top-down implementation. Top-down dynamic programming avoids the overhead of maintaining a priority queue that is required by bottom-up agenda algorithms (Nederhof, 2003; Eisner et al., 2005).

The disadvantage of top-down dynamic programming is that wasted work can be done; structures can be built that never ground out in any full parse. This problem appears when parsing with context-free grammars, and so the CKY algorithm works bottom-up, starting with the smallest constituents and incrementally building larger ones. This is because context-free grammars may contain rules with only non-terminals. Top-down execution may consider the application of such rules in sequence, producing long derivations of non-terminals that never ground out in any symbols in the string. A traditional dependency grammar, on the other hand, always works directly on words when building items, so a top-down implementation can avoid wasted effort.

However, the situation changes with *lattice* dependency parsing. It is possible for a top-down lattice dependency parser to consider some dependencies that could never ground out in a full parse. We address this issue in the next section.

4.5.2 Computational Complexity and Speeding Up Decoding

The lattice parsing algorithm requires $O(E^2V)$ time and $O(E^2+VE)$ space, where E is the number of edges in the lattice and V is the number of nodes. Typical phrase lattices might easily contain tens of thousands of nodes and edges, making exact search prohibitively expensive for all but the smallest lattices. So we use three techniques to speed up decoding: (1) avoiding construction of impossible items, (2) pruning the lattices, and (3) limiting the maximum length of a phrase dependency.

Avoiding Impossible Paths

By design, our phrase lattices impose several types of natural constraints on allowable dependency arcs. For example, each node in the phrase lattice is annotated with a coverage vector—a bit vector indicating which words in the source sentence have been translated—which implies a topological ordering of the nodes. This can tell us whether certain nodes are reachable from other nodes.

But other constraints exist. Generally, we need an efficient way to determine, for any two nodes in the lattice, whether there exists a path from one to the other. If there is no path, we can avoid wasting time figuring out the best way to build items that would end at the two nodes. To discover this, we can use an all-pairs shortest paths algorithm to find the score of the best path between each pair of nodes in the lattice. The algorithm also tells us whether each edge is reachable from each other edge, allowing us to avoid drawing dependencies that will never ground out in a lattice path. In particular, we use the Floyd-Warshall algorithm (Floyd, 1962). This adds some initial overhead to decoding, but in preliminary experiments we found that it saves more time than it costs. We actually run a modified version of the algorithm that computes the length (in words)

of the longest path between any two nodes. If the maximum length between two nodes is ∞, the nodes are unreachable from each other. When building items in the algorithm in Figure 4.7, we first check whether the node endpoints of the item are reachable from one another, and only proceed if so.

The reason we modified the algorithm to output maximum lengths is because we want to use the maximum lengths to compute the target-side vine grammar features and constraints, as mentioned above in Section 4.4.3. In particular we use a feature $f_{\underset{\text{vine}}{\text{tgt}}}$ that is a target-side analog to $f_{\underset{\text{vine}}{\text{src}}}$ but using the Floyd-Warshall maximum path lengths in place of the actual lengths.

Lattice Pruning

To reduce phrase lattice sizes, we prune lattice edges using forward-backward pruning (Sixtus and Ortmanns, 1999), which has also been used by Tromble et al. (2008). This pruning method computes the max-marginal for each lattice edge, which is the score of the best full path that uses that edge, then prunes edges whose max-marginal is below a certain fraction of the best path score in the lattice. Max-marginals have been used for other coarse-to-fine learning frameworks (Sapp et al., 2010) and offer the advantage that the best path in the lattice is preserved during pruning.

For each lattice, we use a grid search to find the most liberal threshold that leaves fewer than 2000 edges in the resulting lattice. As complexity is quadratic in E, forcing E to be less than 2000 improves runtime substantially. After pruning, the lattices still contain more than 10^{16} paths on average and oracle BLEU scores are typically 10-15 points higher than the model-best paths.

Maximum Dependency Lengths

We can easily adapt our vine grammar features to function as hard constraints on allowable dependency trees, as originally done by Eisner and Smith (2005) for monolingual dependency parsing. We use two simple constraints on the maximum length of a phrase dependency used during parsing. One constrains the number of words that are crossed from one aligned source phrase to the other aligned source phrase by the phrase dependency. The other constrains the maximum number of target-side words crossed by any path from one target phrase to the other target phrase in a phrase dependency. During parsing, we avoid building items that would require using dependency arcs that violate these constraints. In Chapter 5 we discuss the values we used in our experiments and compare translation quality and decoding speed for several values.

4.5.3 Interaction with Learning

An outline of the decoding procedure is shown as Algorithm 3. We note that two separate versions of the phrase-based feature weights $\boldsymbol{\lambda}$ are used: one for lattice generation and one for lattice dependency parsing. This is common with coarse-to-fine strategies—separate instances of coarser parameters are required for each subsequent pass.

For learning the coarse and fine parameters, we use the procedure shown as Algorithm 4. We first train parameters for the coarse phrase-based model used to generate phrase lattices. Then,

Input: sentence $\boldsymbol{x}$, dependency parse $\tau_{\boldsymbol{x}}$, coarse parameters $\boldsymbol{\lambda}_C$, fine parameters $\langle \boldsymbol{\lambda}_F, \boldsymbol{\psi}_F \rangle$
Output: translation $\boldsymbol{y}$

1 $\boldsymbol{L}_{\text{Moses}} \leftarrow$ `GenerateLattices` ($\boldsymbol{x}$, $\boldsymbol{\lambda}_C$);
2 $\boldsymbol{L}_{\text{FB}} \leftarrow$ `FBPrune` ($\boldsymbol{L}_{\text{Moses}}$, $\boldsymbol{\lambda}_C$);
3 $\langle \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{b} \rangle \leftarrow$ QGDEPPARSE($\boldsymbol{L}_{\text{FB}}$, $\langle \boldsymbol{\lambda}_F, \boldsymbol{\psi}_F \rangle$);
4 **return** $\boldsymbol{y}$;

Algorithm 3: `CoarseToFineDecode`

Input: input sentences $\tilde{\boldsymbol{X}}$, reference translations $\tilde{\boldsymbol{Y}}$, initial weights $\boldsymbol{\lambda}_0$ for coarse model, initial weights $\boldsymbol{\psi}_0$ for additional features in fine model
Output: coarse model learned weights: $\boldsymbol{\lambda}_C$, fine model learned weights: $\langle \boldsymbol{\lambda}_F, \boldsymbol{\psi}_F \rangle$

`// Run Train to get coarse (Moses-only) feature weights`
1 $\boldsymbol{\lambda}_C \leftarrow$ `Train` ($\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{Y}}$, $\boldsymbol{\lambda}_0$, 500, MOSES);
`// Generate phrase lattices with learned weights`
2 $\boldsymbol{L}_{\text{C}} \leftarrow$ `GenerateLattices` ($\tilde{\boldsymbol{X}}$, $\boldsymbol{\lambda}_C$);
`// Prune lattices`
3 $\boldsymbol{L}_{\text{FB}} \leftarrow$ `FBPrune` ($\boldsymbol{L}_{\text{C}}$, $\boldsymbol{\lambda}_C$);
`// Run Train again to get fine (Moses and QPDG) feature weights`
4 $\langle \boldsymbol{\lambda}_F, \boldsymbol{\psi}_F \rangle \leftarrow$ `Train` ($\boldsymbol{L}_{\text{FB}}$, $\tilde{\boldsymbol{Y}}$, $\langle \boldsymbol{\lambda}_C, \boldsymbol{\psi}_0 \rangle$, 150, QGDEPPARSE);
5 **return** $\boldsymbol{\lambda}_C$, $\langle \boldsymbol{\lambda}_F, \boldsymbol{\psi}_F \rangle$;

Algorithm 4: `CoarseToFineTrain`

after generating the lattices, we prune them and train a second time to learn parameters of the fine model, which includes all phrase-based and QPDG parameters. The arguments to the `Train` function are a vector of input sentences (or lattices), a vector of reference translations, the initial parameter values, the size of the k-best list, and finally the decoder to use. For initialization we use the default Moses feature weights for $\boldsymbol{\lambda}_0$ and for $\boldsymbol{\psi}_0$ we initialize the phrase dependency probability feature weights to 0.002 and the weights for all other features to 0.0.

For the learning algorithms from Chapter 3, we need the k best derivations, for which efficient dynamic programming algorithms exist. We use Algorithm 3 from Huang and Chiang (2005), which lazily finds the k best derivations. In preliminary testing, we found that the k-best lists tended to be dominated by repeated translations with different derivations, so we used the technique presented by Huang et al. (2006) which finds a unique k-best list, returning the highest-scoring derivation for each of k unique translations. This modification requires the maintenance of additional data structures to store all of the previously-found string yields for each item built during parsing. This incurs additional overhead but allows us to obtain a far more diverse k-best list given a fixed time and memory budget.

One concern, however, is that finding a unique k-best list does not match the assumptions that we made in Chapter 3, namely that the k-best lists held the k-best derivations. But unique

k-best lists are actually appropriate for certain loss functions. In particular, ramp losses 1-3 only require cost-augmented or cost-diminished decoding over the k-best lists, but the solutions to these cost-sensitive decoding problems are not changed if inferior *derivations* for a given translation are removed from the k-best lists. Such translations will all have the same cost, so the ones with the highest model score will always be chosen by cost-augmented or cost-diminished decoding. Therefore, we can still train our model with ramp loss even though we use a unique k-best algorithm. We use ramp loss 3 in the experiments in the next chapter as it was the best-performing of the (hard) ramp losses.

4.5.4 Comparison to Earlier Work

The decoder described above differs from those previously published. Our initial decoder was designed for Model W: we built lattices in which each lattice edge contained a single word pair (Gimpel and Smith, 2009b). Approximate inference was done using **cube decoding** (Gimpel and Smith, 2009a), an algorithm that incorporates non-local features in a way similar to cube pruning (Chiang, 2007). After developing Model P, we moved to phrase lattices but still used approximate inference using an agenda algorithm (Nederhof, 2003; Eisner et al., 2005) with pre-pruning of dependency edges in a coarse pass (Gimpel and Smith, 2011a).

All decoders used lattice dependency parsing, but our current decoder uses an exact search algorithm given the pruned lattice and maximum dependency length constraints. Using an exact algorithm allows us to train our model with exact inference. Training with exact inference has been observed in other applications to make learning more stable, and we found this in preliminary experiments as well. Using an exact decoder also helps us to better understand the contributions of our model; we can alleviate concerns about the quality of our approximate search and the effects of approximations on training and decoding.

4.6 Conclusion

The primary contribution of this chapter is the definition of a translation model based on quasi-synchronous phrase dependency grammar; more specifically,

- We studied syntactic divergence in a parallel corpus that had been automatically word-aligned and parsed on both sides and provided examples of non-isomorphisms (Section 4.1).
- We defined **phrase dependency grammars** and used them to define our novel quasi-synchronous translation model, which we call Model P (Section 4.2).
- We introduced several categories of features for translation based on dependency syntax, including both string-to-tree and tree-to-tree features (Section 4.4).
- We proposed **lattice dependency parsing** to solve the decoding problem (Section 4.5) and presented ways to speed up search and prune the search space.

The next chapter includes experiments and analysis using Model P.

Chapter 5

Experiments and Analysis

We now present experimental results using the model we presented in the previous chapter ("Model P"). Since Model P generalizes phrase-based translation models and also includes features on a syntactic derivation structure, we can conduct experiments that simulate phrase-based, string-to-tree, and tree-to-tree translation, merely by specifying which feature sets to include. The goal of this line of research is to build a flexible modeling framework that uses soft constraints and rich feature sets to generalize existing models. We hope to be able to simulate a variety of common translation systems in a single model, thereby combining the strengths of diverse approaches and allowing clean experimental comparisons among modeling strategies. In this chapter, we take initial steps in this direction. We also describe several experiments that use *unsupervised* parsing with the hope of applying Model P to many additional language pairs.

We include several types of experiments. First we test our model for language pairs with supervised parsers, comparing the contributions of feature sets. For German-to-English and Chinese-to-English translation, we have supervised parsers for both languages so we can use our full feature sets for these languages. For Urdu-to-English translation, we can only use target syntactic features since we do not have a supervised parser for Urdu. We analyze the output of our model, showing examples. We then turn to unsupervised parsers, first substituting them for supervised parsers in Chinese-to-English translation, and then using them to train our full model for Urdu-to-English and English-to-Malagasy translation. To go beyond phrase-based translation, our model requires a parser for the target language, so we need unsupervised parsing to do syntax-based English-to-Malagasy translation.

This chapter is laid out as follows. We start with our experimental setup in Section 5.1, then present our first set of results, which uses our full model and supervised parsers, in Section 5.2. Then we discuss unsupervised parsing and present results using unsupervised parsers in Section 5.3. We report on an experimental analysis of our decoder in Section 5.4. In Section 5.5, we briefly report on results obtained with an earlier version of the model and decoder, and discuss how they differ from the primary results reported here. We conclude the chapter in Section 5.6.

This chapter contains material originally published in (Gimpel and Smith, 2011a, 2012b) as well as additional experiments and analysis. Appendix B contains additional material about our

ZH→EN		tune	test 1	test 2	test 3	test avg.
Moses	stack size = 200	36.02	35.48	34.30	31.27	33.68
	stack size = 500	36.15	36.13	34.58	**31.75**	34.15
Model P	target syntax	36.76	35.71	34.37	31.67	33.92
	+ source syntax	**37.39**	**36.37**	**34.91**	31.55	**34.28**

Table 5.1: %BLEU on tune and test sets for ZH→EN translation, showing the contribution of feature sets in Model P. The highest BLEU score in each column is bold.

unsupervised parsers.

5.1 Experimental Setup

In this section we describe details common to the experiments reported in this chapter. The decoding and learning pipelines were described in Section 4.5.3. Full details about language pairs, data sets, and baselines are given in Appendix C. We highlight important details here.

Across all experiments we fixed the learning algorithm to be Algorithm 1 (from Chapter 3) set to minimize $\text{loss}_{\text{ramp 3}}$. We chose this loss because it performs competitively in both small-feature and large-feature settings (Section 3.5) and because it is compatible with our use of unique k-best lists (see discussion in Section 4.5.3).

As a baseline, we report scores when training a standard phrase-based Moses system. We chose Moses because it also provides the search space for Model P through phrase lattice generation. We trained Moses models for each language pair using ramp loss 3. Then we found the parameters that led to the highest BLEU score on the tuning set. We used these parameters to decode the test sets and list the resulting scores as baseline Moses BLEU scores in each table. We used the same parameters to generate phrase lattices for the tuning and test sets. The phrase lattices for the tuning set are used when learning the fine parameters as described in Section 4.5.3. The phrase lattices for the test sets are used when decoding with Model P. When generating these phrase lattices, we increased the Moses stack size to 500 from its default value of 200 in an attempt to increase the diversity of the lattices. We found that this frequently improved the BLEU scores of the model-best path through the lattice, so we also report these BLEU scores. They are listed as "Moses, stack size = 500." The default Moses stack size is 200.

5.2 Results with Supervised Parsers

We start by presenting results when only using supervised parsers. We used the Stanford parser (Levy and Manning, 2003; Rafferty and Manning, 2008) for Chinese and German and Tur-

DE→EN		tune	test 1
Moses	stack size = 200	16.21	18.95
	stack size = 500	16.21	19.15
Model P	source and target syntax	**16.86**	**19.41**

Table 5.2: %BLEU on tune and test set 1 for DE→EN translation, comparing the phrase-based baselines to Model P with both source and target syntax.

UR→EN		tune	test 1	test 2	test avg.
Moses	stack size = 200	24.58	24.61	24.46	24.54
	stack size = 500	24.72	24.77	24.89	24.83
Model P	target syntax	**25.55**	**24.91**	**25.03**	**24.97**

Table 5.3: %BLEU on tune and test sets for UR→EN translation, showing the contribution of target syntax features. We do not have a supervised Urdu parser, so we need unsupervised parsing to incorporate source syntactic features (Section 5.3).

boParser (Martins et al., 2009) for English. Our results for ZH→EN and DE→EN translation are shown in Tables 5.1 and 5.2, respectively. We see that enlarging the search space results in gains in BLEU, as Moses with stack size 500 typically outperforms the Moses baseline. When using both source and target syntax, tuning BLEU scores increase and a portion of this increase transfers to the test sets, although the gains are not large. We note that when only using target syntax, Model P underperforms Moses with stack size 500. Adding source syntax is required to boost the BLEU scores above Moses on average, although Moses still wins on test set 3.

Results for UR→EN translation are shown in Table 5.3. We again see small gains on the test sets, but these results only use features that consider target-side syntax, since we do not have a supervised parser for Urdu.[1]

Nonetheless, even though BLEU improvements are modest, we manually inspected the output to attempt to find systematic differences. We looked at the first 223 sentence pairs in the DE→EN test set. Of these, 43 translations were identical between Moses and Model P. Of the remaining, we determined 99 to be of roughly equivalent translation quality, and 81 to have a difference. Of these 81, in 31 cases the Moses output was better and in 50 cases the Model P output was better. Notably, in 19 of the 50 cases, the translation of the verb was improved. Mis-translation and dropping of verbs is a common problem in German-to-English translation due to the different placement of German verbs as compared to English, as we discussed in Section 4.3.3 (e.g., see Figure 4.5). Some

[1]We note that these gains over the baseline are smaller than those previously published (Gimpel and Smith, 2011a), due to several differences in the experimental setup; we return to this point in Section 5.5.

source	kongress macht zugeständnis : us-regierung darf 700 milliarden dollar in die banken pumpen
reference	congress yields : us government can pump 700 billion dollars into banks
Moses	congress makes concession : us government $ 700 billion in banks must pump
Model P	congress makes concession : us government is pouring $ 700 billion in banks
source	damit wolle man nämlich die investoren vor geschäften auf den asiatischen finanzmaerkten beruhigen , die in der ersten zeitzone liegen , wo eine entscheidung des kongresses die geschäfte am montag beeinflussen könnte .
reference	namely , by doing this they want to calm investors prior to trading on the asian financial markets , which , given their time zones , are the first ones where the decision by congress could influence monday 's trading .
Moses	this may be the investors doing business in the asian financial markets , in the first zone , where a decision the influence of the congress on monday .
Model P	so we want to reassure investors in shops in the asian markets in the first zone , where a decision of the congress could affect the business on monday .
source	zwei extreme , zwischen denen es für jeden genügend raum gibt .
reference	two extremes , between which there is enough space for everyone .
Moses	between those two extremes for each room .
Model P	between those two extremes , there is enough room for everyone .
source	über diese möglichkeit haben wir wiederholt mit allen tschechischen mobilfunkanbietern verhandelt .
reference	we have held repeated negotiations on this possibility with all czech mobile operators .
Moses	this way , we have repeatedly with all czech mobilfunkanbietern .
Model P	in this way , we have repeatedly negotiated with all the czech mobilfunkanbietern .
source	wenn sie lust haben , sich etwas anzuhören , greifen sie nach dem handy in der jackentasche , während der player vergessen zu hause liegt .
reference	whenever you feel like listening to something , you can reach for your cell phone in your pocket , while the mp3 player lies at home , forgotten .
Moses	if you want something , you have to listen to the mobile phone in the jackentasche , while the players to forget .
Model P	if you want something , you have to listen to the mobile phone in the jackentasche , while the player is forgotten at home .
source	der vorteil virtueller anbieter besteht dabei darin , dass man sich auf eine viel konkretere zielgruppe konzentrieren kann .
reference	the advantage of the virtual operators is the opportunity to concentrate on a much more specific target group .
Moses	the advantage is in virtual providers that it can focus on a much more specific target .
Model P	the advantage of virtual suppliers , it is that you can focus on a much more specific target .

Table 5.4: Examples from DE→EN test set in which the translation of the verb improved.

	average # words per phrase	
	ZH→EN	DE→EN
Moses, stack size = 500	1.66	1.45
Model P, source and target syntax	1.60	1.16

Table 5.5: Comparing the average phrase lengths in Moses and Model P output on test set 1 for ZH→EN and DE→EN translation.

examples of improved verb translations are shown in Table 5.4. We also found many instances of improved verb translations in the ZH→EN output.

We do not intend for this analysis to serve as a thorough manual evaluation of our system; it is an informal analysis with the intent of categorizing translation improvements. But the fact that improved verb translations were a noticeable trend gives us hope that human evaluators would prefer our system in a more thorough manual evaluation setting.

5.2.1 Analysis of Phrase Dependency Trees

What do the phrase dependency trees look like? We found that they differed significantly for different language pairs. In the DE→EN output, we noticed that the phrase dependency trees frequently looked very much like lexical dependency trees in a treebank. Most phrases are very short and the attachments and their directions are very similar to what an annotator would produce. For ZH→EN translation, by contrast, the trees used longer phrases frequently and the phrase dependencies were often in the opposite direction of treebank annotations; a typical ZH→EN phrase dependency tree is shown in Figure 4.4. When we compute the average length of target-side phrases used in the translations, we find a clear drop in phrase length when using Model P for DE→EN translation. For ZH→EN translation, however, there was only a slight decrease. Table 5.5 shows the comparison.

Table 5.6 shows the most frequent English phrases used by each model for DE→EN translation. Only phrases with at least two words are shown. One difference between the two lists is that the Model P phrases are more likely to be English phrases that are generated from a single German word. For example, the phrase *of the* appears more than twice as often in the Model P output as in the Moses output. When *of the* is expressed in German, it is often the case that the preposition is dropped and the definite article is chosen to express genitive case, as discussed in Section 4.3.2. So, a single German word (the genitive definite article) naturally translates into a phrase of two English words, and Model P does indeed use phrasal translation rules in this case.

Conversely, the phrases *and the* and *with the* appear frequently in the Moses output but do not appear very often in the Model P output; these are both phrases that typically translate to two-word phrases in German and, furthermore, the phrase translation decomposes naturally into the

Moses		Model P	
count	phrase	count	phrase
258	of the	602	of the
190	in the	179	in the
126	, the	70	, ″
80	, ″	40	of a
71	it is	34	such as
70	and the	32	according to
69	on the	26	on the
54	, and	26	however ,
52	that the	25	, the
51	with the	23	it is
50	to the	22	prime minister
47	. ″	22	, and
43	, but	22	a few
33	for the	20	to the
32	, which	20	. ″
31	from the	19	will be
30	will be	19	united states
30	by the	19	stock exchange
28	is the	16	of their
28	however ,	15	the czech republic
26	″ the	14	one of the
24	such as	14	, however ,
23	according to the	14	by the
22	this is	14	at least
22	of a	13	would be
21	, as	13	part of
19	″ we	13	of this
19	it was	13	has been
19	, however ,	13	financial crisis
19	during the	12	will be
18	to be	12	so that
18	because of the	12	mobile phones
18	, a	12	it is
18	) .	12	have been
17	stock exchange	11	up to
17	, ″ said	11	, ″ said
17	more than	11	in order
17	in a	11	, but
16	would be	10	to be
16	, it is	10	there is
16	is a	10	there are
16	i am	10	kind of

Table 5.6: Comparing the most frequent English phrases used in the Moses and Model P output for DE→EN translation. Only phrases with at least two words are shown.

individual word translations. So, it appears that Model P makes use of phrasal translation rules when they are needed but otherwise prefers to use lexical translation rules.

We suspect this tendency also relates to the amount of data used and issues related to data sparsity. Model P constructs a dependency tree on the target phrases using features based on conditional probabilities of individual parent-child phrase attachments. The probabilities will be more robustly-estimated for frequently-occurring phrases which, all else being equal, tend to be short. We suspect that using more data will improve probability estimates for longer phrases and thereby increase the average phrase length used during translation, thereby gaining more benefits of phrase-based translation modeling. Nonetheless, for now we simply note the result that, by adding dependency syntax, we can maintain and even increase translation quality for DE→EN while reducing the average phrase length.

5.3 Unsupervised Parsing

Tree-to-tree models like Model P can use syntactic parsers for both languages, but there are many languages for which (supervised) parsers do not exist, such as Urdu and Malagasy. Therefore, *unsupervised* parsers, which induce linguistic structure from unannotated text, are necessary to apply syntax-based translation to these languages.

In this section, we conduct experiments using unsupervised parsers with Model P. We explore several different scenarios that reflect real-world usage, including cases in which we have access to a part-of-speech tagger or a parser for one of the two languages.

The most common approach to unsupervised parsing is to train models on sentences from treebanks (without using the annotated trees, of course) along with their gold standard part-of-speech (POS) tags. This must be addressed if we wish to use unsupervised parsers for machine translation. Fortunately, Spitkovsky et al. (2011a) showed that using automatic POS tags for grammar induction can work better than gold-standard POS tags. For syntax-based translation, Zollmann and Vogel (2011) showed that unsupervised tags could work as well as those from a supervised POS tagger.

For Urdu and Malagasy, we use fully-unsupervised POS tagging followed by fully-unsupervised dependency parsing. For POS tagging, we use the approach from Berg-Kirkpatrick et al. (2010) with 40 tags. We use the "direct gradient" version optimized by LBFGS (Liu and Nocedal, 1989). For Chinese and English, we use the gold standard POS tags from their respective treebanks for training the parser, and then use the Stanford POS tagger (Toutanova et al., 2003) to tag the parallel data, tuning, and test sets. Our unsupervised parsers are described in Appendix B.

We also compared the supervised and unsupervised parsers to a *random* parser. We used `cdec` (Dyer et al., 2010) to sample projective dependency trees uniformly at random for each sentence.[2] Well-known algorithms exist for sampling derivations under a context-free grammar for a sentence (Johnson et al., 2007). These algorithms can be used to sample projective dependency trees by representing a projective dependency grammar using a context-free grammar (Smith, 2006; Johnson, 2007).

[2]We thank Chris Dyer for implementing this feature in `cdec` for us.

	tune	test 1	test 2	test 3	test avg.
Moses, stack size = 500	36.15	36.13	34.58	31.75	34.15

	EN parser	ZH parser	tune %BLEU	test 1 %BLEU	test 2 %BLEU	test 3 %BLEU	avg. test %BLEU
Model P	sup.	sup.	**37.39**	**36.37**	**34.91**	31.55	**34.28**
		unsup.	37.14	36.20	34.71	**31.92**	**34.28**
		rand.	36.61	35.67	33.97	31.71	33.78
	unsup.	sup.	36.98	35.72	34.34	31.63	33.90
		unsup.	37.16	36.12	34.23	31.77	34.04

Table 5.7: %BLEU on tune and test sets for ZH→EN translation, comparing supervised and unsupervised parsers for the source and target languages. Model P uses all features, including phrase-based, string-to-tree, and tree-to-tree. The upper table shows the Moses BLEU scores for comparison. The lower table first pairs supervised English parsing with supervised, unsupervised, and random Chinese parsing, then pairs unsupervised English parsing with supervised and unsupervised Chinese parsing. Using unsupervised parsing beats random parsing and occasionally ties or exceeds supervised parsing, particularly for Chinese.

We only compared the random parser for *source*-side parsing. Swapping parsers for the target language requires parsing the target side of the parallel corpus, rerunning rule extraction and feature computation with the new parses, and finally re-tuning to learn new feature weights. By contrast, changing the source-side parser only requires re-parsing the source side of the tuning set and then rerunning the learning procedure.

UR→EN		tune	test 1	test 2	test avg.
Moses	stack size = 200	24.58	24.61	24.46	24.54
	stack size = 500	24.72	24.77	24.89	24.83
Model P	target syntax	25.55	24.91	25.03	24.97
	+ unsup. source syntax	**26.31**	**24.96**	**25.37**	**25.17**

Table 5.8: %BLEU on tune and test sets for UR→EN translation, showing the contribution of source syntax features using our unsupervised Urdu parser.

EN→MG		tune	test 1
Moses	stack size = 200	17.56	15.05
	stack size = 500	17.75	15.08
Model P	source and unsup. target syntax	**18.23**	**15.54**

Table 5.9: %BLEU on tune and test sets for EN→MG translation, using a supervised English parser and an unsupervised Malagasy parser.

5.3.1 Results and Discussion

Table 5.7 shows results when comparing parsers for ZH→EN translation. We pair supervised and unsupervised parsers for English and Chinese. We find that using our unsupervised English parser hurts performance, causing it to be worse than the Moses baseline on average. Nonetheless, when using unsupervised English parsing, the unsupervised Chinese parser works as well as the supervised Chinese parser.

When using supervised English parsing, we find that using our unsupervised Chinese parser in place of the Stanford parser leads to the same average test set BLEU score. When using unsupervised parsers, BLEU scores drop, but all are better than using a random parser. We note further that adding the tree-to-tree features based on a random parser actually leads to worse performance than omitting the tree-to-tree features entirely, by comparing to Table 5.1.

Unsupervised parsing allows us to apply Model P with the full feature set to UR→EN and EN→MG translation; results are shown in Tables 5.8 and 5.9. We see modest but consistent improvement in BLEU.

One idea that we have not explored would be to parse our parallel corpus using each parser (unsupervised and supervised), then extract rules consistent with any of the parses. This might give us some of the benefits of forest-based rule extraction, which has frequently been shown to improve performance (Liu et al., 2007; Mi et al., 2008; Mi and Huang, 2008). Similarly, since we train systems for several language pairs, we could pool the rules extracted from all parallel corpora for computing target-syntactic features. For example, adding the English phrase dependency rules from the DE→EN corpus could improve performance of our ZH→EN and UR→EN systems. Moving beyond translation, we could use the pool of extracted rules from all systems (and using all parsers) to build *monolingual* phrase dependency parsers for use in other applications (Wu et al., 2009).

5.4 Decoding Speed

So far we have reported BLEU scores for various feature sets and parsers, but we have not discussed decoding speed. If BLEU improvements come at the cost of slowing down translation

substantially, it may not be worth it. In this section we report decoding speeds and BLEU scores for UR→EN translation as pruning thresholds are varied. Our lattice dependency parsing decoding algorithm is exact, but it prunes the search space deterministically based on source- and target-side limits on dependency lengths, as discussed in Section 4.5.2.

In this section, we vary these limits and report BLEU scores and decoding speeds of the resulting algorithm. We find that we can set these limits to be relatively strict and get similar BLEU scores in less time. In all previous experiments in this chapter, we used a source-side limit $\omega_{\boldsymbol{x}}$ of 15 and a target-side limit $\omega_{\boldsymbol{y}}$ of 20. That is, all target-side phrase dependencies may cover a maximum of 20 words in the target sentence, and the number of words between the aligned source phrases can be at most 15.

For timing experiments, we ran a single decoding thread on a Sun Fire X2200 M2 x64 server with two 2.6 GHz dual-core CPUs. Decoding during *learning* is time-consuming, since we generate unique 150-best lists for each iteration, so we only use two max dependency length settings for training. But given trained models, finding the 1-best output on the test data is much faster. It also uses less memory and therefore is easily parallelizable. So we experimented with more pruning settings for decoding. Table 5.10 shows our results. The upper table reminds us of the baseline BLEU scores. The lower table shows what happens when we train with two pruning settings: $(\omega_{\boldsymbol{x}} = 10, \omega_{\boldsymbol{y}} = 15)$ and $(\omega_{\boldsymbol{x}} = 15, \omega_{\boldsymbol{y}} = 20)$, and test with many others.

We see that a better BLEU score is reached during learning when using the more liberal pruning thresholds (26.31 vs. 26.14), but the stricter thresholds actually lead to higher test set BLEU scores, even better than those reported in Table 5.8.

The times reported only include the time required for running the Floyd-Warshall algorithm on the lattice and performing lattice dependency parsing. We use the Moses decoder for lattice generation; this typically takes only slightly longer than ordinary decoding, which is generally in the range of a few seconds per sentence. The average time required to run the Floyd-Warshall algorithm on the lattices is 0.83 seconds per sentence, so it begins to dominate the total time as the pruning thresholds go below (5, 5).

Decoding speed at test time is much faster than during learning because outputting the 1-best translation is faster and uses less memory than returning a unique 150-best list. During learning, the model with max lengths (10,15) took 12.93 seconds per sentence on average. The model with (15,20) took 19.36 seconds per sentence. These numbers are wall-time durations per sentence when using 4 decoding threads on a single 4-processor machine with 16GB of memory. When decoding certain sequences of longer sentences, the memory requirements exceeded 16GB, causing paging and slowing down the decoder, so these numbers only give a loose upper bound on the time requirements of the decoder. This memory usage appears to be caused by the requirements of generating unique k-best lists. During test-time decoding, we ran 4 decoding threads on a machine with 4 GB of memory, and encountered no difficulties with memory usage.

5.5 Comparison to Earlier Work

The results in this chapter differ from those reported in Gimpel and Smith (2011a) and other venues, due to the use of a different decoding algorithm, different features, and a difference in language model order. In prior work we used trigram language models for the baseline Moses system and for lattice generation for our system, and found larger BLEU improvements; in particular, we found an average improvement of 1 BLEU point across the three ZH→EN test sets, and only 0.0-0.1 decrease in BLEU when using unsupervised parsers in place of supervised.

We believe there are two reasons for the difference in performance when using trigram lattices in prior work vs. fivegram lattices as in this chapter. Clearly, one reason is that the benefits of some of our model's features, particularly the target-syntactic features, overlaps with benefits obtained by increasing the N-gram order in the language model.

But we believe another reason relates to lattice generation. Phrase lattices are constructed so that all features are local to lattice edges, so states will be split based on word histories needed to compute language model features (Section 2.2.1). When increasing the order of the language model, far more state-splitting appears in the phrase lattices induced by the search. This leads to less diversity of translations in the phrase lattice for a fixed lattice size. Since we prune lattices to contain fewer than 2000 lattice edges, we would expect the trigram lattices to have more diversity in translations. That is, the number of derivations in the two lattices is expected to be roughly equivalent, but the number of unique translations is expected to be different. One indication of this is to compute lattice oracle BLEU scores; indeed, even though the model-best lattice paths give worse BLEU scores in a trigram phrase lattice, the oracle paths have *higher* BLEU scores than a fivegram lattice.

So one reason for our larger improvement in BLEU when using trigram lattices could have to do with the enlarged space of translations that is contained in the phrase lattices. One way to increase the diversity in our phrase lattices would be to generate phrase lattices with a low-order language model, like a unigram or bigram, then incorporate a higher-order language model during lattice dependency parsing by using cube pruning (Chiang, 2007). Since we already use k-best decoding for our training algorithms, cube pruning is easy to implement. We could also forgo the use of Moses to generate phrase lattices and instead build a decoder that uses phrase dependency rules directly to guide the search.

5.6 Conclusion

In this chapter, we presented the results of experiments using Model P on seven test sets across four language pairs, including tasks in which we do not have supervised parsers.

The BLEU score improvements reported in this chapter are admittedly modest. However, they persist across language pairs and manual inspection reveals improvement in the translation of verbs, an important component in preserving the meaning of the source text. In addition, we found that including features that consider source syntax consistently improves performance over models that use no syntax or that only use target syntax.

We found that we can use our unsupervised Chinese parser (presented in Appendix B) in place of a supervised one and achieve the same average BLEU score across test sets. When we instead use a random Chinese parser, we see a clear drop in performance. We used fully-unsupervised part-of-speech tagging followed by our unsupervised grammar induction technique to induce dependency parsers for Urdu and Malagasy; this enabled additional features in Model P, leading to promising preliminary results.

These results point to future experimentation both in model simulation through rich feature sets and in the use of unsupervised parsing to apply syntax-based machine translation for a large number of language pairs.

	tune %BLEU	test 1 %BLEU	test 2 %BLEU	avg. test %BLEU
Moses, stack size = 500	24.72	24.77	24.89	24.83

	tune (ω_x, ω_y)	tune %BLEU	test (ω_x, ω_y)	test 1 %BLEU	test 2 %BLEU	avg. test %BLEU	time (sec./sent.)
Model P	(10, 15)	26.14	(3, 3)	22.12	22.86	22.49	1.14
			(3, 5)	23.36	23.86	23.61	1.31
			(5, 5)	24.65	25.27	24.96	1.45
			(5, 10)	24.88	25.38	25.13	2.12
			(10, 10)	25.03	**25.63**	25.33	3.04
			(10, 15)	**25.12**	25.56	**25.34**	4.07
			(15, 20)	25.11	25.57	**25.34**	6.25
			(20, 20)	25.10	25.58	**25.34**	7.38
			(5, 20)	24.87	25.42	25.15	2.49
			(20, 5)	24.92	25.34	25.13	1.69
	(15, 20)	26.31	(3, 3)	22.06	22.98	22.52	1.14
			(3, 5)	22.99	23.94	23.47	1.30
			(5, 5)	24.62	25.03	24.83	1.44
			(5, 10)	24.50	25.11	24.81	2.13
			(10, 10)	24.94	25.38	25.16	3.01
			(10, 15)	24.90	25.37	25.14	4.00
			(15, 20)	24.96	25.37	25.17	6.18
			(20, 20)	**25.01**	**25.39**	**25.20**	7.33
			(5, 20)	24.55	25.11	24.83	2.51
			(20, 5)	24.95	25.24	25.10	1.70

Table 5.10: %BLEU on tune and test sets for UR→EN translation, comparing several settings for maximum dependency lengths in the decoder. The upper table shows Moses BLEU scores for comparison. The lower table compares two max dependency length settings during training, and four for decoding on test data, showing both BLEU scores and average decoding times per sentence. See text for discussion.

Chapter 6

Conclusion

We conclude this thesis by summarizing our key contributions and discussing some avenues of future research.

6.1 Summary of Contributions

In this thesis we made the following contributions to statistical machine translation:

- We developed novel loss functions and learning algorithms for tuning the feature weights of machine translation systems (Chapter 3). Our approach captures intuitions from several successful MT learning algorithms (Liang et al., 2006a; Watanabe et al., 2007; Chiang et al., 2008b, 2009) within conceptually-simple, easy-to-implement algorithms. Experiments showed performance competitive with the state-of-the-art in small-feature settings. With large feature sets, our algorithms substantially outperformed PRO (Hopkins and May, 2011).

- We built the first machine translation system based on quasi-synchronous grammar (Chapter 4). The underlying model uses a novel formalism—**quasi-synchronous phrase dependency grammar**—that brings together two strands of research in syntax-based translation. First, it addresses syntactic divergence in tree-to-tree translation using soft constraints instantiated through a large number of features (Chiang et al., 2010). Second, it combines structural units of phrase-based and syntax-based translation in a single model, generalizing and subsuming standard phrase-based models. We developed a decoder based on **lattice dependency parsing**.

- We conducted experiments using our systems to rescore the search space of the Moses phrase-based machine translation system (Koehn et al., 2007), achieving consistent improvements across four language pairs. Manual error analysis revealed tangible improvements in translation quality, e.g., better verb translation for German-to-English translation. We reported decoding times in the range of a few seconds per sentence, showing that our decoder's speed is comparable to other syntax-based decoders.

- We showed that *unsupervised* dependency parsing can be used effectively within a syntactic translation system, enabling the use of our system for low-resource languages like Urdu and Malagasy (Chapter 5). This result offers promise for researchers to apply syntactic translation models to the hundreds of languages for which we do not have manually-annotated corpora.

6.2 Future Work

This thesis suggests several directions for future work. In this section, we discuss possibilities for future research in learning, modeling, and unsupervised parsing, citing related work where applicable.

6.2.1 Learning in Machine Translation

While we have analyzed loss functions optimized given a set of k-best lists, we still use the standard procedure of generating new k-best lists and merging them with those of previous iterations. It is unclear how this affects the loss function optimized by the learning procedure as a whole. Certain researchers have incorporated broader exploration of the search space with learning algorithms (Macherey et al., 2008; Kumar et al., 2009; Chiang et al., 2009), which often offers increased stability during learning. Using k-best lists for learning permits us to use cost functions that do not factor across the edges in a lattice or hypergraph, but there exist decomposable metrics that do factor in this way (Chiang et al., 2008a). Future work could analyze the impact of k-best learning on the loss function optimized, and conduct a thorough empirical investigation of training using the entire search space instead of k-best lists.

6.2.2 Model Extensions

One limitation of our experiments is that our model's output space is limited to the search space explored by Moses. Since phrase-based systems like Moses use pruning and a distortion limit, we may be losing some long-distance reorderings. Hierarchical phrase-based and many syntax-based systems use a different search algorithm based on synchronous parsing. We conjecture that using our model to rescore a larger search space like that of a hierarchical phrase-based model may lead to further improvements. One way to test this hypothesis would be to convert a hypergraph from a hierarchical phrase-based model into a phrase lattice, and perform lattice dependency parsing as we do currently.

We can also generate the lattices using different criteria. Current lattice generation is done as a by-product of Moses' beam search, so the lattices will contain a set of translations that score highly using the Moses feature set. However, since we then rescore the lattice paths using a richer derivation structure and a larger feature set, we may actually want to generate lattices that encode a more **diverse** set of translations, rather than simply those that are favored by Moses.

There are several ways we could attempt to do this. Batra et al. (2012) presented a method of finding a diverse set of outputs for structured prediction tasks. Their focus was on generating a

list rather than a lattice, but we could use their method while generating phrase lattices for each of k diverse translations under the model. Then we could compute the union of the lattices and use the result when lattice parsing with Model P.

Diversity is also related to **maximal marginal relevance** (Carbonell and Goldstein, 1998), originally defined for information retrieval. The idea is to return results for a search query that are simultaneously relevant and complementary to one another. This idea is also key to sentence selection in automatic summarization (Mani, 2001). Kulesza and Taskar (2010) developed structured **determinantal point processes**, which naturally model diversity in output sets, and used them effectively for the summarization task.

Alternatively, we can build a new decoding pipeline that bypasses Moses and uses our phrase dependencies to direct the exploration of the search space. Our success in rescoring the Moses search space using our model gives us hope that building a new decoding pipeline tailored to our rules would be a worthwhile endeavor.

Aside from expanding the search space, we expect further feature exploration to yield performance gains. Many supervised dependency parsers are capable of outputting labels on the dependency arcs. We can easily extract these labels when doing rule extraction and add features to our model based on them. These labels may be particularly useful to add to string-to-tree (Section 4.4.4) and tree-to-tree (Section 4.4.5) configuration features. Lexicalizing or augmenting the configurations with word class information may also help. We suspect that more data (both for rule extraction and for learning) would be necessary to get the full benefit of these features, and so we would need to scale up our rule extraction and learning pipelines to handle larger data sets. More data is also expected to strengthen our phrase dependency probability features (Sections 4.4.1 and 4.4.2), since these suffer increasingly from data sparseness as phrase lengths grow.

6.2.3 Unsupervised Grammar Induction for Machine Translation

In this thesis we took an initial step in using unsupervised parsing for machine translation by replacing supervised dependency parsers with state-of-the-art unsupervised dependency parsers. We showed promising initial results in this thesis, but there are many avenues of future research in this direction.

First, unsupervised learning of syntax can be improved if parallel text is available and we have a parser for one of the languages. In this case, the parallel text can be word-aligned and the syntactic annotations can be projected across the word alignments (Yarowsky and Ngai, 2001; Yarowsky et al., 2001). The projected parses can be improved by applying manually-written rules (Hwa et al., 2005) or modeling the noisy projection process (Ganchev et al., 2009; Smith and Eisner, 2009). For machine translation, we always have parallel text, and we often have a parser for one of the two languages, so projection is an option in these cases. If we do not have parsers for either language, grammar induction models have been developed to exploit parallel text without using any annotations on either side (Kuhn, 2004; Snyder et al., 2009).

Researchers have recently begun to target learning of parsers specifically for applications like machine translation. Hall et al. (2011) developed a framework to train supervised parsers for use in particular applications by optimizing arbitrary evaluation metrics; Katz-Brown et al. (2011)

used this framework to train a parser for reordering in machine translation. Relatedly, DeNero and Uszkoreit (2011) tailored unsupervised learning of syntactic structure in parallel text to target reordering phenomena. One possibility would be to change the unsupervised learning criterion of our parsers to maximize the BLEU score achieved by our end-to-end translation system. This may be difficult when training the target-language parser for our current model because we need to parse the target side of the parallel text, but it would be feasible for the source language parser since it is only used at translation time.

In addition, we may not need full monolingual syntactic parses to obtain the benefits of syntax-based translation modeling. Indeed, the widely-used hierarchical phrase-based model of Chiang (2005) induces a synchronous grammar from parallel text without any linguistic annotations. Zollmann and Vogel (2011) and Zollmann (2011) showed that using a supervised POS tagger to label these synchronous rules can improve performance up to the level of a model that uses a supervised full syntactic parser. They further showed that unsupervised POS taggers could be effectively used in place of supervised taggers. These results suggest that it may be fruitful to explore the use of simpler annotation tools such as POS taggers, whether supervised or unsupervised, in order to apply syntax-based translation to new language pairs.

Appendices

Appendix A

Softmax-Margin

This appendix contains additional information and proofs related to softmax-margin training of linear structured prediction models. This appendix includes material originally published as Gimpel and Smith (2010a,b). See those papers for an experimental comparison between softmax-margin and other loss functions for the task of named-entity recognition. Softmax-margin has also been shown to be effective for CCG parsing (Auli and Lopez, 2011) and congressional vote prediction (Stoyanov and Eisner, 2012), using cost functions appropriate for those tasks.

The softmax-margin loss follows (Eq. 3.16, replicated below):

$$\text{loss}_{\text{softhinge}} = -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \tag{A.1}$$

It is a generalization of a loss used by Povey et al. (2008) and similar to that used by Sha and Saul (2006). The simple intuition is the same as the intuition in max-margin learning: high-cost outputs for $\boldsymbol{x}^{(i)}$ should be penalized more heavily. Another view says that we replace the probabilistic score inside the $\exp$ function of log loss with the "cost-augmented" score from max-margin. A third view says that we replace the "hard" maximum of max-margin with the "soft-max" ($\log \sum \exp$) from log loss; hence we use the name "softmax-margin."

In the rest of this appendix, we describe some properties of the softmax-margin loss.

A.1 Convexity

Like log loss and hinge loss, the softmax-margin loss is convex. Convexity can be shown in several ways. One simple proof proceeds by showing softmax-margin to be the projection of log loss onto a subset of its parameters. Starting with a conditional log-linear model (i.e., Eq. 2.4) with k features, define the $(k+1)$th feature as the function $\text{cost}(\boldsymbol{y}^{(i)}, \cdot)$.[1] Define $g(\boldsymbol{\theta}_{1:k}, \theta_{k+1})$ as the log loss for this model, so g is convex in its parameters. Define the set $\mathcal{M} \triangleq \{m\}$, for some $m \in \mathbb{R}$; m

[1] While cost considers more information than the other features, this is not problematic for showing convexity.

is the fixed weight for the cost function. We make use of the fact that the function

$$g'(\boldsymbol{\theta}_{1:k}) = \inf_{\theta \in \mathcal{M}} g(\boldsymbol{\theta}_{1:k}, \theta)$$

is convex in $\boldsymbol{\theta}_{1:k}$ (Boyd and Vandenberghe, 2004). Since $\mathcal{M}$ contains only one element, we have $g'(\boldsymbol{\theta}_{1:k}) = g(\boldsymbol{\theta}_{1:k}, m)$. The function g' is convex and, by construction, $g'(\boldsymbol{\theta}_{1:k})$ equals the softmax-margin loss. □

A.2 Relation to Other Losses

We next show that softmax-margin upper bounds log loss, hinge loss, and Bayes risk. First, since the softmax function is a differentiable, convex upper bound on the max function, softmax-margin is a differentiable, convex upper bound on the hinge loss. To relate softmax-margin to log loss and risk, we first let $Z_i = \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta})\}$. We have:

$$\begin{aligned}
\text{loss}_{\text{softhinge}} &= -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \\
&= -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \log \left\{ Z_i \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} \frac{\exp\{\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta})\} \exp\{\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\}}{Z_i} \right\} \\
&= -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \log \left\{ Z_i \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}^{(i)})} \left[\exp\{\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \right] \right\} \\
&= -\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \log Z_i + \log \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}^{(i)})} \left[\exp\{\text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y})\} \right] \\
&= \text{loss}_{\log} + \text{loss}_{\text{softramp 1}}
\end{aligned}$$

So the softmax-margin loss is the sum of log loss and soft ramp 1. We showed in Section 3.3.4 that soft ramp 1 is an upper bound on Bayes risk. Therefore, since $\text{loss}_{\log} \geq 0$ and $\text{loss}_{\text{softramp 1}} \geq 0$ (assuming cost is non-negative), softmax-margin is a convex upper bound on both the log loss and the Bayes risk.

A.3 Minimum Divergence

In order to provide a probabilistic interpretation for softmax-margin, we relate it to the framework of minimum divergence learning (Jelinek, 1997), which is a generalization of the well-known maximum entropy framework (Berger et al., 1996).

The minimum divergence problem for conditional models can be posed as follows:

$$p^* = \operatorname*{argmin}_{p \in \mathbb{P}_y} \sum_{i=1}^{n} D(p(\cdot \mid \boldsymbol{x}^{(i)} \| q(\cdot \mid \boldsymbol{x}^{(i)})) \tag{A.2}$$

$$\text{subject to} \sum_{i=1}^{n} \mathbb{E}_{p(\cdot|\boldsymbol{x}^{(i)})}[\boldsymbol{f}(\boldsymbol{x}^{(i)}, \cdot)] = \sum_{i=1}^{n} \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \tag{A.3}$$

$$\sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} p(\boldsymbol{y} \mid \boldsymbol{x}^{(i)}) = 1, \ \forall i \tag{A.4}$$

$$p(\boldsymbol{y} \mid \boldsymbol{x}^{(i)}) \geq 0, \ \forall i, \forall \boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}} \tag{A.5}$$

where $\mathbb{P}_{\boldsymbol{y}}$ is the set of all distributions with support equal to $\mathcal{Y}_{\boldsymbol{x}}$ and $D(p\|q)$ is the Kullback-Leibler divergence between p and q. That is, the goal of minimum divergence learning is to find a distribution p^* that is as close as possible to q subject to the constraints that the feature expectations match the empirical feature values. The distribution q remains fixed and is frequently referred to as the **base distribution**. The maximum entropy problem is a special case of minimum divergence in which the base distribution is uniform. Maximum entropy modeling has been used extensively in NLP, while applications of minimum divergence learning have been less common; examples of the latter include language modeling (Jelinek, 1997; Berger and Printz, 1998) and machine translation (Foster, 2000).

It is well-known that the dual of the maximum entropy problem is maximum likelihood for a log-linear model (Berger et al., 1996; Smith, 2011). With small change to that derivation, it can be shown that the dual of the minimum divergence problem for conditional models is maximum likelihood for the following model (Jelinek, 1997):

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{q(\boldsymbol{y} \mid \boldsymbol{x}) \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\}}{\sum_{\boldsymbol{y}' \in \mathcal{Y}_{\boldsymbol{x}}} q(\boldsymbol{y}' \mid \boldsymbol{x}) \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}')\}} \tag{A.6}$$

The maximum likelihood problem for this model (and the dual of the minimum divergence problem) is

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} -\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) + \log \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(i)}}} q(\boldsymbol{y} \mid \boldsymbol{x}^{(i)}) \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\} \tag{A.7}$$

Note that Eq. A.7 does not contain the base distribution $q(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)})$ in the numerator term, because it separates out via the $\log$ function and can be removed because it does not depend on $\boldsymbol{\theta}$.

The softmax-margin approach can be seen as minimum divergence with a base distribution q' that depends on the true output $\boldsymbol{y}^*$ in addition to $\boldsymbol{x}$:

$$q'(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{y}^*) = \frac{\exp\{\text{cost}(\boldsymbol{y}^*, \boldsymbol{y})\}}{\sum_{\boldsymbol{y}' \in \mathcal{Y}_{\boldsymbol{x}}} \exp\{\text{cost}(\boldsymbol{y}^*, \boldsymbol{y}')\}} \tag{A.8}$$

Note that this base distribution is not useful for classification since it favors outputs $\boldsymbol{y}$ that have high cost. Ordinarily, when a model learned with minimum divergence is used for prediction, the output structure is chosen according to Eq. A.6. However, when training with softmax-margin, prediction simply chooses the output structure according to $\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\}$, ignoring q.

Typically, minimum divergence is chosen over maximum entropy when a base distribution is available that serves as a useful starting point for the modeling task of interest. For example, when building a domain-specific language model, one might wish to minimize divergence to a broad-domain language model while satisfying certain constraints obtained from the domain-specific data (Jelinek, 1997). Our motivation for softmax-margin training departs from this notion. The base distribution *inflates* the model scores of high-cost output structures, while the training procedure aims to *decrease* their scores. In this case, the base distribution acts as a handicap, forcing the training procedure to work extra hard to penalize high-cost outputs. Once the base distribution is removed at prediction time, the remaining score (i.e., $\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\}$) is strongly biased away from these high-cost outputs. Broadly, the intuition is similar to that of max-margin training for structured prediction, while here the margin constraints are captured through the use of a particular base distribution in the minimum divergence framework.

Appendix B

Concavity and Initialization for Unsupervised Dependency Parsing

In this appendix, we present our contributions to unsupervised dependency grammar induction that are relevant to this thesis. Grammar induction—and unsupervised in NLP more generally—has been plagued by non-concavity of the likelihood function maximized by EM. We introduce an initializer for dependency grammar induction that has a concave log-likelihood function. We use it to initialize the dependency model of Klein and Manning (2004) and show improvement on average across 18 languages. We use our concave initializers to learn grammar induction systems for English, Chinese, Urdu, and Malagasy, and apply them to translation in Chapter 5.

The rest of this appendix is laid out as follows. Section B.1 gives background and motivates the approach. In Section B.2, we begin our investigation of unsupervised models with concave log-likelihood functions, of which IBM Model 1 (Brown et al., 1993) is the most well-known example. We show why other models are so seldom concave. In Section B.3 we present concave models for dependency grammar induction and validate them experimentally in Section B.4.

This appendix contains material originally published as Gimpel and Smith (2012b). In other work not included here, we developed richer models for grammar induction using logistic-normal priors and Bayesian inference (Cohen et al., 2008).

B.1 Background and Motivation

In natural language processing, unsupervised learning typically implies optimization of a "bumpy" objective function riddled with local maxima. However, one exception is IBM Model 1 (Brown et al., 1993) for word alignment, which is the only model commonly used for unsupervised learning in NLP that has a concave log-likelihood function.[1] For other models, such as those used in unsupervised part-of-speech tagging and grammar induction, and indeed for more sophisticated word alignment models, the log-likelihood function maximized by EM is non-concave.

[1] However, it is not *strictly* concave (Toutanova and Galley, 2011).

As a result, researchers are obligated to consider initialization in addition to model design (Klein and Manning, 2004; Goldberg et al., 2008).

For example, consider the dependency grammar induction results shown in Table B.1 when training the widely used dependency model with valence (DMV; Klein and Manning, 2004). Using uniform distributions for initialization (UNIF) results in an accuracy of 17.6% on the test set, well below the baseline of attaching each word to its right neighbor (ATTACHRIGHT, 31.7%). Furthermore, when using a set of 50 random initializers (RAND), the standard deviation of the accuracy is an alarming 8.3%.

In light of this sensitivity to initialization, it is compelling to consider unsupervised models with concave log-likelihood functions, which may provide stable, data-supported initializers for more complex models. In the sections that follow, we explore the issues involved with such an expedition and elucidate the limitations of such models for unsupervised NLP. We then present simple concave models for dependency grammar induction that are easy to implement and offer efficient optimization. We also show how linguistic knowledge can be encoded without sacrificing concavity.

Using our models to initialize the DMV, we find that they lead to an improvement in average accuracy across 18 languages. In the experiments in Chapter 5, we replace supervised parsers for English and Chinese with the best unsupervised models we will present. We also apply the same technique to learn unsupervised parsers for Urdu and Malagasy, which we also use in the experiments in Chapter 5.

B.2 IBM Model 1 and Concavity

IBM Model 1 is a conditional model of a target-language sentence $\boldsymbol{y}$ of length m and an alignment $\boldsymbol{a}$ given a source-language sentence $\boldsymbol{x}$ of length n. The generation of m is assumed to occur with some (inconsequential) uniform probability ϵ. The alignment vector $\boldsymbol{c}$, a hidden variable, has an entry for each element of $\boldsymbol{y}$ that contains the index in $\boldsymbol{x}$ of the aligned word. These entries are used to define which translation parameters $t(y_j \mid x_{c_j})$ are active. Model 1 assumes that the probability of the ith element in $\boldsymbol{c}$, denoted $c(i \mid j, n, m)$, is simply a uniform distribution over all n source words plus the null word. These assumptions result in the following log-likelihood for a sentence pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ under Model 1 (marginalizing $\boldsymbol{c}$):

$$\log p(\boldsymbol{y} \mid \boldsymbol{x}) = \log \frac{\epsilon}{(n+1)^m} + \sum_{j=1}^{m} \log \sum_{i=0}^{n} t(y_j \mid x_i) \tag{B.1}$$

The only parameters to be learned in the model are $\boldsymbol{t} = \{t(y \mid x)\}_{y,x}$. Since a parameter is concave in itself, the sum of concave functions is concave, and the log of a concave function is concave, Eq. B.1 is concave in $\boldsymbol{t}$ (Brown et al., 1993; Toutanova and Galley, 2011).

IBM Model 2 involves a slight change to Model 1 in which the probability of a word link depends on the word positions. However, this change renders it no longer concave. Consider the

log-likelihood function for Model 2:

$$\log \epsilon + \sum_{j=1}^{m} \log \sum_{i=0}^{n} t(y_j \mid x_i) \cdot c(i \mid j, n, m) \tag{B.2}$$

Eq. B.2 is not concave in the parameters $t(y_j \mid x_i)$ and $c(i \mid j, n, m)$ because a product is neither convex nor concave in its vector of operands. This can be shown by computing the Hessian matrix of $f(a, b) = ab$ and showing that it is indefinite.

In general, concavity is lost when the log-likelihood function contains a product of model parameters enclosed within a $\log \sum$. If the sum is not present, the $\log$ can be used to separate the product of parameters, making the function concave. It can also easily be shown that a "featurized" version (Berg-Kirkpatrick et al., 2010) of Model 1 is not concave. More generally, any non-concave function enclosed within $\log \sum$ will cause the log-likelihood function to be non-concave, though there are few other non-concave functions with a probabilistic semantics than those just discussed.

B.3 Concave, Unsupervised Models

Nearly every other model used for unsupervised learning in NLP has a non-concave log-likelihood function. We now proceed to describe the conditions necessary to develop concave models for two tasks.

B.3.1 Part-of-Speech Tagging

Consider a standard first-order hidden Markov model for POS tagging. Letting $\boldsymbol{z}$ denote the tag sequence for a sentence $\boldsymbol{y}$ with m tokens, the single-example log-likelihood is:

$$\log \sum_{\boldsymbol{z}} p(\text{stop} \mid z_m) \prod_{j=1}^{m} p(z_j \mid z_{j-1}) \cdot p(y_j \mid z_j) \tag{B.3}$$

where z_0 is a designated "start" symbol. Unlike IBM Models 1 and 2, we cannot reverse the order of the summation and product here because the transition parameters $p(z_j \mid z_{j-1})$ cause each tag decision to affect its neighbors. Therefore, Eq. B.3 is non-concave due to the presence of a product within a $\log \sum$.

However, if the tag transition probabilities $p(z_j \mid z_{j-1})$ are all constants and also do not depend on the previous tag z_{j-1}, then we can rewrite Eq. B.3 as the following concave log-likelihood function (using $C(z)$ to denote a constant function of tag z, e.g., a fixed tag prior distribution):

$$\log C(\text{stop}) + \log \prod_{j=1}^{m} \sum_{z_j} C(z_j) \cdot p(y_j \mid z_j)$$

Lacking any transition modeling power, this model appears weak for POS tagging. However, we note that we can add additional conditioning information to the $p(y_j \mid z_j)$ distributions and

retain concavity, such as nearby words and tag dictionary information. We speculate that such a model might learn useful patterns about local contexts and provide an initializer for unsupervised part-of-speech tagging. Convex models for clustering (Lashkari and Golland, 2008) might serve as useful initializers as well.

B.3.2 Dependency Grammar Induction

To develop dependency grammar induction models, we begin with a version of Model 1 in which a sentence $\boldsymbol{y}$ is generated from a copy of itself (denoted $\boldsymbol{y}'$):

$$\log p(\boldsymbol{y} \mid \boldsymbol{y}') = \log \frac{\epsilon}{(m+1)^m} + \sum_{j=1}^{m} \log \sum_{i=0, i \neq j}^{m} c(y_j \mid y'_i) \tag{B.4}$$

If a word y_j is "aligned" to y'_0, y_j is a root. This is a simple child-generation model with no tree constraint. In order to preserve concavity, we are forbidden from conditioning on other parent-child assignments or including any sort of larger constraints.

However, we can condition the child distributions on additional information about $\boldsymbol{y}'$ since it is fully observed. This conditioning information may include the direction of the edge, its distance, and any properties about the words in the sentence. We found that conditioning on direction improved performance: we rewrite the c distributions as $c(y_j \mid y'_i, \text{sign}(j-i))$ and denote this model by CCV1.

We note that we can also include constraints in the sum over possible parents and still preserve concavity. Naseem et al. (2010) found that adding parent-child constraints to a grammar induction system can improve performance dramatically. We employ one simple rule: roots are likely to be verbs.[2] We modify CCV1 to restrict the summation over parents to exclude y'_0 if the child word is not a verb.[3] We only employ this restriction during EM learning for sentences containing at least one verb. For sentences without verbs, we allow all words to be the root. We denote this model by CCV2.

In related work, Brody (2010) also developed grammar induction models based on the IBM word alignment models. However, while our goal is to develop concave models, Brody employed Bayesian nonparametrics in his version of Model 1, which makes the model non-concave.

B.4 Experiments

We ran experiments to determine how well our concave grammar induction models CCV1 and CCV2 can perform on their own and when used as initializers for the DMV (Klein and Manning, 2004). The DMV is a generative model of POS tag sequences and projective dependency trees

[2] This is similar to the rule used by Mareček and Žabokrtský (2011) with empirical success.

[3] As verbs, we take all tags that map to V in the universal tag mappings from Petrov et al. (2012). Thus, to apply this constraint to a new language, one would have to produce a similar tag mapping or identify verb tags through manual inspection.

		Train ≤ 10		Train ≤ 20	
		Test		Test	
Model	Init.	≤ 10	$\leq \infty$	≤ 10	$\leq \infty$
ATTACHRIGHT	N/A	38.4	31.7	38.4	31.7
CCV1	UNIF	31.4	25.6	31.0	23.7
CCV2	UNIF	43.1	28.6	43.9	27.1
DMV	UNIF	21.3	17.6	21.3	16.4
	RAND*	41.0	31.8	-	-
	K&M	44.1	32.9	51.9	37.8
	CCV1	45.3	30.9	53.9	36.7
	CCV2	54.3	43.0	**64.3**	**53.1**
Shared LN	K&M	61.3	41.4		
L-EVG	RAND†	**68.8**	-		
Feature DMV	K&M	63.0	-		
LexTSG-DMV	K&M	67.7	**55.7**		
Posterior Reg.	K&M	64.3	53.3		
Punc/UnsupTags	K&M′			-	59.1‡

Table B.1: English attachment accuracies on Section 23, for short sentences (≤ 10 words) and all ($\leq \infty$). We include selected results on this same test set: Shared LN = Cohen and Smith (2009), L-EVG = Headden III et al. (2009), Feature DMV = Berg-Kirkpatrick et al. (2010), LexTSG-DMV = Blunsom and Cohn (2010), Posterior Reg. = Gillenwater et al. (2010), Punc/UnsupTags = Spitkovsky et al. (2011a). K&M′ is from Spitkovsky et al. (2011b). *Accuracies are averages over 50 random initializers; $\sigma = 10.9$ for test sentences ≤ 10 and 8.3 for all. †Used many random initializers with unsupervised run selection. ‡Used staged training with sentences ≤ 45 words.

over them. It is the foundation of most state-of-the-art unsupervised grammar induction models (several of which are listed in Tab. B.1). The model includes multinomial distributions for generating each POS tag given its parent and the direction of generation: where y_e is the parent POS tag and y_e the child tag, these distributions take the form $c(y_j \mid y_i, \text{sign}(j-i))$, analogous to the distributions used in our concave models. The DMV also has multinomial distributions for deciding whether to stop or continue generating children in each direction considering whether any children have already been generated in that direction.

The majority of researchers use the original initializer from Klein and Manning (2004), denoted here K&M. K&M is a deterministic harmonic initializer that sets parent-child token affinities inversely proportional to their distances, then normalizes to obtain probability distributions. K&M is often described as corresponding to an initial E step for an unspecified model that favors short attachments.

Init.	eu	bg	ca	zh	cs	da	nl	en	de	el	hu
UNIF	24/21	32/26	27/29	44/40	32/30	24/19	21/21	21/18	31/24	37/32	23/18
K&M	**32/26**	**48/40**	24/25	38/33	31/29	**34/23**	39/33	44/33	47/**37**	50/41	23/20
CCV1	22/21	34/27	**44/51**	**46/45**	33/31	19/14	24/24	45/31	46/31	**51/45**	32/28
CCV2	26/25	34/26	29/35	**46**/44	**50/40**	29/18	**50/43**	**54/43**	**49**/33	50/**45**	**60/46**
	it	ja	pt	sl	es	sv	tr	avg. accuracy		avg. log-likelihood	
UNIF	31/24	35/30	49/36	20/20	29/24	26/22	33/30	29.8 / 25.7		-15.05	
K&M	32/24	39/**31**	44/28	**33/27**	19/11	**46/33**	**39/36**	36.7 / 29.4		-14.84	
CCV1	34/25	42/27	**50/38**	30/25	41/33	45/**33**	37/29	37.5 / 30.9		-14.93	
CCV2	**55/48**	**49/31**	**50/38**	22/21	**57/50**	**46**/32	31/22	**43.7 / 35.5**		**-14.45**	

Table B.2: Test set attachment accuracies for 18 languages; first number in each cell is accuracy for sentences ≤ 10 words and second is for all sentences. For training, sentences ≤ 10 words from each treebank were used. In order, languages are Basque, Bulgarian, Catalan, Chinese, Czech, Danish, Dutch, English, German, Greek, Hungarian, Italian, Japanese, Portuguese, Slovenian, Spanish, Swedish, and Turkish.

Procedure We run EM for our concave models for 100 iterations. We evaluate the learned models directly as parsers on the test data and also use them to initialize the DMV. When using them directly as parsers, we use dynamic programming to ensure that a valid tree is recovered. When using the concave models as initializers for the DMV, we copy the c parameters over directly since they appear in both models. We do not have the stop/continue parameters in our concave models, so we simply initialize them uniformly for the DMV. We train each DMV for 200 iterations and use minimum Bayes risk decoding with the final model on the test data. We use several initializers for training the DMV, including the uniform initializer (UNIF), K&M, and our trained concave models CCV1 and CCV2.

Data We use data prepared for the CoNLL 2006/07 shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007).[4] We follow standard practice in removing punctuation and using short sentences (≤ 10 or ≤ 20 words) for training. For all experiments, we train on separate data from that used for testing and use gold POS tags for both training and testing. We report accuracy on (i) test set sentences ≤ 10 words and (ii) all sentences from the test set.

Results Results for English are shown in Table B.1. We train on §2–21 and test on §23 in the Penn Treebank. The constraint on sentence roots helps a great deal, as CCV2 by itself is competitive with the DMV when testing on short sentences. The true benefit of the concave models, however,

[4] In some cases, we did not use official CoNLL test sets but instead took the training data and reserved the first 80% of the sentences for training, the next 10% for development, and the final 10% as our test set; dataset details are the same as those given by Cohen (2011).

appears when using them as initializers. The DMV initialized with CCV2 achieves a substantial improvement over all others. When training on sentences of length ≤ 20 words (bold), the performance even rivals that of several more sophisticated models shown in the table, despite only using the DMV with a different initializer.

Table B.2 shows results for 18 languages. On average, CCV2 performs best and CCV1 does at least as well as K&M. This shows that a simple, concave model can be as effective as a state-of-the-art hand-designed initializer (K&M), and that concave models can encode linguistic knowledge to further improve performance.

Average log-likelihoods (micro-averaged across sentences) achieved by EM training are shown in the final column of Table B.2. CCV2 leads to substantially-higher likelihoods than the other initializers, suggesting that the verb-root constraint is helping EM to find better local optima.[5]

B.5 Discussion and Relation to Thesis

We have investigated concave models for unsupervised learning, elucidated the properties necessary to achieve concavity, and developed concave models for dependency grammar induction. Staged training has been shown to help unsupervised learning in the past, from early work in grammar induction (Lari and Young, 1990) and word alignment (Brown et al., 1993) to more recent work in dependency grammar induction (Spitkovsky et al., 2010). While we do not yet offer a generic procedure for extracting a concave approximation from any model for unsupervised learning, our results contribute evidence in favor of the general methodology of staged training in unsupervised learning, and provide a simple and powerful initialization method for dependency grammar induction.

For the English and Chinese unsupervised dependency parsers used for the experiments in Chapter 5, we used the parsers trained here, using gold part-of-speech tags and CCV2 for initialization of the DMV. For Urdu and Malagasy, we first obtained unsupervised part-of-speech tags by running the fully-unsupervised induction model from Berg-Kirkpatrick et al. (2010), using 40 tags. We then used CCV1 to initialize training for the DMV on the tagged corpus.

[5] However, while CCV1 leads to a higher average accuracy than K&M, the latter reaches slightly higher likelihood, suggesting that the success of the concave initializers is only partially due to reaching high training likelihood.

Appendix C

Data and Experimental Details

In this appendix, we include details of data used in experiments and experimental settings used for training baseline systems. Details of which data was used for various tasks (translation model estimation, language model estimation, tuning, Brown cluster estimation) are also provided.

C.1 Language Pairs

In this thesis, we consider five language pairs, three for which large amounts of parallel data are available and two involving low-resource languages. For large-data experiments, we consider Arabic→English (AR→EN), Chinese→English (ZH→EN), and German→English (DE→EN). Experiments with low-resource language pairs include Urdu→English (UR→EN) and English→Malagasy (EN→MG) translation. All English data that was parsed was done so using TurboParser (Martins et al., 2009).

The following sections describe the procedures used to prepare the data for each language pair. The line and token counts are summarized in Tables C.1 and C.3.

Arabic→English For AR→EN, we used data provided by the LDC for the NIST evaluations, including 3,286,205 sentence pairs of UN data and 982,088 sentence pairs of non-UN data. The Arabic data was preprocessed using an HMM segmenter that splits off attached prepositional phrases, personal pronouns, and the future marker (Lee et al., 2003). The common stylistic sentence-initial *wa#* (*and ...*) was removed from the training and test data. The resulting corpus contained 130M Arabic tokens and 130M English tokens.

Phrase-based models were trained using the full parallel corpus. For hierarchical phrase-based models, rules were only extracted from the non-UN parallel data. This was necessary due to computational constraints stemming from the large number of extracted rules for hierarchical phrase-based translation. Also, the "MaxSpan" rule extraction parameter was reduced to 10 from its default value of 15. This parameter limits the span size (in the original sentence) of extracted rules. The full parallel corpus was used for all other aspects of training hierarchical models, including word alignment, computation of lexical weighting features, and language modeling. As tune/test

	lines	source tokens	target tokens
AR→EN phrase-based	4,268,293	130,337,729	130,454,385
AR→EN hierarchical	982,088	22,861,985	23,738,300
ZH→EN	302,996	7,984,637	9,350,506
DE→EN	1,010,466	23,154,642	24,044,528
UR→EN	165,159	1,169,367	1,083,807
EN→MG	83,670	1,500,922	1,686,022

Table C.1: Statistics of data used for rule extraction and feature computation.

	tune		
	lines	source tokens	target tokens
AR→EN	1,263	24,245	27,801
ZH→EN	919	24,152	28,870
DE→EN	1,300	29,791	31,318
UR→EN	882	18,004	16,606
EN→MG	1,359	28,408	32,682

Table C.2: Statistics of data used for learning (often called "tuning"). The numbers of target tokens are averages across four reference translations for AR→EN, ZH→EN, and UR→EN, rounded to the nearest token.

sets, we used a subset of MT06 for tuning ("tune"), a subset of MT05 for one test set ("test 1"), the MT08 newswire test set ("test 2"), and the MT08 weblog test set ("test 3").

Chinese→English For ZH→EN, we used 303k sentence pairs from the FBIS corpus (LDC2003E14). We segmented the Chinese data using the Stanford Chinese segmenter (Chang et al., 2008) in "CTB" mode, giving us 7.9M Chinese tokens and 9.4M English tokens. For tuning and testing, we used MT03 ("tune"), MT02 ("test 1"), MT05 ("test 2"), and MT06 ("test 3"). The Chinese text was parsed using the Stanford parser (Levy and Manning, 2003).

German→English We took the WMT12 aligned Europarl corpus, tokenized, filtered sentences with more than 50 words, and downcased the text. We then discarded every other sentence, beginning with the second, leaving half of the corpus remaining. We did this to speed our experiment cycle. The corpus still has about 850k sentence pairs. We did the same processing with the news commentary corpus, but did not discard half of the sentences the way we did with the Europarl

	test 1			test 2			test 3		
	lines	source tokens	target tokens	lines	source tokens	target tokens	lines	source tokens	target tokens
AR→EN	764	18,275	20,327	813	25,500	29,281	547	19,160	22,487
ZH→EN	878	22,708	26,877	1,082	29,956	35,227	1,664	38,787	48,169
DE→EN	2,525	62,699	65,595	2,489	61,263	61,924	3,003	72,660	74,753
UR→EN	883	21,659	19,575	1,792	42,082	39,889		N/A	
EN→MG	1,133	24,362	28,301		N/A			N/A	

Table C.3: Test data statistics. The numbers of target tokens are averages across four reference translations for AR→EN, ZH→EN, and UR→EN, rounded to the nearest token.

corpus. There were about 150k news commentary sentences, giving us a total of about 1M lines of DE→EN parallel training data. For tuning, we used the first 1300 sentences from the 2008 2051-sentence test set ("tune"). We did this in order to speed up our experiment cycle. For testing, we used the 2009 test set ("test 1"), the 2010 test set ("test 2"), and the 2011 test set ("test 3"). All tuning/test sets are from the newswire domain. The German text was parsed using the factored model in the Stanford parser (Rafferty and Manning, 2008).

Urdu→English For UR→EN, we used parallel data from the NIST MT08 evaluation. While there are 165,159 lines of parallel data, there are many dictionary and otherwise short entries, so it is close to an order of magnitude smaller than ZH→EN. We used half of the documents (882 sentences) from the MT08 test set for tuning ("tune"). We used the remaining half for one test set ("test 1") and MT09 as our other test set ("test 2"). We only used two test sets for UR→EN experiments. The Urdu text was parsed using an unsupervised dependency parsing pipeline as described in Chapter 5.

English→Malagasy For EN→MG translation, we used data obtained from the Global Voices weblogging community (`http://globalvoicesonline.org`), prepared by Victor Chahuneau.[1] We used release 12.06 along with its recommended training, development, and test sets. There is only one test set for this language pair. Like Urdu, the Malagasy text was parsed using the unsupervised dependency parsing pipeline described in Chapter 5.

[1]The data is publically available at `http://www.ark.cs.cmu.edu/global-voices/`.

C.2 Experimental Details

Translation Models For phrase-based models, we used the Moses machine translation toolkit (Koehn et al., 2007). We mostly used default settings and features, including the default lexicalized reordering model. Word alignment was performed using GIZA++ (Och and Ney, 2003) in both directions, the `grow-diag-final-and` heuristic was used to symmetrize the alignments, and a max phrase length of 7 was used for phrase extraction. The only exception to the defaults was setting the distortion limit to 10 in all experiments. For hierarchical phrase-based models, we again used the Moses toolkit (Hoang et al., 2009) with default features and settings except for changing the `max-chart-span` parameter to 10.

Language Models Language models were trained using the target side of the parallel corpus in each case augmented with 24,760,743 lines (601,052,087 tokens) of randomly-selected sentences from the Gigaword v4 corpus (excluding the New York Times and Los Angeles Times). The minimum count cut-off for unigrams, bigrams, and trigrams was one and the cut-off for four-grams and five-grams was three. Language models were estimated using the SRI Language Modeling toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998). Language model inference was performed using KenLM (Heafield, 2011) within Moses.

For EN→MG, we estimated a 5-gram language model using only the target side of the parallel corpus, which contained 89,107 lines with 2,031,814 tokens. We did not use any additional Malagasy data for estimating the EN→MG language models in order to explore a scenario in which target-language text is limited or expensive to obtain.

Brown Clustering Brown clusters (Brown et al., 1992) were generated using code provided by Liang (2005). For each language pair, 100 word clusters were generated for the target language. A count cut-off was also used. This causes the algorithm to only cluster words appearing more times than the cut-off. When the clusters are used, all words with counts below the cut-off are assigned a special "unknown word" cluster. So in practice, if a clustering with 100 clusters is generated, there are 101 clusters used when the clusters are applied.

For ZH→EN, DE→EN, and UR→EN, the target side of the parallel data was used along with 412,000 lines of randomly-selected Gigaword data comprising 10,001,839 words. This data was a subset of the Gigaword data used for language modeling. The count cut-off was 2. For EN→MG, only the target side of the parallel corpus was used. The count cut-off was 1. In all cases, the data was tokenized and downcased prior to cluster generation.

Appendix D

Notation and Definitions

i, j, k, l	integers
$\boldsymbol{x}, \boldsymbol{y}$	vectors
x_i	entry i in vector $\boldsymbol{x}$
$\boldsymbol{x}_i^j$	sequence from entry i to entry j (inclusive) in vector $\boldsymbol{x}$
$[i]$	the set containing the first i positive integers
$\mathbb{I}[P]$	indicator function; returns 1 if P evaluates to true, 0 otherwise
$\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$	source language sentence
n	length of $\boldsymbol{x}$
$\boldsymbol{y} = \langle y_1, \ldots, y_m \rangle$	target language sentence, translation of $\boldsymbol{x}$
m	length of $\boldsymbol{y}$
$\boldsymbol{h}$	latent derivation variable
$\mathcal{X}$	set of all strings in the source language
$\mathcal{Y}_{\boldsymbol{x}}$	for a particular $\boldsymbol{x} \in \mathcal{X}$, the set of its possible translations (correct and incorrect) in the target language
$\mathcal{H}_{\boldsymbol{x}}$	set of possible values of $\boldsymbol{h}$ for the input sentence $\boldsymbol{x}$
$\mathcal{T}_{\boldsymbol{x}} \subseteq \mathcal{Y}_{\boldsymbol{x}} \times \mathcal{H}_{\boldsymbol{x}}$	set of "valid" $\langle \boldsymbol{y}, \boldsymbol{h} \rangle$ pairs for $\boldsymbol{x}$, i.e., output pairs licensed by a model
$\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle$	alignments from words in $\boldsymbol{x}$ to words in $\boldsymbol{y}$. if $a_i = j$, x_i is aligned to y_j. If $a_i = 0$, x_i is aligned to NULL.
$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$	vector of feature functions on $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{h}$
$\boldsymbol{\theta}$	vector of feature weights
$\pi = \boldsymbol{x}_j^k$	source-sentence **phrase**: sequence of words in the source sentence $\boldsymbol{x}$, i.e., $1 \leq j \leq k \leq n$; the number of words in π is denoted $\lvert\pi\rvert$
$\phi = \boldsymbol{y}_j^k$	target-sentence **phrase**: sequence of words in the target sentence $\boldsymbol{y}$, i.e., $1 \leq j \leq k \leq m$; the number of words in ϕ is denoted $\lvert\phi\rvert$
$\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_{n'} \rangle$	segmentation of $\boldsymbol{x}$ into phrases such that for $i \in [n']$, $\pi_i = \boldsymbol{x}_j^k$ is a source-sentence phrase and $\pi_1 \cdot \ldots \cdot \pi_{n'} = \boldsymbol{x}$
$\boldsymbol{\phi} = \langle \phi_1, \ldots, \phi_{n'} \rangle$	segmentation of $\boldsymbol{y}$ into phrases such that for $i \in [n']$, $\phi_i = \boldsymbol{y}_j^k$ is a target-sentence phrase and $\phi_1 \cdot \ldots \cdot \phi_{n'} = \boldsymbol{y}$
$\boldsymbol{b} : \{1, \ldots, n'\} \to \{1, \ldots, n'\}$	one-to-one alignment (bijection) from phrases in $\boldsymbol{\phi}$ to phrases in $\boldsymbol{\pi}$; for all $i \in [n']$, if $\boldsymbol{b}(i) = j$, then π_j is a subvector of $\boldsymbol{x}$ and ϕ_i is a subvector of $\boldsymbol{y}$
$\tau_{\boldsymbol{x}} : \{1, \ldots, n\} \to \{0, \ldots, n\}$	dependency tree on source words $\boldsymbol{x}$, where $\tau_{\boldsymbol{x}}(i)$ is the index of the parent of word x_i (0 is the wall symbol \$)
$\tau_{\boldsymbol{\phi}} : \{1, \ldots, n'\} \to \{0, \ldots, n'\}$	dependency tree on target phrases $\boldsymbol{\phi}$, where $\tau_{\boldsymbol{\phi}}(i)$ is the index of the parent of phrase ϕ_i (0 is the wall symbol \$)

Table D.1: Key notation and definitions used in this thesis.

Bibliography

A. Abeillé, Y. Schabes, and A.K. Joshi. Using lexicalized TAGs for machine translation. In *Proc. of ACL*, 1990. [24]

H. Alshawi, S. Bangalore, and S. Douglas. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60, 2000. [27, 67]

V. Ambati and A. Lavie. Improving syntax driven translation models by re-structuring divergent and non-isomorphic parse tree structures. In *Proc. of the Eighth Conference of the Association for Machine Translation in the Americas*, 2008. [i, 2, 3, 24]

A. Arun and P. Koehn. Online learning methods for discriminative training of phrase based statistical machine translation. In *Proc. of MT Summit XI*, 2007. [i, 2, 5, 29, 47]

A. Arun, C. Dyer, B. Haddow, P. Blunsom, A. Lopez, and P. Koehn. Monte carlo inference and maximization for phrase-based translation. In *Proc. of CoNLL*, 2009. [13, 34]

A. Arun, B. Haddow, and P. Koehn. A unified approach to minimum risk training and decoding. In *Proc. of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, 2010a. [34]

A. Arun, B. Haddow, P. Koehn, A. Lopez, C. Dyer, and P. Blunsom. Monte carlo techniques for phrase-based translation. *Machine Translation*, 24(2), 2010b. [34]

M. Auli and A. Lopez. Training a log-linear parser with loss functions via softmax-margin. In *Proc. of EMNLP*, 2011. [48, 114]

N.F. Ayan and B.J. Dorr. Going beyond aer: An extensive analysis of word alignments and their impact on mt. In *Proc. of COLING-ACL*, 2006. [11]

L.R. Bahl, F. Jelinek, and R.L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2), 1983. [10]

D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich. Diverse m-best solutions in markov random fields. In *Proc. of European Conference on Computer Vision*, 2012. [109]

M. Bazrafshan, T. Chung, and D. Gildea. Tuning as linear regression. In *Proc. of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2012. [34]

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. Painless unsupervised learning with features. In *Proc. of NAACL*, 2010. [6, 101, 120, 122, 124]

A. Berger and H. Printz. A comparison of criteria for maximum entropy/minimum divergence

feature selection. In *Proc. of EMNLP*, 1998. [116]

A. Berger, V. J. Della Pietra, and S. A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996. [115, 116]

A. Birch, P. Blunsom, and M. Osborne. A quantitative analysis of reordering phenomena. In *Proc. of the Fourth Workshop on Statistical Machine Translation*, 2009. [i, 2, 4, 71]

P. Blunsom and T. Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proc. of EMNLP*, 2010. [6, 122]

P. Blunsom and M. Osborne. Probabilistic inference for machine translation. In *Proc. of EMNLP*, 2008. [2, 13, 24, 35, 48]

P. Blunsom, T. Cohn, and M. Osborne. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL*, 2008. [13, 48]

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. [29, 115]

E. Brill and M. Marcus. Automatically acquiring phrase structure using distributional analysis. In *Proc. of DARPA Workshop on Speech and Natural Language*, 1992. [ii, 3]

S. Brody. It depends on the translation: Unsupervised dependency parsing via word alignment. In *Proc. of EMNLP*, 2010. [121]

P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A statistical approach to language translation. In *Proc. of COLING*, 1988. [9]

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2): 79–85, 1990. [1, 4, 9, 12]

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based N-gram models of natural language. *Computational Linguistics*, 18, 1992. [78, 128]

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993. [11, 14, 17, 28, 29, 118, 119, 124]

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*, 2006. [ii, 2, 3, 5, 123]

J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998. [110]

J.G. Carbonell, S. Klein, D. Miller, M. Steinbaum, T. Grassiany, and J. Frey. Context-based machine translation. In *Proc. of AMTA*, 2006. [16]

X. Carreras and M. Collins. Non-projective parsing for statistical machine translation. In *Proc. of EMNLP*, 2009. [27]

F. Casacuberta and C. de la Higuera. Computational complexity of problems on probabilistic grammars and transducers. In L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications; 5th International Colloquium, ICGI 2000*, pages 15–24. Springer, 2000. [12]

D. Cer. *Parameterizing Phrase Based Statistical Machine Translation Models: An Analytic Study*. PhD thesis, Stanford University, 2011. [48]

D. Cer, D. Jurafsky, and C. Manning. Regularization and search for minimum error rate training. In *Proc. of ACL-2008 Workshop on Statistical Machine Translation*, 2008. [33]

P. Chang, M. Galley, and C. Manning. Optimizing Chinese word segmentation for machine translation performance. In *Proc. of ACL-2008 Workshop on Statistical Machine Translation*, 2008. [126]

Y Chang and M. Collins. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proc. of EMNLP*, 2011. [18]

E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*, 2005. [89]

C. Chen and O. Mangasarian. Hybrid misclassification minimization. *Advances in Computational Mathematics*, 5, 1996. [32]

S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical report 10-98, Harvard University, 1998. [50, 128]

C. Cherry and G. Foster. Batch tuning strategies for statistical machine translation. In *Proc. of NAACL*, 2012. [12, 34, 35, 47, 59]

D. Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL*, 2005. [4, 6, 14, 22, 50, 81, 111]

D. Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007. [5, 13, 22, 28, 57, 94, 105]

D. Chiang. Learning to translate with source and target syntax. In *Proc. of ACL*, 2010. [i, 2, 3, 5, 24]

D. Chiang. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13:11591187, 2012. [34, 39]

D. Chiang, S. DeNeefe, Y. Chan, and H. Ng. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proc. of EMNLP*, 2008a. [109]

D. Chiang, Y. Marton, and P. Resnik. Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*, 2008b. [i, 2, 5, 12, 24, 38, 39, 42, 46, 108]

D. Chiang, W. Wang, and K. Knight. 11,001 new features for statistical machine translation. In *Proc. of NAACL-HLT*, 2009. [2, 5, 12, 34, 37, 38, 39, 42, 46, 108, 109]

D. Chiang, J. Graehl, K. Knight, A. Pauls, and S. Ravi. Bayesian inference for finite-state transducers. In *Proc. of NAACL*, 2010. [108]

C. Christodoulopoulos, S. Goldwater, and M. Steedman. Two decades of unsupervised pos induction: how far have we come? In *Proc. of EMNLP*, 2010. [83]

J. H. Clark, C. Dyer, A. Lavie, and N. A. Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of ACL*, 2011. [33]

S. Cohen. *Computational Learning of Probabilistic Grammars in the Unsupervised Setting*. PhD thesis, Carnegie Mellon University, 2011. [6, 123]

S. Cohen and N. A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of NAACL*, 2009. [122]

S. Cohen, K. Gimpel, and N. A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proc. of NIPS*, 2008. [118]

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, 2002. [37, 44, 45]

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML*, 2006. [37]

T.M. Cover, J.A. Thomas, J. Wiley, et al. *Elements of information theory*, volume 6. Wiley Online Library, 1991. [10]

B. Cowan, I. Kučerová, and M. Collins. A discriminative model for tree-to-tree translation. In *Proc. of EMNLP*, 2006. [i, 2, 3, 23, 24]

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006. [37]

J. M. Crego and F. Yvon. Gappy translation units under left-to-right SMT decoding. In *Proc. of EAMT*, 2009. [81]

D. Das and N. A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*, 2009. [66]

H. Daumé III. MegaM: Maximum entropy model optimization package, ACL Data and Code Repository, ADCR2008C003, 2008. [50]

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977. [11, 28]

S. DeNeefe, K. Knight, W. Wang, and D. Marcu. What can syntax-based MT learn from phrase-based MT? In *Proc. of EMNLP-CoNLL*, 2007. [i, 2, 4, 71]

J. DeNero and D. Klein. Tailoring word alignments to syntactic machine translation. In *Proc. of ACL*, 2007. [17]

J. DeNero and J. Uszkoreit. Inducing sentence structure from parallel corpora for reordering. In *Proc. of EMNLP*, 2011. [111]

J. DeNero, S. Kumar, C. Chelba, and F. J. Och. Model combination for machine translation. In *Proc. of NAACL*, 2010. [4]

Y. Ding and M. Palmer. Machine translation using probabilistic synchronous dependency insertion grammar. In *Proc. of ACL*, 2005. [i, 2, 3, 23, 24, 25, 27, 66]

C. B. Do, Q. Le, C. H. Teo, O. Chapelle, and A. Smola. Tighter bounds for structured estimation. In *Proc. of NIPS*, 2008. [ii, 2, 5, 36, 37, 38]

B. J. Dorr. Machine translation divergences: a formal description and proposed solution. *Computational Linguistics*, 20(4), 1994. [i, 1, 2, 3]

M. Dredze, J. Blitzer, P.P. Talukdar, K. Ganchev, J. Graca, and F. Pereira. Frustratingly hard domain

adaptation for dependency parsing. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL*, 2007. [6]

R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. John Wiley, New York, 1973. [32]

C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL*, 2010. [22, 101]

V. Eidelman. Optimization strategies for online large-margin learning in machine translation. In *Proc. of the Seventh Workshop on Statistical Machine Translation*, 2012. [5, 47]

J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*, 1996. [89]

J. Eisner. Learning non-isomorphic tree mappings for machine translation. In *Proc. of ACL*, 2003. [3, 24]

J. Eisner and N. A. Smith. Parsing with soft and hard constraints on dependency length. In *Proc. of IWPT*, pages 30–41, 2005. [85, 92]

J. Eisner, E. Goldlust, and N. A. Smith. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*, 2005. [91, 94]

R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6), 1962. [91]

G. Foster. A maximum entropy/minimum divergence translation model. In *Proc. of ACL*, 2000. [116]

G. Foster and R. Kuhn. Stabilizing minimum error rate training. In *Proc. of Fourth Workshop on Statistical Machine Translation*, 2009. [33]

H. J. Fox. Phrasal cohesion and statistical machine translation. In *Proc. of EMNLP*, 2002. [2, 3, 27, 67]

Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3), 1999. [45]

M. Galley and C. D. Manning. Quadratic-time dependency parsing for machine translation. In *Proc. of ACL-IJCNLP*, 2009. [25, 27, 74]

M. Galley and C. D. Manning. Accurate non-hierarchical phrase-based translation. In *Proc. of NAACL*, 2010. [i, 2, 81]

M. Galley, M. Hopkins, K. Knight, and D. Marcu. What's in a translation rule? In *Proc. of HLT-NAACL*, 2004. [23]

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proc. of COLING-ACL*, 2006. [i, 2, 4, 23]

K. Ganchev, J. Gillenwater, and B. Taskar. Dependency grammar induction via bitext projection constraints. In *Proc. of ACL*, 2009. [6, 110]

J. Gao, G. Andrew, M. Johnson, and K. Toutanova. A comparative study of parameter estimation

methods for statistical natural language processing. In *Proc. of ACL*, 2007. [32]

Y. Gao, P. Koehn, and A. Birch. Soft dependency constraints for reordering in hierarchical phrase-based translation. In *Proc. of EMNLP*, 2011. [27]

F. Gécseg and M. Steinby. Tree automata. 1984. [25]

D. Gildea. Loosely tree-based alignment for machine translation. In *Proc. of ACL*, 2003. [3]

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, , and B. Taskar. Posterior sparsity in unsupervised dependency parsing. *Journal of Machine Learning Research*, 2010. [122]

K. Gimpel and N. A. Smith. Rich source-side context for statistical machine translation. In *Proc. of ACL-2008 Workshop on Statistical Machine Translation*, 2008. [24]

K. Gimpel and N. A. Smith. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proc. of EACL*, 2009a. [94]

K. Gimpel and N. A. Smith. Feature-rich translation by quasi-synchronous lattice parsing. In *Proc. of EMNLP*, 2009b. [7, 65, 89, 94]

K. Gimpel and N. A. Smith. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*, 2010a. [7, 36, 39, 48, 114]

K. Gimpel and N. A. Smith. Softmax-margin training for structured log-linear models. Technical report, Carnegie Mellon University, 2010b. [7, 36, 114]

K. Gimpel and N. A. Smith. Quasi-synchronous phrase dependency grammars for machine translation. In *Proc. of EMNLP*, 2011a. [7, 8, 65, 71, 89, 94, 95, 97, 105]

K. Gimpel and N. A. Smith. Generative models of monolingual and bilingual gappy patterns. In *Proc. of EMNLP-2011 Workshop on Statistical Machine Translation*, 2011b. [36]

K. Gimpel and N. A. Smith. Structured ramp loss minimization for machine translation. In *Proc. of NAACL*, 2012a. [7, 36, 54]

K. Gimpel and N. A. Smith. Concavity and initialization for unsupervised dependency parsing. In *Proc. of NAACL*, 2012b. [8, 95, 118]

Y. Goldberg, M. Adler, and M. Elhadad. EM can find pretty good HMM POS-taggers (when given a good start). In *Proc. of ACL*, 2008. [119]

J. Graehl and K. Knight. Training tree transducers. In *Proc. of HLT-NAACL*, 2004. [25]

J. Graehl, K. Knight, and J. May. Training tree transducers. *Computational Linguistics*, 34(3), 2008. [25]

K. Hall, R. McDonald, J. Katz-Brown, and M. Ringgaard. Training dependency parsers by jointly optimizing multiple objectives. In *Proc. of EMNLP*, 2011. [110]

G. Hanneman and A. Lavie. Automatic category label coarsening for syntax-based machine translation. In *Proc. of the Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2011. [i, 2, 3, 24]

R. Haque, S. K. Naskar, Y. Ma, and A. Way. Using supertags as source language context in SMT. In *Proc. of EAMT*, 2009. [24]

R. Haque, S. Kumar Naskar, A. Van Den Bosch, and A. Way. Supertags as source language context in hierarchical phrase-based smt. Association for Machine Translation in the Americas, 2010. [24]

R. Haque, S.K. Naskar, A. van den Bosch, and A. Way. Integrating source-language context into phrase-based statistical machine translation. *Machine translation*, pages 1–47, 2011. [24]

W. Headden III, M. Johnson, and D. McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proc. of NAACL*, 2009. [122]

K. Heafield. Kenlm: Faster and smaller language model queries. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, 2011. [50, 128]

H. Hoang, P. Koehn, and A. Lopez. A Unified Framework for Phrase-Based, Hierarchical, and Syntax-Based Statistical Machine Translation. In *Proc. of IWSLT*, 2009. [22, 50, 128]

M. Hopkins and J. May. Tuning as ranking. In *Proc. of EMNLP*, 2011. [5, 12, 29, 31, 33, 34, 36, 47, 50, 51, 57, 59, 62, 108]

C. Hsieh, K. Chang, C. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proc. of ICML*, 2008. [46]

L. Huang. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*, 2008. [89]

L. Huang and D. Chiang. Better k-best parsing. In *Proc. of IWPT*, 2005. [93]

L. Huang and D. Chiang. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL*, 2007. [13]

L. Huang, K. Knight, and A. Joshi. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA*, 2006. [4, 23, 93]

T. Hunter and P. Resnik. Exploiting syntactic relationships in a phrase-based decoder: an exploration. *Machine Translation*, 24(2), 2010. [27]

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–25, 2005. [6, 110]

F. Jelinek. *Statistical methods for speech recognition*. MIT Press, 1997. [48, 115, 116, 117]

M. Johnson. Transforming projective bilexical dependency grammars into efficiently-parsable cfgs with unfold-fold. In *Proc. of ACL*, 2007. [101]

M. Johnson, T. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proc. of NAACL*, 2007. [101]

Aravind Joshi and Yves Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, 1997. [24]

B. H. Juang, W. Chou, and C. H. Lee. Minimum classification error rate methods for speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 5(3):257–265, may 1997. [32]

M. Kääriäinen. Sinuhe: statistical machine translation using a globally trained conditional expo-

nential family translation model. In *Proc. of EMNLP*, 2009. [16, 35]

J. Kaiser, B. Horvat, and Z. Kacic. A novel loss function for the overall risk criterion based discriminative training of hmm models. In *Proc. of ICSLP*, 2000. [33]

J. Katz-Brown, S. Petrov, R. McDonald, F. Och, D. Talbot, H. Ichikawa, M. Seno, and H. Kazawa. Training a parser for machine translation reordering. In *Proc. of EMNLP*, 2011. [110]

D. Klein and C. Manning. Parsing and hypergraphs. In *Proc. of IWPT*, 2001. [23]

D. Klein and C. D. Manning. A generative constituent-context model for improved grammar induction. In *Proc. of ACL*, 2002. [ii, 3, 6]

D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS 15*, 2003. [3]

D. Klein and C. D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*, 2004. [ii, 3, 6, 118, 119, 121, 122]

K. Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4), 1999. [13]

P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, 2003. [i, 2, 4, 13, 14, 15, 17, 18, 50, 82, 83]

P. Koehn, A. Axelrod, A.B. Mayne, C. Callison-Burch, M. Osborne, and D. Talbot. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proc. of IWSLT*, 2005. [18]

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL (demo session)*, 2007. [5, 15, 50, 108, 128]

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010. [9]

S. Kübler, R. McDonald, and J. Nivre. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool, US, 2009. ISBN 9781598295962. [25]

J. Kuhn. Experiments in parallel-text based grammar induction. In *Proc. of ACL*, 2004. [6, 110]

A. Kulesza and B. Taskar. Structured determinantal point processes. In *Proc. NIPS*, 2010. [110]

S. Kumar and W. Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Proc. of HLT-NAACL*, 2004. [13]

S. Kumar, W. Macherey, C. Dyer, and F. Och. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *Proc. of ACL-IJCNLP*, 2009. [33, 34, 109]

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, 2001. [37, 40, 47]

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990. [124]

D. Lashkari and P. Golland. Convex clustering with exemplar-based models. In *Proc. of NIPS*, 2008. [121]

Y. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. Language model based Arabic word segmentation. In *Proc. of ACL*, 2003. [125]

R. Levy and C. D. Manning. Is it harder to parse chinese, or the chinese treebank? In *Proc. of ACL*, 2003. [72, 96, 126]

Z. Li and J. Eisner. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proc. of EMNLP*, 2009. [33, 34, 49]

Z. Li and S. Khudanpur. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proc. of the Second Workshop on Syntax and Structure in Statistical Translation*, 2008. [21]

Z. Li, C. Callison-Burch, C. Dyer, S. Khudanpur, L. Schwartz, W. Thornton, J. Weese, and O. Zaidan. Joshua: An open source toolkit for parsing-based machine translation. In *Proc. of WMT*, 2009a. [22]

Z. Li, J. Eisner, and S. Khudanpur. Variational decoding for statistical machine translation. In *Proc. of ACL*, 2009b. [13]

Z. Li, T. Liu, and W. Che. Exploiting multiple treebanks for parsing with quasi-synchronous grammars. In *Proc. of ACL*, 2012. [66]

P. Liang. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005. [128]

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. An end-to-end discriminative approach to machine translation. In *Proc. of COLING-ACL*, 2006a. [i, 2, 5, 29, 35, 37, 39, 45, 108]

P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *Proc. of HLT-NAACL*, 2006b. [17]

C. Lin and F. J. Och. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. of COLING*, 2004. [31, 51]

Dekang Lin. A path-based transfer model for machine translation. In *Proc. of COLING*, 2004. [23, 27]

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528, 1989. [101]

Y. Liu, Q. Liu, and S. Lin. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL*, 2006. [i, 2, 23]

Y. Liu, Y. Huang, Q. Liu, and S. Lin. Forest-to-string statistical translation rules. In *Proc. of ACL*, 2007. [103]

Y. Liu, Y. Lü, and Q. Liu. Improving tree-to-tree translation with packed forests. In *Proc. of ACL-IJCNLP*, 2009a. [i, 2, 3, 23, 24]

Y. Liu, H. Mi, Y. Feng, and Q. Liu. Joint decoding with multiple translation models. In *Proc. of ACL-IJCNLP*, 2009b. [4]

R. B. Lyngsø and C. N. S. Pedersen. The consensus string problem and the complexity of comparing hidden markov models. *Journal of Computing and System Science*, 65:545–569, 2002. [12]

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. Lattice-based minimum error rate training for statistical machine translation. In *EMNLP*, 2008. [33, 34, 109]

D. M. Magerman and M. P. Marcus. Parsing a natural language using mutual information statistics. In *Proc. of AAAI*, 1990. [ii, 3]

I. Mani. *Automatic summarization*, volume 3. John Benjamins Pub Co, 2001. [110]

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, 1993. [ii, 2, 5, 23]

D. Mareček and Z. Žabokrtský. Gibbs sampling with treeness constraint in unsupervised dependency parsing. In *Proc. of Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, 2011. [121]

A. F. T. Martins, N. A. Smith, and E. P. Xing. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL*, 2009. [68, 76, 97, 125]

A. F. T. Martins, K. Gimpel, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M A. T. Figueiredo. Learning structured classifiers with dual coordinate descent. Technical report, Carnegie Mellon University, 2010. [46]

Y. Marton and P. Resnik. Soft syntactic constraints for hierarchical phrased-based translation. *Proc. of ACL*, 2008. [24]

J. May and K. Knight. A better n-best list: Practical determinization of weighted finite tree automata. In *Proc. of HLT-NAACL*, 2006. [13]

D. McAllester and J. Keshet. Generalization bounds and consistency for latent structural probit and ramp loss. In *Proc. of NIPS*, 2011. [5, 37, 39]

D. McAllester, T. Hazan, and J. Keshet. Direct loss minimization for structured prediction. In *Advances in NIPS 23*, 2010. [39, 46]

D. McClosky, E. Charniak, and M. Johnson. Automatic domain adaptation for parsing. In *Proc. of NAACL*, 2010. [6]

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*, 2005. [26]

I.D. Melamed. Multitext grammars and synchronous parsers. In *Proc. of NAACL*, 2003. [24]

H. Mi and L. Huang. Forest-based translation rule extraction. In *Proc. of EMNLP*, 2008. [103]

H. Mi, L. Huang, and Q. Liu. Forest-based translation. In *Proc. of ACL*, 2008. [103]

R. C. Moore and C. Quirk. Random restarts in minimum error rate training for statistical machine translation. In *Proc. of Coling*, 2008. [33]

T. Naseem, H. Chen, R. Barzilay, and M. Johnson. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP*, 2010. [6, 121]

M.-J. Nederhof. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29

(1), 2003. [91, 94]

NIST. NIST 2009 machine translation evaluation official results, 2009. URL `www.itl.nist.gov/iad/mig/tests/mt/2009/`. [4]

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proc. of CoNLL*, 2007. [3, 123]

F. J. Och. Minimum error rate training for statistical machine translation. In *Proc. of ACL*, 2003. [4, 12, 28, 32, 33, 36]

F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*, 2002. [i, 2, 5, 11, 12, 15, 28, 29, 47]

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1), 2003. [17, 50, 68, 128]

F. J. Och, N. Ueffing, and H. Ney. An efficient a* search algorithm for statistical machine translation. In *Proc. of the workshop on Data-driven Methods in Machine Translation*, 2001. [21]

F.J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4), 2004. [15]

Naoaki Okazaki. Classias: a collection of machine-learning algorithms for classification, 2009. URL `http://www.chokkan.org/software/classias/`. [51]

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, 2001. [30, 50]

J.H. Park, W.B. Croft, and D.A. Smith. A quasi-synchronous dependence model for information retrieval. In *Proc. of CIKM*, 2011. [66]

M. A. Paskin. Grammatical bigrams. In *Advances in NIPS 15*, 2002. [ii, 3]

S. Petrov. *Coarse-to-Fine Natural Language Processing*. PhD thesis, University of California at Berkeley, 2009. [89]

S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset. In *Proc. of LREC*, 2012. [ii, 2, 5, 121]

D. Povey and P. C. Woodland. Minimum phone error and I-smoothing for improved discrimative training. In *Proc. of ICASSP*, 2002. [33]

D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah. Boosted MMI for model and feature space discriminative training. In *Proc. of ICASSP*, 2008. [48, 114]

K. Probst, L. Levin, E. Peterson, A. Lavie, and J. Carbonell. Mt for minority languages using elicitation-based learning of syntactic transfer rules. *Machine Translation*, 17(4), 2002. [24]

A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *NIPS 17*, 2004. [47]

C. Quirk, A. Menezes, and C. Cherry. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*, 2005. [27]

A. N. Rafferty and C. D. Manning. Parsing three german treebanks: lexicalized and unlexicalized

baselines. In *Proc. of the Workshop on Parsing German*, 2008. [68, 96, 127]

N. Ratliff, J. A. Bagnell, and M. Zinkevich. Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Output Spaces*, 2006. [46]

S. Riezler and J.T. Maxwell III. Grammatical machine translation. In *Proc. of NAACL*, 2006. [27]

R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10(3), 1996. [57]

B. Sapp, D. Weiss, and B. Taskar. Sidestepping intractable inference with structured ensemble cascades. In *NIPS*, 2010. [92]

Y. Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proc. of COLING*, 1992. [24]

F. Sha and L. K. Saul. Large margin hidden Markov models for automatic speech recognition. In *Proc. of NIPS*, 2006. [114]

L. Shen, J. Xu, and R. Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL*, 2008. [23, 25, 27, 74]

L. Shen, J. Xu, and R. Weischedel. String-to-dependency statistical machine translation. *Computational Linguistics*, 36(4):649–671, 2010. [27]

S.M. Shieber and Y. Schabes. Synchronous tree-adjoining grammars. In *Proc. of ACL*, 1990. [3, 24]

K. Sima'an. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151, 2002. [12]

M. Simard, N. Cancedda, B. Cavestro, M. Dymetman, E. Gaussier, C. Goutte, K. Yamada, P. Langlais, and A. Mauser. Translating with non-contiguous phrases. In *Proc. of HLT/EMNLP*, 2005. [81]

P. Simianer, S. Riezler, and C. Dyer. Joint feature selection in distributed stochastic learning for large-scale discriminative training in smt. In *Proc. of ACL*, 2012. [35]

A. Sixtus and S. Ortmanns. High quality word graphs using forward-backward pruning. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1999. [57, 92]

D. A. Smith and J. Eisner. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of HLT-NAACL Workshop on Statistical Machine Translation*, 2006a. [i, viii, 2, 3, 5, 25, 65, 66, 67, 73, 87]

D. A. Smith and J. Eisner. Minimum risk annealing for training log-linear models. In *Proc. of COLING-ACL*, 2006b. [29, 33, 36, 48]

D. A. Smith and J. Eisner. Parser adaptation and projection with quasi-synchronous features. In *Proc. of EMNLP*, 2009. [6, 66, 110]

N. A. Smith. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. PhD thesis, Johns Hopkins University, 2006. [6, 101]

N.A. Smith. Linguistic structure prediction. *Synthesis Lectures on Human Language Technologies*, 4 (2):1–274, 2011. [5, 116]

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with

targeted human annotation. In *Proc. of AMTA*, 2006. [51]

B. Snyder, T. Naseem, and R. Barzilay. Unsupervised multilingual grammar induction. In *Proc. of ACL*, 2009. [6, 110]

Anders Søgaard and Jonas Kuhn. Empirical lower bounds on alignment error rates in syntax-based machine translation. In *Proc. of the Third Workshop on Syntax and Structure in Statistical Translation*, 2009. [2, 3, 24, 25, 67]

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Proc. of NAACL-HLT*, 2010. [6, 124]

V. I. Spitkovsky, H. Alshawi, A. X. Chang, and D. Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proc. of EMNLP*, 2011a. [101, 122]

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. Punctuation: Making a point in unsupervised dependency parsing. In *Proc. of CoNLL*, 2011b. [122]

M. Steedman. *The Syntactic Process*. MIT Press, 2000. [24]

A. Stolcke. SRILM—an extensible language modeling toolkit. In *Proc. of ICSLP*, 2002. [50, 128]

V. Stoyanov and J. Eisner. Minimum-risk training of approximate crf-based nlp systems. In *Proc. of NAACL*, 2012. [48, 114]

J. Su, Y. Liu, H. Mi, H. Zhao, Y. Lü, and Q. Liu. Dependency-based bracketing transduction grammar for statistical machine translation. In *Proc. of COLING*, 2010. [27]

X. Sun, T. Matsuzaki, D. Okanohara, and J. Tsujii. Latent variable perceptron algorithm for structured classification. In *Proc. of IJCAI*, 2009. [45]

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in NIPS 16*, 2003. [39, 46]

L. Tesnière. *Élément de Syntaxe Structurale*. Klincksieck, 1959. [4, 25]

K. Toutanova and M. Galley. Why initialization matters for IBM Model 1: Multiple optima and non-strict convexity. In *Proc. of ACL*, 2011. [118, 119]

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*, 2003. [101]

R. Tromble, S. Kumar, F. Och, and W. Macherey. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *EMNLP*, 2008. [35, 51, 57, 92]

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6, 2005. [46]

Z. Tu, Y. Liu, Y. Hwang, Q. Liu, and S. Lin. Dependency forest for statistical machine translation. In *Proc. of COLING*, 2010. [27]

N. Ueffing, F. J. Och, and H. Ney. Generation of word graphs in statistical machine translation. In *Proc. of EMNLP*, 2002. [18, 89]

V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. [31]

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algo-

rithm. *IEEE Transactions on Information Processing*, 13(2), 1967. [13]

M. Wang, N. A. Smith, and T. Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*, 2007. [66]

T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki. Online large-margin training for statistical machine translation. In *Proc. of EMNLP-CoNLL*, 2007. [i, 2, 5, 12, 29, 42, 46, 47, 108]

B. Wellington, S. Waxmonsky, and I.D. Melamed. Empirical lower bounds on the complexity of translational equivalence. In *Proc. of COLING-ACL*, 2006. [2, 3, 24, 67]

K. Woodsend and M. Lapata. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proc. of EMNLP*, 2011. [66]

K. Woodsend, Y. Feng, and M. Lapata. Title generation with quasi-synchronous grammar. In *Proc. of EMNLP*, 2010. [66]

D. Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, 1997. [22, 24, 67]

Y. Wu, Q. Zhang, X. Huang, and L. Wu. Phrase dependency parsing for opinion mining. In *Proc. of EMNLP*, 2009. [72, 74, 103]

J. Wuebker, A. Mauser, and H. Ney. Training phrase translation models with leaving-one-out. In *Proc. of ACL*, 2010. [35]

F. Xia and M. McCord. Improving a statistical mt system with automatically learned rewrite patterns. In *Proc. of COLING*, 2004. [24]

J. Xie, H. Mi, and Q. Liu. A novel dependency-to-string model for statistical machine translation. In *Proc. of EMNLP*, 2011. [27]

D. Xiong, Q. Liu, and S. Lin. A dependency treelet string correspondence model for statistical machine translation. In *Proc. of the Second Workshop on Statistical Machine Translation*, 2007. [27]

D. Xiong, M. Zhang, A. Aw, and H. Li. A linguistically annotated reordering model for btg-based statistical machine translation. In *Proc. of ACL*, 2008. [24]

Y. Xiong, J. Zhu, H. Huang, and H. Xu. Minimum tag error for discriminative training of conditional random fields. *Inf. Sci.*, 179(1-2):169–179, 2009. [49]

K. Yamada and K. Knight. A syntax-based statistical translation model. In *Proc. of ACL*, 2001. [i, 2, 3, 4, 23]

K. Yamada and K. Knight. A decoder for syntax-based statistical mt. In *Proc. of ACL*, 2002. [23]

D. Yarowsky and G. Ngai. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proc. of NAACL*, 2001. [6, 110]

D. Yarowsky, G. Ngai, and R. Wicentoswki. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proc. of HLT*, 2001. [6, 110]

C. J. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proc. of ICML*, 2009. [42, 47]

A. L. Yuille and Anand Rangarajan. The concave-convex procedure (cccp). In *Advances in NIPS*

14. MIT Press, 2002. [40]

D. Yuret. *Discovery of Linguistic Relations Using Lexical Attraction*. PhD thesis, MIT, 1998. [ii, 3]

M. Zaslavskiy, M. Dymetman, and N. Cancedda. Phrase-based statistical machine translation as a traveling salesman problem. In *Proc. of ACL*, 2009. [18]

R. Zens and H. Ney. A comparative study on reordering constraints in statistical machine translation. In *Proc. of ACL*, 2003. [67]

R. Zens, F. Och, and H. Ney. Phrase-based statistical machine translation. *KI 2002: Advances in Artificial Intelligence*, 2002. [15]

R. Zens, S. Hasan, and H. Ney. A systematic comparison of training criteria for statistical machine translation. In *Proc. of EMNLP*, 2007. [33, 48]

H. Zhang, L. Huang, D. Gildea, and K. Knight. Synchronous binarization for machine translation. In *Proc. of NAACL*, 2006. [67]

J. Zhang, F. Zhai, and C. Zong. Augmenting string-to-tree translation models with fuzzy use of source-side syntax. In *Proc. of EMNLP*, 2011. [i, 2, 3, 24]

A. Zollmann. *Learning Multiple-Nonterminal Synchronous Grammars for Statistical Machine Translation*. PhD thesis, Carnegie Mellon University, 2011. [111]

A. Zollmann and A. Venugopal. Syntax augmented machine translation via chart parsing. In *Proc. of NAACL 2006 Workshop on Statistical Machine Translation*, 2006. [i, 2, 4, 23]

A. Zollmann and S. Vogel. A word-class approach to labeling pscfg rules for machine translation. In *Proc. of ACL*, 2011. [6, 101, 111]