ABSTRACT

Title of dissertation: IMPROVED ONLINE LEARNING AND MODELING FOR FEATURE-RICH DISCRIMINATIVE MACHINE TRANSLATION

Vladimir Eidelman, Doctor of Philosophy, 2013

Dissertation directed by: Professor Philip Resnik
Department of Linguistics and
Institute for Advanced Computer Studies

Machine translation represents one of the core tasks in natural language processing: performing an automatic analysis of input text to produce a structured output. Most modern statistical machine translation (SMT) systems *learn* how to translate by constructing a discriminative model based on statistics from the data. While this allows the introduction of feature functions representing important attributes of the translation process, the associated model parameters must be learned from data. A growing number of methods for discriminative training in SMT have been proposed, but most suffer from limitations that hinder their utility for training feature-rich models on large amounts of data.

In this thesis, we present novel models and learning algorithms that address this issue by tackling three core problems for discriminative training: what to optimize, how to optimize, and how to represent the input. The first issue amounts to selecting an appropriate objective function, the second to properly searching for parameters that optimize that objective, and the third to extracting informative features of the data. In addressing these issues, we develop fast learning algorithms that are both suitable for large-scale

SMT training and capable of generalization in high-dimensional feature spaces.

The algorithms are developed in an online margin-based framework. Their online nature facilitates easily scaling to perform discriminative training on large training sets, and maximum margin optimization allows efficient feature-rich learning, permitting exploration into many new types of sparse features for translation.

While online margin-based methods are firmly established in the machine learning community, their adaptation to machine translation is not straightforward. Thus, the first problem we address is *what* to optimize when learning for SMT, which involves loss-augmented inference over latent variables. To this end, we define a family of objective functions for large-margin learning in SMT and investigate their optimization performance in standard and high-dimensional feature spaces. We also show that this approach shows promise not just with respect to the size of the feature space, but with respect to the size of the tuning data.

After establishing what to optimize, the second problem we focus on is *how* to improve learning in the feature-rich space. While the goal of all learning methods is to produce a model from a limited number of training instances that generalizes well to unseen data, this becomes increasingly difficult as the feature dimension grows. Following recent developments in machine learning that show that generalization ability can be improved by incorporating higher order information into the optimization, we develop an online gradient-based algorithm based on Relative Margin Machines that improves upon large-margin learning by considering and bounding the spread of the data while maximizing the margin.

By utilizing the learning regimes developed thus far, we are able to focus on the

third problem and introduce new features specifically targeting generalization to new domains. While domain adaptation has typically been done with manually defined domains and corpora, we employ topic models to perform unsupervised domain induction, and introduce translation model adaptation features based on probabilistic domain membership.

As a final question, we look at how to take advantage of the latent derivation structure for optimization. In current models of SMT, there is an exponential number of derivations by which the same translation output can be produced. However, only the final output is observed, thereby necessitating the treatment of these paths as a latent variable. The standard practice is to sidestep this ambiguity by treating each derivation as an individual translation, and performing training and inference toward a single way of producing the output. While addressing the learning problem above, we were still firmly situated in that standard regime. In the final part of the thesis, we revisit and present a more thorough approach for *what* to optimize, defining a framework for latent variable models which explicitly takes advantage of all derivations in both learning and inference. We present a novel loss function for large-margin learning in that setting, and develop a suitable optimization algorithm for training an SMT system with this objective.

IMPROVED ONLINE LEARNING AND MODELING FOR FEATURE-RICH DISCRIMINATIVE MACHINE TRANSLATION

by

Vladimir Alexander Eidelman

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Philip Resnik, Chair/Advisor
Professor Hal Daumé III
Professor Noah Smith
Professor Jimmy Lin
Professor Héctor Corrada Bravo

Dedication

lovingly to my grandparents, parents, and Vedia

Acknowledgments

He who would learn to fly one day must first learn to stand and walk and run and climb and dance; one cannot fly into flying.

— Friedrich Nietzsche

While the following words of gratitude and reverence cannot do justice to the many people who have made the last five years the most fulfilling, memorable, and productive of my life, I hope they can express how extremely fortunate I was in having such wonderful friends, colleagues, and mentors to guide me through this dissertation.

I owe the largest debt of gratitude to my advisor, Philip Resnik, whose immeasurable energy, inexhaustible spring of ideas, and general zeal for life and research are at once intimidating and infectious. His ability to turn my words into comprehensible, and at times even intelligent thoughts, still astounds me, as does the way he corners the core research questions while remaining cognizant of the overarching potential. If I have become a better researcher, it is through his support and guidance, and hopefully a good dose of learning through osmosis.

The members of my dissertation committee - Hal Daumé III, Héctor Corrada Bravo, Jimmy Lin, and Noah Smith - deserve special thanks for taking the time to encourage me to take a broader view of my work, help me establish connections which I had not made, ask probing and insightful questions about theory and experimentation, and positively affect this dissertation overall with their invaluable feedback.

I wish to thank Mary Harper for advising me in my formative years as a graduate student, and patiently guiding me through my first research endeavors. I am also very fortunate to have been surrounded by the other diverse and brilliant faculty in the CLIP

lab. Bonnie Dorr, Doug Oard, Jordan Boyd-Graber, and Louiqa Raschid broadened my horizon through many interesting conversions.

I would like to thank Goran Trajkovski for introducing me to research at the NSF REU program in 2006, Kathy McKeown at Columbia, without whose guidance I would not have found my way into computational linguistics, and Massimo Poesio and the group at the 2007 CLSP summer workshop at JHU, especially Yannick Versley and Simone Ponzetto, for cementing my interest in graduate study.

Along the way, my amazing CLIP lab mates Amit Goyal, Asad Sayeed, Ferhan Ture, Hendra Setiawan, Jagadeesh Jagarlamudi, Ke Wu, Kristy Hollingshead Seitz, Nitin Madnani, Raul Guerra, Saif Mohammad, Viet-An Nguyen, Yuval Marton, and Zhongqiang Huang have been an invaluable source of inspiration, knowledge and friendship, creating a true atmosphere of camaraderie, in which I was lucky enough to feel at home. I also enjoyed interacting with many other group members, especially Adam Lee, Chang Hu, Denis Filimonov, Earl Wagner, Eric Hardisty, Greg Sanders, Hassan Sayyadi, Jiarong Jiang, Junhui Li, Ke Zhai, Matt Snover, Taesun Moon, Tamer Elsayad, Yakov Kronrod, and Yuening Hu. A special thanks goes to Chris Dyer for his continual mentorship and for creating the best decoder you could ask for.

I have been greatly influenced and fortunate enough to learn from others in the community with more expertise than I, notably Adam Lopez, Colin Cherry, George Foster, Jason Smith, Jon May, Kevin Gimpel, Matt Post, Pannaga Shivaswamy, Phil Blunsom, Spence Green, Taro Watanabe, Trevor Cohn, and especially David Chiang, whose work laid the foundations for much of this dissertation and without whose seminal contributions, my own would not have been possible. I also learned a great deal during my

internship at BBN, for which I thank Jacob Devlin, John Makhoul, Rabih Zbib and Spyros Matsoukas.

I am grateful for the financial and computational[1] support provided by the National Science Foundation Graduate Research Fellowship and the Department of Defense through the National Defense Science and Engineering Graduate Fellowship Program, which have allowed me the freedom to pursue my sometimes esoteric research interests and have supported most of the work in this dissertation.[2] While I may have finished eventually, it would have taken another five years if it were not for the immediate and patient technical support provided by Joe Webster and the UMIACS support staff.

I am forever grateful and send my deepest love to my entire family. To my mom, Marina, for her unequivocal encouragement; to my dad, Alex, for never letting me rest on my laurels; and to both, for having sacrificed a great deal to provide me with a life that allowed me to pursue my dreams. To Grandma Zhenya for her endless nurturing support, and to Grandma Ida for inciting in me a desire for intellectual curiosity.

Finally, to my dearest wife Vedia, who has made the lows of my (*our*) journey bearable, and the highs worthwhile. Your love and presence have not only offered welcome respite on many occasions, but have more often served as an inspiration. I would not be here today without you.

[1]This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575, specifically allocation TG-CCR130026 on Stampede.

[2]This research was also supported in part by the GALE and BOLT programs of the Defense Advanced Research Projects Agency, Contracts HR0011-06-2-001 and HR0011-12-C-0015, respectively, by NSF under awards CCF-1018625, IIS-0916043 and IIS-1144034, and ARL Cooperative Agreement W911NF-09-2-0072. Any opinions, findings, conclusions, or recommendations expressed are the authors and do not necessarily reflect those of the sponsors.

Contents

List of Tables

List of Figures

1 Introduction

> *We know very little, and yet it is astonishing that we know so much, and still more astonishing that so little knowledge can give us so much power.*
>
> — Bertrand Russell

Machine translation represents one of the core tasks in natural language processing (NLP): performing an automatic analysis of input text to produce a structured output; in this case, a translation of a sentence. Although early machine translation systems used manually encoded knowledge, given the complexities and ambiguity inherent in language, coupled with the available computational power and amount of translated examples of data, modern statistical machine translation (SMT) systems *learn* how to translate by constructing a discriminative model based on statistics from the data (Koehn, 2010).

Translation rule patterns are extracted using a corpus of translation examples, allowing important attributes for the translation process to be selected and encoded with the use of feature functions. The chief aim of learning is then to properly set the corresponding model parameters to score alternative translation hypotheses in accordance with an external translation quality metric. However, there are a number of difficulties that make the application of machine learning in this setting uniquely challenging. Despite a grow-

ing number of proposed methods for discriminative training, most suffer from limitations that hinder their utility for training feature-rich models on large amounts of data.

In this thesis, we present novel models and learning algorithms that address three core problems for discriminative training: *what* to optimize, *how* to optimize, and *how* to represent the input. The first issue amounts to selecting an appropriate objective function, the second to properly searching for parameters that optimize that objective, and the third to extracting informative features of the data. In addressing these issues, we develop *efficient learning algorithms* that are both suitable for *large-scale SMT training* and capable of *generalization in high-dimensional feature spaces* for structured output.

While training is a crucial step in SMT system development, it remains one of the most time and resource consuming. To achieve faster training time and facilitate easily scaling to perform discriminative training on large training sets, our algorithms are developed in an **online learning** framework. Although most current systems tend to employ only a handful of features of the data which are believed to be important, there could potentially be millions (Chiang et al., 2009; Watanabe et al., 2007). Thus, a **large-margin criterion** is used for optimization, as it allows efficient **feature-rich learning**, permitting exploration into many new types of sparse features for translation.

While online margin-based methods are firmly established in the machine learning community, their adaptation to machine translation is not straightforward. Thus, each chapter in the dissertation is devoted to defining and examining an issue with training for SMT, presenting novel learning algorithms and methods to address it, as well as empirical results on an array of languages (French, Czech, German, Russian, Chinese and Arabic) to verify the effectiveness of our methods.

1.1 Outline of the dissertation

We begin in Chapter 2 by formally introducing the problem of statistical machine translation, specifically highlighting the structure of translation models and the problems they pose for inference and learning. The focus then turns more specifically to learning, introducing the fundamental algorithms which are used throughout the rest of the dissertation. Chapters 3, 4, 5, and 6 propose new methods for learning, introducing both algorithms and features for better estimation of SMT model parameters.

Chapter 3 introduces and addresses the *divergence* between the standard application of online large margin learning, and learning in machine translation. Since MT presents a setting *without* unique correct outputs and *with* unobserved structures, the first problem we address is *what* to optimize when learning with *loss-augmented inference over latent variables*. By defining a family of objective functions for large-margin learning in SMT and empirically investigating their optimization performance in standard and high-dimensional feature spaces, we show that this approach shows promise not just with respect to the size of the feature space, but with the size of the tuning data.

After establishing what to optimize, in Chapter 4 we focus on *how* to improve learning in the feature-rich space. While the goal of all learning methods is to produce a model from a limited number of training instances that generalizes well to unseen data, this becomes increasingly difficult as the feature dimension grows. Following recent developments in machine learning that show that generalization ability can be improved by incorporating higher order information into the optimization, we develop an online gradient-based algorithm based on Relative Margin Machines (Shivaswamy and Jebara,

2009b) that improves upon large-margin learning by considering and *bounding the spread of the data* while maximizing the margin.

Utilizing the learning regimes presented in Chapters 3 and 4, Chapter 5 introduces a new type of feature specifically targeting *generalization to new domains*. While domain adaptation has typically been done with manually defined domains and corpora, we employ topic models to perform unsupervised domain induction and bias machine translation systems toward relevant translations based on topic-specific contexts by introducing adaptation features based on probabilistic domain membership.

As a final question, in Chapter 6, we look at how to take advantage of all the latent derivations during optimization. In current models of SMT, there are an *exponential* number of derivational paths by which the same translation output string can be produced. However, only the final output is observed, thereby necessitating the treatment of these derivations as a latent variable. While the probability of a translation is the total probability of all its derivations, finding the maximum probability translation is intractable. Thus, the standard practice is to sidestep this ambiguity by treating each derivation as an individual translation, and performing training and inference toward a single way of producing the output by using the maximum Viterbi derivation. While addressing the learning problem above, we were still firmly situated in this standard regime. Chapter 6 revisits *what* to optimize and presents an approach moving to maximum translation decoding and training, by introducing a new way of combining a probabilistic framework for dealing with latent variables with a geometric model that performs margin maximization. We present a unified representation of a family of *latent structured losses* which explicitly account for this ambiguity in both learning and inference, from which we de-

velop a novel loss function for large-margin learning in the latent variable setting along with a suitable optimization algorithm for training an SMT system with this objective.

1.2 Contributions

This thesis makes a number of research contributions to machine learning and its application in machine translation.

1.2.1 Machine Learning

- We show that commonly used loss-augmented objectives for large-margin learning can be generalized into a single family of loss functions with different parameterizations.
- We introduce a loss function for structured relative margin with loss-augmented inference and latent variables, and derive an online gradient-based solver, RM, with a closed-form parameter update to optimize the relative margin loss.
- We present a unified representation and show the relationship between different latent structured losses, from which we introduce a novel objective for combining a probabilistic framework for dealing with latent variables with a geometric model that performs margin maximization.

1.2.2 Applications to Machine Translation

- We perform a comprehensive empirical analysis of optimization strategies within the family of loss-augmented objective functions applicable to SMT. We show that

the choice of inference for loss-augmented hypotheses used in training plays an important role in the stability and effectiveness of the large-margin update.

- We show that incorporating additional information, through bounding the spread utilizing our RM algorithm, significantly outperforms the standard large-margin update.

- We show that we can reinterpret the common notion of domains in MT with unsupervised domain induction using topic models, and introduce a method for performing dynamic domain adaptation using features representing probabilistic domain membership which yields significant gains for SMT.

- We describe how to learn using the maximum probability translation and show that maximum translation learning and inference does improve upon the standard maximum derivation regime.

- We develop an open-source tool for large-scale, decoder-agnostic, large-margin learning with Hadoop, and we perform experiments to show the practical efficiency, as well as preliminary advantages and challenges, of large-scale learning.

2 Statistical Machine Translation

> *Behold, they are one people, and they have all one language ... and nothing that they propose to do will now be impossible for them. Come, let us go down and there confuse their language, so that they may not understand one another's speech.*
>
> — Genesis 11:6-7

> *Shall I refuse my dinner because I do not fully understand the process of digestion?*
>
> — Oliver Heaviside

This chapter formally introduces the problem of statistical machine translation. Although a thorough review of SMT is beyond the scope of this thesis, we introduce the fundamental concepts necessary for the remainder of this dissertation, specifically highlighting points of particular relevance to this work. The structure of the discussion is divided into three main research areas: translation modeling and decoding (§2.2), model parameterization (§2.3), and learning (§2.4).

We delve into the most detail on background related to the last of these, as it draws heavily upon concepts in machine learning, and encompasses prior work directly related to the main topic of this thesis, discriminative optimization of SMT system parameters.[1]

[1] For a complete introduction to SMT, we refer the reader to Lopez (2008) and Koehn (2010).

2.1 Background

The roots of machine translation can be traced to the creation of programmable digital computers over half a century ago, when it was hoped that imbuing a machine with programmed knowledge about translation rules might allow it to act intelligently enough to automatically translate from a foreign **source** language sentence to a desired **target** language sentence accurately and fluently. Researchers endeavored to develop rule-based systems with a manually determined set of translation rules and lexicon (Koehn, 2010). As it became apparent that the problem of language translation, as with other problems in computational linguistics, encompasses far more ambiguity and complexity than a prescribed knowledge-based formalism can manage, research efforts were at first stymied, and then after some time, turned toward building a system based on the empirical evidence of human translation (Hutchins and Somers, 1992). However, it was not until the availability of large data sources and increased computational processing power allowed the introduction of machine learning that the field of **statistical machine translation** began to show promise.

After successfully applying statistical techniques to automatic speech recognition, Brown et al. (1990) at IBM adapted optimization techniques to machine translation that captured uncertainty and made decisions through probabilistic modeling. Statistical machine translation became the problem of automating language translation by designing systems that **learn** how to translate using a statistical analysis of large bilingual corpora of human translations. In essence, translation is reduced to a machine learning problem. Thus, in principle, given enough samples of source sentences paired with correspond-

ing translated target sentences, an SMT system could be built for any language pair with minimal time and labor.

As this mode of thinking about machine translation has become all but ubiquitous in academic and commercial applications, the core questions of research interest have turned out to be the following:

- **Modeling:** How do we **model** and **evaluate** the mapping between the source and target sentences in order to create and select the best translation? This breaks down into:
  - The type of rules that we use to map from source to target
  - Features of the source and target used to evaluate the quality of the translation
  - A weighting on the features that tells us how much importance we place in each features value
  - A model that makes use of the features and weights to allow us to make a decision
- **Learning:** How do we **optimize** the weights on each feature so that the translation with the best model score is *actually* the best translation?
- **Decoding:** How do we **search** through our model's possible translations to select the best one?

In the following sections, we address each of these core questions in turn, beginning with how we model the translation process.

2.2 Modeling

To construct a statistical MT system we rely on training data, which consists of a parallel corpus of translated text in the source and target languages.[2] Formally, our training data $\mathcal{T}$ consists of parallel sentences $(\boldsymbol{f}, \boldsymbol{e}) \in (\mathcal{X}, \mathcal{Y}(\boldsymbol{f}))$.[3] Here $\boldsymbol{f}$ is the source sentence, coming from the set of all possible source sequences, $\mathcal{X}$, and $\boldsymbol{e}$ is the target sentence, coming from the set of all possible translations of $\boldsymbol{f}$, $\mathcal{Y}(\boldsymbol{f})$.

Let $\boldsymbol{f}$ be a sequence of words $f_1, f_2, ...f_j$, and analogously $\boldsymbol{e}$, consists of words $e_1, e_2, ...e_k$. Thus, given $\boldsymbol{f}$, an SMT system needs to produce a translation $\boldsymbol{e}$ that is both **faithful** to the original source (alternatively *accurate* or *adequate*), and **fluent** in the target language.

2.2.1 Language Model

Language models are a core component of every translation system, necessary for evaluating the *fluency* aspect of translation. They tell us how good a sentence is in the target language by evaluating how probable each sequence of words is and assigning it a score. Typically n-gram language models are used, which use the Markov independence assumption to break up a long sequence of words, such that only the previous context of n-1 words is considered. The probability of the 5-word sequence $e_1...e_5$ with a trigram

[2] The bitext is usually accompanied by a large monolingual corpus used to build the language model, described below.

[3] For this introduction, we use the standard notation introduced by Brown et al. (1993) to represent the source sentence as $\boldsymbol{f}$ for French (or foreign) and the target as $\boldsymbol{e}$ for English.

language model (where n=3), and the general form for a word sequence of length $|\boldsymbol{e}|$, are:

$$\begin{aligned} p(e_1...e_5) &= p(e_1)p(e_2|e_1)p(e_3|e_1,e_2)p(e_4|e_2,e_3)p(e_5|e_3,e_4) \\ p(\boldsymbol{e}) &= \prod_{i=1}^{|\boldsymbol{e}|} p(e_i|e_{i-n+1}...e_{i-1}) \end{aligned} \tag{2.1}$$

Although the independence assumption constrains the length of possible word relationships, it also allows efficient estimation of language model probabilities from large monolingual corpora.[4]

2.2.2 Translation Model

The faithfulness, or adequacy aspect, is the responsibility of the translation model. The translation model we employ contains 1) the set of **rules** or correspondences that are necessary to produce $\boldsymbol{e}$ from $\boldsymbol{f}$ and 2) a **scoring function** that allows us to judge the quality of competing translations. From a high level, it is the mechanism by which we break $\boldsymbol{f}$ into pieces and transform it into $\boldsymbol{e}$.

The translation model breaks down the translation process into several steps: segmenting the source sentence into spans according to the translation rules being used, translating each source span into the target vocabulary by applying the appropriate translation rule, and scoring the different complete translation hypotheses.

Translation models are largely distinguished by how they define the mapping between source and target, i.e. translation equivalence, which is constrained by the set of

[4] As there will be many words sequences that have not occurred in the training data, typically the probabilities will be smoothed by reserving some probability for unseen events (Kneser and Ney, 1995; Katz, 1987).

translation rules they use. The set of rules is automatically extracted from the training data to build a model of translation. Initially the translation models introduced by IBM consisted of increasingly complex word-based models (Brown et al., 1993), known collectively as IBM Models 1 through 4. However, these only accounted for word-to-word correspondences. As increased context might help relieve problems of ambiguity and reordering between languages, it would be beneficial to have the smallest atomic unit of translation be larger than single words.

Phrase-based models (§2.2.3) were introduced to allow the translation of continuous multi-word sequences with one rule. While a much more powerful formalism than word correspondences, phrase-based models lacked the ability to model the hierarchical structure of language. Thus, hierarchical phrase-based models (§2.2.4) extended phrase-based models by allowing rules to represent phrases that can be reordered recursively. These are the two dominant formalisms today.[5]

While we will be using hierarchical phrase-based models in this thesis, as they are an extension of phrase-based models, elucidating the latter will be essential to understanding the strengths and weaknesses of the former, and will directly tie into the challenges that we will encounter later in this chapter when decoding and learning with these models.

2.2.3 Phrase-Based Models

In the first formalism, phrase-based translation (Koehn et al., 2003; Och and Ney, 2004), translation equivalence is modeled using pairs of contiguous word sequences, re-

[5]There are also other approaches to machine translations that are active areas of research: string-to-tree (Galley et al., 2006; Shen et al., 2010), tree-to-string (Huang et al., 2006), and tree-to-tree (Liu et al., 2009).

ferred to as **phrases**. These phrases need not correspond to any syntactic notion of phrasal structure; rather they are merely word sequences that tend to cooccur together in the corpus. This captures the intuition that sequences of words should often translate as a single unit, thereby eliminating potential causes of error which come from translating every word individually and then having to order them correctly. Therefore, in phrase-based models, **phrase pairs** are the representation of the model's rules on correspondences.

In order to learn phrase pairs automatically from the training corpus, we need to know which word sequences $\overline{\boldsymbol{f}} = f_1, ...f_j$ on the source side align with which target word sequences $\overline{\boldsymbol{e}} = e_1, ...e_k$. However, since all we know from our training data is that $\boldsymbol{e}$ is a translation of $\boldsymbol{f}$, but not which words in $\boldsymbol{e}$ are translations of which words in $\boldsymbol{f}$, this poses a problem. Before we can compute a phrasal alignment and extract phrase pairs $(\overline{\boldsymbol{f}}, \overline{\boldsymbol{e}})$, we must first compute a word alignment.

A **word alignment** marks word-to-word correspondences between $\boldsymbol{f}$ and $\boldsymbol{e}$ with alignment links. Since the word alignments are not observable, we induce them automatically from the parallel corpus, typically using the expectation-maximization algorithm, and include them in our model as a **latent** variable (Dempster et al., 1977; Och and Ney, 2003). As word alignments are typically induced in an unsupervised manner, they often result in noisy and problematic word correspondences.[6] Since phrase-based and hierarchical models will depend on these word alignments to extract translation rules, these errors will be propagated further, resulting in problems we discuss below.

Our training sentence pairs become tuples of $(\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{a}) \in (\mathcal{X}, \mathcal{Y}(\boldsymbol{f}), \mathcal{D}(\boldsymbol{f}, \boldsymbol{e}))$, where

[6]While there has been work on discriminative word-alignment, by far the vast majority of systems are built with generative alignment models.

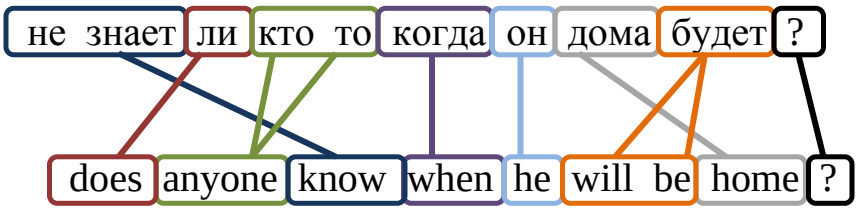


Figure 2.1: Example of word-to-word alignment links between a Russian and English sentence pair.

$\boldsymbol{a} \in \mathcal{D}(\boldsymbol{f}, \boldsymbol{e})$ represents an alignment between words in $\boldsymbol{f}$ and $\boldsymbol{e}$. It is important to point out here that there can be *exponentially* many alignments $\boldsymbol{a}$ for a sentence pair, with each $\boldsymbol{a}$ defining a possibly different set of translation equivalences between $\boldsymbol{f}$ and $\boldsymbol{e}$. As each $\boldsymbol{a}$ defines a translation $\boldsymbol{e}$, typically we will pair ($\boldsymbol{e}$,$\boldsymbol{a}$) together as the output of the translation model.

Figure 2.1 shows alignment links drawn between a Russian and English sentence pair. Looking at the alignment, we notice several phenomena which make machine translation difficult. First, we see two Russian words, кто то , aligning to a single word *anyone*, and a single Russian word будет , aligning to two English words *will be*. Second, we see links crossing each other, indicating that the relative order of the words must change during translation. Finally, the Russian negative particle не is unaligned on the target side, indicating that it has a NULL alignment.

Once we have obtained word alignments, we use them to constrain the possible phrase pairs ($\overline{\boldsymbol{f}}, \overline{\boldsymbol{e}}$) we can extract.[7] Specifically, the phrases need to be *consistent* with the word alignment. Formally, this is indicated by:

[7]Word alignments constrain phrase extraction since the fewer word alignments we have, the more phrases we can extract, and vice versa.

$$\begin{aligned} &\forall e_i \in \overline{\boldsymbol{e}} : (e_i, f_j) \in \boldsymbol{a} \Rightarrow f_j \in \overline{\boldsymbol{f}} \\ \text{AND } &\forall f_j \in \overline{\boldsymbol{f}} : (e_i, f_j) \in \boldsymbol{a} \Rightarrow e_i \in \overline{\boldsymbol{e}} \\ \text{AND } &\exists e_i \in \overline{\boldsymbol{e}}, f_j \in \overline{\boldsymbol{f}} : (e_i, f_j) \in \boldsymbol{a} \end{aligned} \tag{2.2}$$

Conceptually, this means that any words in $\overline{\boldsymbol{f}}$ that are aligned, must be aligned to words in $\overline{\boldsymbol{e}}$, and vice versa. Figure 2.2 presents an alternative view of the word alignment shown before, now as a two-dimensional grid, with $\boldsymbol{f}$ and $\boldsymbol{e}$ on the x and y axis, respectively. In this presentation, we can clearly visualize which phrase pairs are consistent with the alignment, and will thus be extracted. Graphically, extractable phrase pairs are those around which we can draw a rectangle and have no extraneous points in any of the four cardinal directions. Furthermore, while the latent word alignment $\boldsymbol{a}$ may contain NULL alignments, each latent phrase alignment is a nonempty one-to-one relationship. Phrases may be of different lengths, as in the case for both phrases marked in Figure 2.2. Theoretically, we may be able to learn larger phrases, memorizing whole chunks of sentences, but thus far, phrases beyond length 3 have not been shown to be useful (Koehn et al., 2003).

All extracted phrase pairs contained in a translation model are represented via a *phrase table*, which may look like the example shown in Table 2.1. Along with each phrase pair itself, the phrase table encodes statistics that can be used to score the phrase. As we will see in §2.2.4.1, the statistics constitute **features** of the phrase pair.

Formally, this model is equivalent to a finite-state transducer (FST) (Lopez, 2008), which has also proven successful in automatic speech recognition (ASR). While FST's are a powerful computational paradigm, unlike speech recognition, which is highly mono-

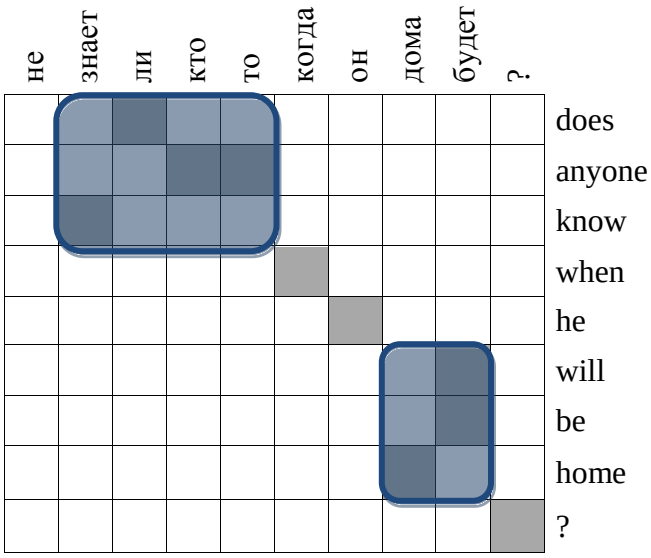


Figure 2.2: Two-dimensional grid showing word-to-word alignment between the same Russian and English sentence pair as Figure 2.1.

tonic, translation often exhibits local and long-distance reordering. This is problematic for FSTs, as they are meant for monotonic alignments, and thus have no efficient reordering mechanism. Thus, phrase-based models can only only handle local reordering, usually with a distance-based reordering cost. Even so, an exact search for the best translation is intractable (Koehn et al., 2003). Thus, approximate inference algorithms are employed for decoding, typically with a heuristic beam-search to extract the maximum weighted path (Koehn et al., 2003).[8] While phrase-based models have several drawbacks, chief among them being the inability to efficiently model non-local reordering, they nonetheless still power some of the most widely used translation systems.

[8] Although Rush et al. (2013) recently presented a beam-search based decoding algorithm based on Lagrangian relaxation that is capable of producing the optimal hypothesis.

Source	Target	Probability p($\overline{e}|\overline{f}$)
знает	know	0.7
дома будет	will be home	0.35
будет дома	will be home	0.45
будет	will be	0.82
знает ли кто то	does anyone know	0.23
когда он	when he	0.5

Table 2.1: Partial excerpt of the phrase translation table for the sentence pair in Figure 2.1.

2.2.4 Hierarchical Phrase-Based Models

Hierarchical phrase-based translation models (Chiang, 2005, 2007) were designed to capitalize on the notion that language is hierarchical by incorporating syntax into phrase-based translation.[9] In relation to phrase-based models, the **hierarchical phrase pairs** contain words as before, which we will now refer to as terminal symbols, but are now also allowed to contain *gaps* represented as a nonterminal variable. This nonterminal is a placeholder for another phrase. The principal benefit of this is to create **hierarchical** phrases, which can contain other phrases, which addresses two major problems of phrase based models: allowing recursive reordering and discontinuous phrases. By introducing these nonterminals into the grammar, such a system is able to utilize both word and phrase-level reordering to capture the hierarchical structure of language. This means that they can easily capture long-distance phrase reordering without incurring the intractability of phrase-based models.

Just as phrase-based models can be seen as instances of a finite-state formalism,

[9] Although not necessarily a *linguistic* syntax, in the sense of using constituents motivated by linguistic theory. Chiang (2007) discusses the difference between a system that is formally structured on a CFG, versus one that utilizes linguistic resources such as treebanks or lexicons.

hierarchical models are instances of context-free grammars (CFG). The phrase table is generalized in the resulting model as a grammar, which can be represented with a rule table.

Formally, hierarchical phrase-based models are synchronous context free grammars (SCFG). An SCFG is a generalization of monolingual CFG that allows for the simultaneous production of two output strings via a single synchronous *derivation*, where a derivation encodes the sequence of synchronous production rules $r \in \mathcal{G}$ used in translating the sentence. Each production rule r has the form

$$\mathrm{X} \longrightarrow \langle \boldsymbol{\gamma}, \alpha, \sim \rangle$$

where the left-hand side is always the single nonterminal X, and the right-hand side indicates an aligned sequence of terminals and nonterminals in the source γ and target α. The alignment of nonterminals in γ and α, which is a one-to-one correspondence, is given by $\sim$. This rule states that in a synchronous derivation, the expansion of X$\rightarrow \gamma$ on the source side happens synchronously with X$\rightarrow \alpha$ on the target. Note that $\sim$ can be empty, resulting in a traditional phrase pair translation rule. Thus, we are still able to have pairs of word sequences, the main advantage of phrase-based models, but now with the additional element of a recursive structure.

The grammar is represented as a rule table, a partial example of a which is given in Table 2.2.4. Much like a phrase table, it contains the rules themselves, along with features of those rules, described below. The first three hierarchical rules with one nonterminal present different possibilities of reordering the translation of *when* in relation to когда .

The next two rules have the maximum of two nonterminals, with the same source side, but different target reorderings. Notice that the last rule represents a phrasal translation, without any gaps.

$$
\begin{array}{lccc}
\mathrm{X} \longrightarrow & \langle\ \mathrm{X}_0\ \text{когда} & , & \mathrm{X}_0\ \text{when}\ \rangle \\
\mathrm{X} \longrightarrow & \langle\ \mathrm{X}_0\ \text{когда} & , & \text{when}\ \mathrm{X}_0\ \rangle \\
\mathrm{X} \longrightarrow & \langle\ \text{когда}\ \mathrm{X}_1 & , & \text{when}\ \mathrm{X}_0\ \rangle \\
\mathrm{X} \longrightarrow & \langle \mathrm{X}_0 \text{когда}\ \mathrm{X}_1 & , & \mathrm{X}_1\ \mathrm{X}_0\ \text{when}\ \rangle \\
\mathrm{X} \longrightarrow & \langle \mathrm{X}_0\ \text{когда}\ \mathrm{X}_1 & , & \mathrm{X}_0\ \text{когда}\ \mathrm{X}_1\ \rangle \\
\mathrm{X} \longrightarrow & \langle\ \text{когда} & , & \text{when}\ \rangle
\end{array}
$$

Table 2.2: Partial grammar for the sentence pair in Figure 2.1.

Hierarchical translation rules are extracted automatically from a word-aligned parallel corpus without the need for syntactic annotation. First, an initial phrase extraction is performed as described above, and all phrase rules $(\overline{\boldsymbol{f}}, \overline{\boldsymbol{e}})$ are added to the grammar $\mathcal{G}$. Next, we generalize our current rule set to include hierarchical phrases by taking any existing phrase pair $(\overline{\boldsymbol{f}}, \overline{\boldsymbol{e}}) \in \mathcal{G}$ contained inside another phrase, and replacing it with a nonterminal symbol X. This new hierarchical phrase pair is then added to $\mathcal{G}$, and the process is extended recursively, allowing rules with multiple nonterminal symbols.

Figure 2.3 graphically presents several such hierarchical phrase pairs, or equivalently, synchronous context-free rules, which could be extracted from our earlier example. Notice that the two parts of the sentence indicating a question, знает ли and *?*, are far away from each other, and thus could not be modeled in a single rule by a purely phrase-based model. However, by replacing corresponding word spans on both sides with a variable, we are able to capture a more general single rule containing both those terminal items. Furthermore, while phrase-based models would capture the word reordering of знает ли кто то to *does anyone know* by purely lexical means, the hierarchical rule

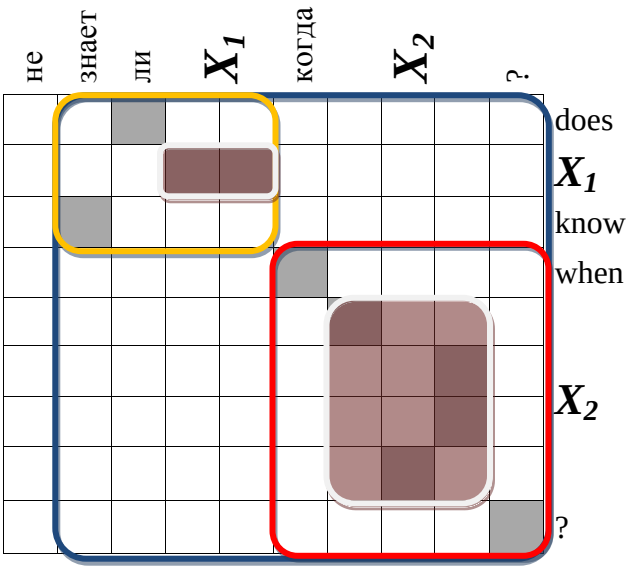


Figure 2.3: Two-dimensional grid showing possible initial phrase pairs, and example hierarchical rules demarcated by transparent rectangles with colored borders, for the Russian and English sentence pair in Figure 2.2. The red covering with white borders indicates a subphrase that was replaced by a nonterminal.

replaces the reordered part with co-indexed nonterminals, thereby allowing other fillers that fit the same pattern to take advantage of this reordering.

2.2.4.1 Features

For reasons we discuss later in this chapter, which relate largely to the historical models and optimization procedures employed, typically only a handful of features have been used in SMT systems. These are frequently occurring, i.e. *dense*, features, as they are present for all, or many, rules in the translation model. These features are commonly the log-probability of a previously learned generative model. Standard features include:

- target-to-source rule translation log $p(\overline{\boldsymbol{f}}|\overline{\boldsymbol{e}})$: relative frequency estimate of how often we see $\overline{\boldsymbol{e}}$ translated as $\overline{\boldsymbol{f}}$.

- source-to-target rule translation log $p(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$: relative frequency estimate of how often we see $\overline{\boldsymbol{f}}$ translated as $\overline{\boldsymbol{e}}$.

- target n-gram language model log $p(\boldsymbol{e})$: smoothed relative frequency estimate of how likely $\boldsymbol{e}$ is in the target language.

- source-to-target lexical translation log $p_{\text{lex}}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$: smooths the phrase probability by estimating probabilities of the individual words in $\overline{\boldsymbol{e}}$ given $\overline{\boldsymbol{f}}$.

- target-to-source lexical translation probability $p_{\text{lex}}(\overline{\boldsymbol{f}}|\overline{\boldsymbol{e}})$: smooths the phrase probability by estimating probabilities of the individual words in $\overline{\boldsymbol{f}}$ given $\overline{\boldsymbol{e}}$.

- 'pass-through' penalty: allows but penalizes passing the source word, since we might not know how to translate it, to the target side without translating it.

- Arity count: counts the number of times that rules with 0, 1, or 2 nonterminals were used.

- Total rule count: counts the total number of rules used in a derivation.

- Source word penalty: counts the number of source words.

- Target word penalty: controls target sentence length by adding a penalty for each target word.

- Glue rule: discourages piecemeal translation by concatenation.[10]

In later chapters, we will use this set of features for our experiments, where we may refer to it as either the baseline, low-dimensional, small, or dense feature set.

[10] We introduce glue rules in Section 2.2.5

2.2.5 Decoding

To **decode**, or, translate using a SCFG, we parse the source side by applying rules from $r \in \mathcal{G}$ that span parts of the source sentence, until the entire source has been covered. Because $\mathcal{G}$ contains *synchronous* rules, this simultaneously produces a corresponding parse structure on the target side whose yield is the target translation, which we can obtain by reading off the target terminal symbols.

A translation forest, or more formally a **hypergraph** (Gallo et al., 1993; Chiang, 2007), encodes the entire search space of the translation model. A hypergraph is a compact structure encoding exponentially many different translation strings $\boldsymbol{e} \in \mathcal{Y}(\boldsymbol{f})$ of the source sentence $\boldsymbol{f}$, along with many different **derivations** $\boldsymbol{d} \in \mathcal{D}(\boldsymbol{f}, \boldsymbol{e})$ that produce the same $\boldsymbol{e}$, where $\mathcal{D}(\boldsymbol{f}, \boldsymbol{e})$ represents the set of derivations $\boldsymbol{d}$ that yield translation string $\boldsymbol{e}$. Since we are only able to observe $\boldsymbol{e}$, we model $\boldsymbol{d}$ as a latent variable. Thus, derivations are the latent structure SCFG models produce during the translation process. The derivation $\boldsymbol{d} \in \mathcal{D}(\boldsymbol{f}, \boldsymbol{e})$ represents the particular set of rules $r \in \mathcal{G}$ used when producing $\boldsymbol{e}$ from $\boldsymbol{f}$, and is analogous to the alignment variable $\boldsymbol{a}$ in phrase-based models. An SCFG and a hypergraph are equivalent representations of a given parse forest. Figure 2.4 shows an example of a small hypergraph encoding alternate derivations of the hierarchical phrase-based translation of the sentence from Figure 2.1.

An alternative view to parsing for the decoding process is that of performing a composition of binary relations (Dyer, 2010b). As our further work will take advantage of established algorithms for performing efficient inference over hypergraphs, it is useful to make the connection explicit. The synchronous parse described above can be consid-

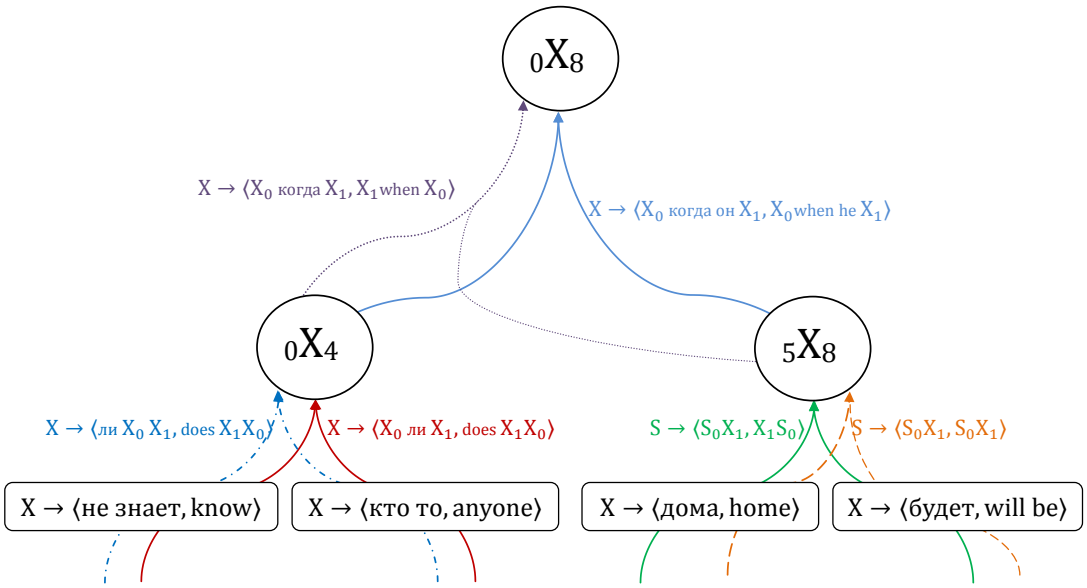


Figure 2.4: A small hypergraph encoding a partial SCFG for the sentence from Figure 2.1.

ered as computing the composition of the source sentence $\boldsymbol{f}$, represented as finite-state transducer, with the source side of the translation model SCFG $\mathcal{G}$, $\boldsymbol{f} \circ \mathcal{G}$, where $\circ$ denotes the composition operation. The result of this composition is another SCFG, $\mathcal{G}(\boldsymbol{f})$, which while being able to generate exponentially many translation strings $\boldsymbol{e}$ on the target side, can only generate $\boldsymbol{f}$ on the source side. We can further compose the target side of $\mathcal{G}(\boldsymbol{f})$ with a FST representation of the target string $\boldsymbol{e}$, $\mathcal{G}(\boldsymbol{f}) \circ \boldsymbol{e}$, to obtain another SCFG $\mathcal{G}(\boldsymbol{e}, \boldsymbol{f})$ which encodes all the derivations that exactly derive $\boldsymbol{f}$ on the source side and $\boldsymbol{e}$ on the target side. Dyer (2010b) presented a two-parse algorithm for performing this composition and showed that synchronous parsing could be performed without restricting the search space with heuristics. We will utilize this approach for decoding in later chapters.

While the parsing algorithm above allows us to decode exactly over an SCFG for the best translation, we have not yet factored in a crucial element of the process: the language model probabilities. In principle, composing the language model with $\mathcal{G}(\boldsymbol{f})$

can be carried out efficiently. We can treat the language model $\boldsymbol{l}$, as an FST where each state is a sequence of n-1 words, and compose the target side of $\mathcal{G}(\boldsymbol{f})$ with it to produce $\mathcal{G}(\boldsymbol{f}, \boldsymbol{l})$. However, this composition becomes prohibitively expensive to compute due to the necessary changes to the structure of the hypergraph that cause an explosion in the size of the resulting grammar $\mathcal{G}(\boldsymbol{f}, \boldsymbol{l})$. Thus, approximate algorithms are needed which prune the search space as they compose $\mathcal{G}(\boldsymbol{f})$ with $\boldsymbol{l}$ (Wu, 1996; Bar-Hillel et al., 1961). The most popular heuristic algorithm for performing this composition is *cube-pruning* (Huang and Chiang, 2007). Cube pruning is an efficient algorithm for integrating the LM with the complete hypergraph, which only looks at the top k subtranslations at each node (Chiang, 2007).

2.2.6 Causes of Ambiguity

The extraction mechanism outlined above allows the introduction of an exponential number of possible rules in a hierarchical phrase-based model. While it might seem advantageous to have the largest grammar possible in order to allow the greatest flexibility, this actually creates several important practical problems.

First, decoding complexity increases with the size of the grammar, as the search space grows to encode all the possible translation paths, and can become prohibitively slow. Second, and more important for our work, a larger rule set exacerbates the existing problems of **translation ambiguity** and **spurious ambiguity**.

Translation ambiguity results when the same source phrase in different contexts might have different meanings or senses, and thus can produce multiple target transla-

tion's. This might be caused by legitimate contextual differences, or be a by-product of noisy word alignments and rule extraction. Spurious ambiguity arises when many distinct derivations, which are the sequence of rule applications necessary to obtain a translation, produce the *same* translation string with the same features. We return to this problem in Chapter 6, but for now it suffices to say that both of these problems combine to create an enormous number of latent derivations for a given source sentence.

In order to create a more manageable grammar, hierarchical rule extraction is constrained in several ways. In a standard hierarchical phrase-based system, the grammar only contains one nonterminal symbol, X, which may appear no more than twice on either side of a rule, and never adjacently on the source side. This eliminates a major cause of spurious ambiguity (Chiang, 2007), while also reducing the size of the grammar from exponential to polynomial (Koehn, 2010). Furthermore, rules must always have at least one, but no more than N, typically with N=5, aligned words. There is also a special symbol S, which is used in so-called glue rules to combine rules together by concatenation:[11]

$$\begin{aligned} \mathrm{S} &\longrightarrow \quad \langle\, \mathrm{X}_0 \,,\, \mathrm{X}_0 \,\rangle \\ \mathrm{S} &\longrightarrow \quad \langle\, \mathrm{S}_0\, \mathrm{X}_1 \,,\, \mathrm{S}_0\, \mathrm{X}_1 \,\rangle \end{aligned}$$

[11] There has been interest for some time in enriching the SCFG model with a larger nonterminal vocabulary (Blunsom et al., 2008a; Zollmann and Venugopal, 2006). This would allow more expressive power for reordering, as well as introduce some context sensitivity into our *context-free* model. However this comes at the cost of increased complexity, both for rule extraction and decoding.

	Phrase-Based	Hierarchical
Translation equivalence	flat continuous word sequence	terminal and nonterminal sequence
Latent Variable	phrasal alignment	derivation
Packed representation	lattice	hypergraph
Decoding strategy	beam-search	cube-pruning

Table 2.3: Comparison of typical modeling and decoding methods in phrase-based and hierarchical models.

2.2.7 Translation Model Summary

Over the past 25 year, the field of machine translation has progressed from word-based, to phrase-based, to hierarchical phrase-based models of translation. Table 2.3 summarizes the relationships between the major modeling components of phrase-based and hierarchical models. In practice, both of these are based on a pipeline of components that propagate errors; beginning with data normalization (word segmentation, casing, etc.), to create word alignments, leading to rule extraction based on those word-alignments, and finally pruning the search-space we create from those rules. All these errors contribute to additional ambiguity in the translation process, a challenge for learning that we discuss in Section 2.4.

A key benefit of using the more powerful hierarchical phrase-based models is that they combine the strengths of phrase-based modeling with a representation of syntax, thus providing the ability to handle longer reorderings. Due to these advantages, and their proven state-of-the-art performance across many language pairs (Chiang, 2007), this is the translation formalism we use in this dissertation.

2.3 Model Parameterization

We have so far described the formalism through which our model views the translation process, and thus we can now enumerate the possible ways in which a source sentence can be translated. However, the packed representation of those possibilities, the hypergraph, encodes an enormous number of different translations, and we as yet cannot determine which of those will be a *good* translation. Since there are an exponential number of translations licensed by the rules of the translation model grammar, in order to select the desired translation, we need a mechanism for scoring the best translation higher than other possible hypotheses.[12] Model **parameterization** is the process by which we are able to efficiently represent such a score function. Following Brown et al. (1993), this is done by expressing the problem in a probabilistic framework and parameterizing the model such that we can compute a real-valued score for every pair of source and target translations, $p(\boldsymbol{e}|\boldsymbol{f})$. The decoding problem in SMT then becomes finding the translation that maximizes this probability:

$$\boldsymbol{e}^* = \underset{\boldsymbol{e} \in \mathcal{Y}(\boldsymbol{f})}{\operatorname{argmax}}\, p(\boldsymbol{e}|\boldsymbol{f}) \tag{2.3}$$

2.3.1 Generative Models

The first applications of machine learning to SMT were based on modeling translation using the **noisy-channel** paradigm (Brown et al., 1993), adapting an approach that

[12] We are deliberately being ambiguous as to what *best* means at the moment. This will be elucidated later.

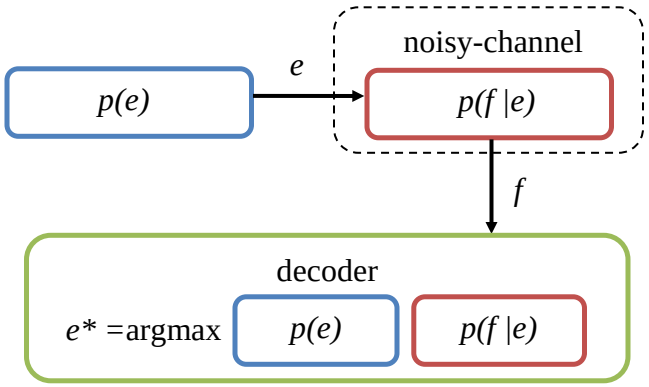


Figure 2.5: Graphical depiction of the noisy-channel model of SMT. The blue "source" model generates $\boldsymbol{e}$, which is then corrupted by the channel $p(\boldsymbol{f}|\boldsymbol{e})$ into the "target" $\boldsymbol{f}$. The decoder then uses these two models to find the best "source" $\boldsymbol{e}$ that generated $\boldsymbol{f}$. Notice that this notation inverts the definition of source and target languages.

had been successful in ASR (Bahl et al., 1983; Brown et al., 1990). This model supposes that the source sentence is actually a corrupted version of the target sentence. After the target sentence is generated according to the source model, $p(\boldsymbol{e})$, it is sent through the channel $p(\boldsymbol{f}|\boldsymbol{e})$ to produce $\boldsymbol{f}$. Figure 2.5 presents a depiction of the model. Following Weaver (1955), the process of recovering the original target $\boldsymbol{f}$ is referred to as decoding.

This model has the advantage of breaking the posterior probability in Eq. 2.3 into two independent probability models. By applying Bayes's Rule, we can decompose the posterior probability $p(\boldsymbol{e}|\boldsymbol{f})$ into:

$$\boldsymbol{e}^* = \underset{\boldsymbol{e}\in\mathcal{Y}(\boldsymbol{f})}{\operatorname{argmax}}\, p(\boldsymbol{e}|\boldsymbol{f}) = \underset{\boldsymbol{e}\in\mathcal{Y}(\boldsymbol{f})}{\operatorname{argmax}}\, \frac{p(\boldsymbol{f}|\boldsymbol{e})p(\boldsymbol{e})}{p(\boldsymbol{f})} \tag{2.4}$$

Since $\boldsymbol{f}$ is fixed for a given $\boldsymbol{e}$, and we do not necessarily require a proper probability distribution for $p(\boldsymbol{e}|\boldsymbol{f})$ if we only care about computing the argmax, we only need to have an estimate for two sets of parameters: $p(\boldsymbol{f}|\boldsymbol{e})$, our translation model (§2.2), containing

the rules discussed above, and $p(\boldsymbol{e})$, the language model (§2.2.1), encoding fluency in the target language.

These two distributions are further decomposed using the chain rule and independence assumptions. We have already seen how $p(\boldsymbol{e})$ is decomposed into a product of smaller probabilities. Similarly, $p(\boldsymbol{f}|\boldsymbol{e})$ is decomposed into a product of translation probabilities over each single translation decision that our model can make. For phrase-based models this means a parameter for each phrase pair, while in hierarchical translation it means one for each production rule. The parameters of these distributions can be estimated from the data in a number of ways, most commonly using relative frequencies based on the maximum likelihood criterion (Koehn, 2010).

2.3.2 Decision Rules

However, we cannot simply score the output translation $\boldsymbol{e}$, because the translation is actually the yield produced by the rule applications encoded in the latent structure, $\boldsymbol{d} \in \mathcal{D}(\boldsymbol{f}, \boldsymbol{e})$ during the translation process. Thus, it is actually to each *derivation* that we must assign a score. Many different derivations may produce the same translation, and although our models are actually defined over derivations, since constructing a derivation produces a translation, and we only observe the output sentence $\boldsymbol{e}$, derivations will always be paired with translations, $(\boldsymbol{e}, \boldsymbol{d})$, and will be included in our probability model: $p(\boldsymbol{e}, \boldsymbol{d}|\boldsymbol{f})$. Thus, Eq. 2.3 becomes:

$$(\boldsymbol{e}^*, \boldsymbol{d}^*) = \underset{\boldsymbol{e} \in \mathcal{Y}(\boldsymbol{f}), \boldsymbol{d} \in \mathcal{D}(\boldsymbol{f}, \boldsymbol{e})}{\operatorname{argmax}} p(\boldsymbol{e}, \boldsymbol{d}|\boldsymbol{f}) \tag{2.5}$$

This is the problem of jointly finding the maximum probability derivation and translation. Since we are actually interested in obtaining $p(\boldsymbol{e}|\boldsymbol{f})$, we can treat $\boldsymbol{d}$ as a nuisance variable and marginalize over it to obtain the best translation:

$$\boldsymbol{e}^* = \underset{\boldsymbol{e} \in \mathcal{Y}(f)}{\operatorname{argmax}}\, p(\boldsymbol{e}|\boldsymbol{f}) = \underset{\boldsymbol{e} \in \mathcal{Y}(f)}{\operatorname{argmax}} \sum_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{f}, \boldsymbol{e})} p(\boldsymbol{e}, \boldsymbol{d}|\boldsymbol{f}) \tag{2.6}$$

Equation 2.6 presents the problem of finding the **maximum probability translation**, which amounts to using the maximum a posteriori (MAP) decision for $\boldsymbol{e}^*$.

How we select the best hypothesis is known as the **decision rule**. Unfortunately, decoding with the maximum probability translation decision rule proves to be NP-hard (Sima'an, 1996). For this reason, most systems address the decoding problem by approximating the maximum probability translation with the **maximum derivation** using Eq. 2.5 (Koehn, 2010; Lopez, 2007). Using the maximum derivation decision rule is also known as **Viterbi decoding**, since we are selecting the most probable hyperpath through the hypergraph (Viterbi, 1967). In effect, the probability of a translation $\boldsymbol{e}$ is approximated by the probability of $\boldsymbol{e}$'s best scoring derivation.

The maximum derivation in a hypergraph can be computed exactly in polynomial time using dynamic programming. It is an approximation insofar as the hypergraph itself has been pruned after language model integration, thus the derivation we are selecting is likely not going to be the same as the best derivation we would have selected if we had the unpruned hypergraph. It is also worth noting that we can also find the *total* probability of all derivations in a hypergraph using dynamic programming.[13]

[13]The INSIDE algorithm assumes the hypergraph is acyclic, a condition that is met by our hypergraphs.

Another common decision rule is minimum Bayes risk (MBR), which tries to minimize expected loss according to a given external cost function Δ (Kumar and Byrne, 2004).

$$\boldsymbol{e}^*_{risk} = \operatorname*{argmin}_{\boldsymbol{e} \in \mathcal{Y}(f)} \sum_{\boldsymbol{e}' \in \mathcal{Y}(f)} \Delta(\boldsymbol{e}, \boldsymbol{e}') p(\boldsymbol{e}|\boldsymbol{f}) \tag{2.7}$$

For most of this dissertation, we will use the maximum derivation approximation in Eq. 2.5. However, as ideally we would like to train and decode toward the maximum probability translation, thereby not ignoring most of the translation space, several approximate inference algorithms for decoding in hierarchical models have been developed (Blunsom and Osborne, 2008; Blunsom et al., 2008b; Li et al., 2009). This will also be the subject of Chapter 6.

In addition to find the best derivation, we may be interested in finding the k-best derivations. These are often used in training to approximate the entire hypergraph search space with a finite set of derivations. We extract k-best approximations for training using the method developed by Huang and Chiang (2005).

2.3.3 Discriminative Models

The noisy-channel generative model was supplanted around a decade ago in favor of discriminative models (Och and Ney, 2002; Och, 2003), which allow the ability to optimize parameters toward an arbitrary external error function. Thus, discriminative models allow moving away from the maximum-likelihood criterion, which the noisy-channel model was based on, to a translation quality metric.

Beyond this ability they relax the independence assumptions imposed by generative models, and allow integration of expressive and non-independent features for translation. The ability of discriminative models to define novel features is an important aspect for this thesis, as work within the noisy-channel modeling paradigm was restricted to the translation and language model features. In the last five years, there has been a surge in exploration of features for use in SMT (Lopez, 2008; Chiang et al., 2008a, 2009; Simianer et al., 2012; Green et al., 2013).

The forerunner of current discriminative modeling for SMT, and still the most widely used parametrization today, is the log-linear model presented by Och and Ney (2002):

$$(\boldsymbol{e}^*, \boldsymbol{d}^*) = \underset{\boldsymbol{e}\in\mathcal{Y}(\boldsymbol{f}),\boldsymbol{d}\in\mathcal{D}(\boldsymbol{f},\boldsymbol{e})}{\operatorname{argmax}} p(\boldsymbol{d},\boldsymbol{e}|\boldsymbol{f}) \tag{2.8a}$$

$$= \underset{\boldsymbol{e}\in\mathcal{Y}(\boldsymbol{f}),\boldsymbol{d}\in\mathcal{D}(\boldsymbol{f},\boldsymbol{e})}{\operatorname{argmax}} \frac{\exp\sum_i^n \mathrm{w}_i \mathrm{f}_i(\boldsymbol{f},\boldsymbol{e},\boldsymbol{d})}{\sum_{\boldsymbol{e}\in\mathcal{Y}(\boldsymbol{f}),\boldsymbol{d}\in\mathcal{D}(\boldsymbol{f},\boldsymbol{e})}\exp\sum_i^n \mathrm{w}_i \mathrm{f}_i(\boldsymbol{f},\boldsymbol{e},\boldsymbol{d})} \tag{2.8b}$$

$$= \underset{\boldsymbol{e}\in\mathcal{Y}(\boldsymbol{f}),\boldsymbol{d}\in\mathcal{D}(\boldsymbol{f},\boldsymbol{e})}{\operatorname{argmax}} \exp\sum_i \mathrm{w}_i \mathrm{f}_i(\boldsymbol{f},\boldsymbol{e},\boldsymbol{d}) \tag{2.8c}$$

$$= \underset{\boldsymbol{e}\in\mathcal{Y}(\boldsymbol{f}),\boldsymbol{d}\in\mathcal{D}(\boldsymbol{f},\boldsymbol{e})}{\operatorname{argmax}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{f},\boldsymbol{e},\boldsymbol{d}) \tag{2.8d}$$

where, as opposed to Eq. 2.4, we model the conditional probability $p(\boldsymbol{d},\boldsymbol{e}|\boldsymbol{f})$ directly. The model is parameterized by a real-valued n-dimensional **parameter vector** $\mathbf{w}$. Each parameter w_i is associated with a corresponding feature $\mathrm{f}_i(\boldsymbol{f},\boldsymbol{e},\boldsymbol{d})$ from a **vector of feature functions** $\mathbf{f}(\boldsymbol{f},\boldsymbol{e},\boldsymbol{d})$. The score of each translation and derivation pair $(\boldsymbol{e},\boldsymbol{d})$ can then be computed as the dot product between the parameter and feature vectors, where

the weight determines the contribution of each feature to the final score:

$$\text{score}(\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{d}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{d}) \tag{2.9}$$

Since oftentimes we will not require the score to be a proper probability, we can reduce the log-linear model in Eq. 2.8b to the *linear* scoring function in Eq. 2.9. Thus, we will refer to Eq. 2.9 as the score of $\boldsymbol{d}$ without implying any probabilistic interpretation.

Feature functions $\mathbf{f}(\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{d})$ represent knowledge sources and salient aspects of the translation process. For instance, the translation and language models we encountered earlier, $p(\boldsymbol{f}|\boldsymbol{e})$ and $p(\boldsymbol{e})$, remain two of the primary features employed. In fact, notice that if we restrict our feature set to these two with uniform $\mathbf{w}$, we recover the noisy channel model.

While features can be very expressive, they must conform to certain restrictions. Most importantly, features must decompose over rules in the derivation, such that we can compute the global score of a derivation $\boldsymbol{d}$ using local features h on each rule of the derivation $r \in \boldsymbol{d}$.

$$\text{score}(\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{d}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{d}) = \sum_{r \in \boldsymbol{d}} \mathbf{w}^\top \mathrm{h}(r) \tag{2.10}$$

This implies that every hyperedge in the hypergraph also has a score assigned to it based on the local feature functions. When performing inference over the hypergraph, each edge corresponds to a partial translation and has associated with it a decomposable score which is comprised of the features on that edge and the current weight vector. These can then be

summed together to produce the score associated with a specific hypothesis translation. If a feature does *not* decompose neatly over rules, such as the language model, we call it a **stateful** feature. Such features necessitate splitting the states in the hypergraph in order to incorporate them, and can thus create practical problems due to the resulting complexity of the search space and ensuing search error.

We focus on the problem of developing feature functions that improve translation in Chapter 5.

Since statistical models represent knowledge in the form of features, different weight settings give precedence to the contribution of different features to the final score. Thus, properly setting the weights in order to emphasize the important feature contributions for scoring alternative translations is an integral part of constructing a state-of-the-art SMT system. The problem of parameter estimation, or tuning the model to obtain optimal weight settings is the subject of Chapters 3, 4 and 6.

2.4 Learning

As described in Section 2.3.3, discriminative models have superseded generative models, and have become the standard in SMT, due primarily to their ability to allow independent feature functions and arbitrary optimization criteria.

Now that the formalism for the translation model is settled (hierarchical phrase-based modeling (§2.2)), and there is a parameterization of the probability model (linear (Eq. 2.8d)), we turn to the problem of optimally setting the parameters of $\mathbf{w}$ in Eq. 2.8d for $\mathbf{f}(\boldsymbol{f}, \boldsymbol{e}, \boldsymbol{d})$ in order to correctly rank the model's translation hypotheses according to

how good they actually are. The goal of training then is to **estimate**, or **optimize** the model parameters $\mathbf{w}$ to maximize a given objective.[14]

In this section, we first introduce the necessary formal foundations of learning theory, and then elucidate the adaptation of machine learning to the SMT problem. Since a complete review of machine learning and its application to SMT is beyond the scope of this thesis, we will focus the discussion on the major approaches to learning. For a thorough introduction to machine learning, we refer the reader to Bishop (2006) and Mitchell (1997).

2.4.1 Background

One of the core applications of machine learning is to produce some intelligent behavior, but without having to specify the mechanism for that behavior, only the product. For SMT, this means that instead of analyzing the process of how humans translate, and getting experts to specify translation rules, we should instead focus our attention on creating algorithms that learn from examples of what good translation looks like.

More generally, we want our algorithms to learn from experience. This means figuring out and remembering the correct answer for the examples they have seen, and doing it in such a way as to then be able to correctly predict the answer to previously unseen examples. Designing learning algorithms that are able to efficiently learn the correct predictions and then generalize well to unseen data is a core challenge in machine learning.

The field of machine learning has been making steady advances over recent years,

[14] We could alternatively minimize a given loss, since the two are complementary.

both in theory and practice, and advances in machine learning have led directly to advances in statistical NLP. However, adapting existing machine learning algorithms to SMT is often complicated, due to the differences between SMT and other tasks. This will be discussed in Section 2.4.3.

2.4.2 Structured Prediction

Formally, prediction, or classification, involves problems where given a training set $\mathcal{T}$ of examples $(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y}$ drawn from an unknown but fixed distribution $t(\boldsymbol{x}, \boldsymbol{y})$, we want to learn a discriminant function $f_{\mathbf{w}} : \mathcal{X} \rightarrow \mathcal{Y}$ that is capable of mapping an input $\boldsymbol{x}$ to a discrete output $\boldsymbol{y}$. The *hypothesis* space is the set of functions $f_{\mathbf{w} \in \mathbf{W}}$ mapping $\boldsymbol{x}$ to $\boldsymbol{y}$, which is parameterized by $\mathbf{w}$, whose values are set by the learning algorithm. In general terms, the learning algorithm searches the hypothesis space to find $f_{\mathbf{w}}$ that optimizes some criterion ℓ, such as the large-margin or conditional likelihood criteria. The most difficult parts of the learning problem are then how to represent the $\boldsymbol{x}$ and $\boldsymbol{y}$, and selecting an appropriate loss function ℓ.

Since we have output labels for all our inputs, prediction is a form of *supervised* learning. The most common form of prediction in machine learning is binary classification, where the learner has to choose one of two answers. In that case, $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^n \times \{\pm 1\}$ and $f_{\mathbf{w}} : \mathcal{X} \rightarrow \{\pm 1\}$.

While binary classification is a fundamental problem, accounting for much of the research into classification, the output space is not rich enough to handle many problems in statistical NLP. Multiclass classification extends binary classification by allowing $\mathcal{Y}$ to

be a finite set of size k, where k is greater than two. However, even that setting is still not capable of handling the exponentially sized output space that we need to deal with in SMT.

Structured prediction further generalizes multiclass classification, and involves problems where we want to map $\boldsymbol{x}$ to a $\boldsymbol{y}$ that has a rich internal structure. Now $\mathcal{Y}$ is an exponentially large set of possible output structures. In NLP, examples of structured prediction include sequence labeling, where $\boldsymbol{y}$ is a sequence of labels, and parsing, where $\boldsymbol{y}$ is a parse tree (Taskar et al., 2005; Tsochantaridis et al., 2004). SMT can also be seen as an instance of structured prediction, where $\boldsymbol{y}$ is the translation.

For structured prediction, we also introduce an auxiliary evaluation function

$$g_{\mathbf{w}} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R},$$

capable of taking $\boldsymbol{x}$ and a proposed $\boldsymbol{y}$ as input and assessing the quality of $\boldsymbol{y}$:

$$f_{\mathbf{w}}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} g_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y}) \tag{2.11}$$

The structured prediction models we consider represent this function as a linear model, $\mathbf{w}^\top \mathbf{f}(x, y)$, where $\mathbf{w}$ are the model parameters and $\mathbf{f}(x, y)$ is a set of feature functions over $\boldsymbol{x}$ and $\boldsymbol{y}$. This weighted linear combination produces a score:

$$g_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \text{score}(\boldsymbol{x}, \boldsymbol{y}) \tag{2.12}$$

and the goal is to learn $\mathbf{w}$ such that for a given feature vector representation of $\boldsymbol{x}$, the

correct $\boldsymbol{y}$ scores higher than any incorrect $\boldsymbol{y}'$. Then, given a previously unseen $\boldsymbol{x}$, we can predict the optimal $\boldsymbol{y}^*$ as:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}}{\operatorname{argmax}} \operatorname{score}(\boldsymbol{x}, \boldsymbol{y}) \tag{2.13}$$

Finally, in order to find the best $\mathbf{w}$, we need to define a loss function to optimize. Since we are dealing with complex structures, we need to be able to articulate the fine-grained distinctions between possible outputs. Some mistakes will be notably worse than others. Thus, we assume a task-specific cost function $\Delta(\boldsymbol{y}, \boldsymbol{y}') = \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, which quantifies how bad predicting $\boldsymbol{y}'$ is when the true label is $\boldsymbol{y}$. By definition, if $\boldsymbol{y}$=$\boldsymbol{y}'$, $\Delta(\boldsymbol{y}, \boldsymbol{y}') = 0$, while it increases the worse $\boldsymbol{y}'$ becomes. Instances of $\Delta(\boldsymbol{y}, \boldsymbol{y}')$ are the Hamming loss for sequence labeling (Taskar et al., 2005), and F-score for parsing (McDonald et al., 2005). The exact form of the cost function we use is discussed in Section 2.4.4.

With this in hand, we can say that we want to select the $f_{\mathbf{w}}$ that minimizes the expected loss, under the true distribution of our training data $\mathcal{T}$, $t(\boldsymbol{x}, \boldsymbol{y})$:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathrm{W}}{\operatorname{argmin}} \mathbb{E}_{t(\boldsymbol{x}, \boldsymbol{y})} \Delta(\boldsymbol{y}, f_{\mathbf{w}}(\boldsymbol{x})) \tag{2.14}$$

Eq. 2.14 is referred to as Bayes risk minimization (Bishop, 2006). However, recall that $t(\boldsymbol{x}, \boldsymbol{y})$ is unknown, thus making the direct minimization of Eq. 2.14 impossible. **Structural risk minimization** (Vapnik, 1995), however, can be used as a stochastic approximation, as it is based on the observed distribution of $\mathcal{T}$, and minimizes the regularized empirical risk:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w} \in \mathrm{W}} \mathcal{R} + \frac{C}{N} \sum_{i=1}^{N} \Delta(\boldsymbol{y}_i, f_{\mathbf{w}}(\boldsymbol{x}_i)) \tag{2.15}$$

where the second term approximates the expected loss, and the first term $\mathcal{R}$ is a regularizer to prevent overfitting. $\mathcal{R}$ penalizes complex solutions, and usually takes the form of the ℓ_1 or ℓ_2 norm. Unfortunately, as directly minimizing Eq. 2.15 for a linear model is still usually impractical, most techniques instead focus on defining and minimizing an alternative loss function ℓ that forms a convex upper-bound (Zhang, 2003). Thus, the final form of our optimization problem is:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w} \in \mathrm{W}} \mathcal{R} + \frac{C}{N} \sum_{i=1}^{n} \ell_{\mathbf{w}}(\boldsymbol{x}_i, \boldsymbol{y}_i) \tag{2.16}$$

where $\ell_{\mathbf{w}}$ can be instantiated by well known loss functions, such as conditional log-likelihood (CRF (Lafferty et al., 2001)), or margin maximization (Max-Margin Markov Networks, SVM (Taskar et al., 2004; Joachims, 1998)). The learning frameworks which we discuss below are derived from regularized risk minimization.

2.4.3 Structured Prediction à la SMT

There are a few important points of divergence between standard structured prediction problems, even in NLP, and structured prediction for SMT, which introduce unique challenges. These differences necessitate the adaptation of standard machine learning algorithms, and are in large part the motivating factor behind the work presented in this dissertation. The first problem is related to the output space $\mathcal{Y}$, and our model's ability to generate hypotheses in that space, while the second problem is related to the derivation

our model produces which necessitates the inclusion of a latent structure $\boldsymbol{d}$ in our model.

While traditionally most problems in supervised learning posit a single correct output,[15] by the inherent nature of translation trying to pass the meaning of a sentence from one symbolic representation to another, there are a multitude of possibilities for what the **reference** translation can be.[16]

This is actually two separate but equally troubling problems: on the one hand, we want *fewer* reference translations, on the other hand, we want *more*! Most evaluation corpora for SMT have from one to four translations. If we have multiple references, then we may want to select one of them to treat as the correct one, but which one? While one translation may be more fluent, another may be more easily attainable by the model, and a third might induce the best set of derivations to generalize to new examples. Thus, systems usually try to leverage all of available references.

However, as there are on average many more ways of translating a sentence than we can manually enumerate, our reference set is still missing most of the possible references.[17] This is problematic not only because we are missing potential sources of information, but more importantly, because our model might not be able to derive the references we do have at all.

Recall that our translation model is based on the SCFG formalism, where we use a parallel training corpus to extract all the possible translation rules which we can use.

[15] In general, the problem of multi-label classification, where each input is associated with a set of correct labels (Tsoumakas and Katakis, 2007), has been increasingly popular. Although typical multi-label techniques are inapplicable to multi-label structured prediction, for the same reasons as standard multi-class prediction is inapplicable to structured prediction, recent work on multi-label structured prediction has shown that efficient techniques can be developed (Lampert, 2011).

[16] Reference translations are stipulated to be the correct translations provided by a human translator.

[17] This is an ongoing research area, with attempts being made to manually create reference lattices, or generate a broader set of references via automatic paraphrases (Madnani, 2010).

Due to a number of factors mentioned above, including a cascade of errorful models used to align and extract the grammar, as well as the beam-search or pruning mechanism used to decode, we may encounter a sentence for which the decoder may not be able to generate translations exactly matching the reference. Causes of this include the grammar not containing the necessary rules to produce the reference translation, or their being pruned out during the search. The effect is that the reference becomes **unreachable**: $\boldsymbol{y}_i \notin \mathcal{Y}(\boldsymbol{x}_i)$. Most structured prediction problems in machine learning assume $\boldsymbol{y}_i \in \mathcal{Y}(\boldsymbol{x}_i)$. Take for example the following simple grammar:[18]

Rule 1 |	X $\longrightarrow$ ⟨ morgen , tomorrow ⟩
Rule 2 |	X $\longrightarrow$ ⟨fliege ,will fly ⟩
Rule 3 |	X $\longrightarrow$ ⟨X_1 ich , I X_1 ⟩
Rule 4 |	X $\longrightarrow$ ⟨ich , I ⟩

source: morgen fliege ich
reference: tomorrow I will fly

Given a source sentence *morgen fliege ich*, with reference translation *tomorrow I will fly*, we can apply Rules 1-2 to translate *morgen fliege* to *tomorrow will fly*, and then reorder *fliege ich* to *I will fly* using Rule 3, and our output, *tomorrow I will fly*, matches the reference. Suppose, however, that the grammar we have extracted lacks Rule 3, and we only have Rules 1, 2 and 4. Then our output becomes *tomorrow will fly I*. In this case, the reference is unreachable, since our system cannot construct any output that matches the true reference.

This is problematic because in order to perform discriminative training in many regimes, we need to be able to produce and score the correct output structure. If $\boldsymbol{y}_i \notin$

[18] Example sentence adapted from Koehn (2010).

$\mathcal{Y}(\boldsymbol{x}_i)$, this precludes the construction of its feature vector to use when updating our parameters. Thus, we need to rely on an approximation of the reference which our model *can* generate.

A major question for most linear discriminative machine translation models is therefore what to update $\mathbf{w}$ towards, i.e. what to use as the oracle or surrogate reference output $\boldsymbol{y}^+ \in \mathcal{Y}(\boldsymbol{x}_i)$, in place of the real reference $\boldsymbol{y}_i$ if it is unreachable, since it is this output's feature vector we will use when computing the loss function. As a look ahead, the procedure to select $\boldsymbol{y}^+$ will require **cost-augmented inference** (Smith, 2011). We explore surrogate reference selection and cost-augmented inference in Chapter 3.

The second problem is that while the structured objective, the derivation, that produces each translation is unobserved and must be modeled as a latent variable, most losses examined in machine learning do not posit latent variables. In principle, latent variables do not complicate the formulation, since they are simply added as an additional unobserved output variable, and modeled jointly with the observed output. However, they complicate the learning, since their presence generally makes the loss function ℓ non-convex (Nowozin and Lampert, 2011). Convexity provides nice theoretical guarantees (Boyd and Vandenberghe, 2004), and non-convexity means that our optimization may not find the global optimum. When compounded with the cost-augmented inference we must perform to select $\boldsymbol{y}^+$, we end up with latent structured loss functions that are not yet well established in machine learning (McAllester and Keshet, 2011; Gimpel and Smith, 2012).

2.4.4 Evaluation Metric

Up until this point, we have assumed that an external automated metric for judging MT quality is readily available. However, the development of such a metric is quite an undertaking in of itself (Macháček and Bojar, 2013).

Human evaluation of MT quality has traditionally required trained specialist's to rank each sentence on a scale of 1-5 for adequacy and fluency (White et al., 1993). Understandably, this process is both laborious and expensive. With the development of crowd sourcing, human MT evaluation has taken on a new life. Using crowdsourcing platforms such as Amazon's Mechanical Turk, Zaidan and Callison-Burch (2010) were able to obtain reasonably good quality judgments relatively cheaply. While improving the cost, this process is still not capable of producing the quick and repeated judgments necessary for system development. For training and evaluating changes to our system, we need to repeatedly evaluate thousands of sentences on the order of seconds. Automated metrics will likely never be as informative as actual human evaluation, but we must rely on them.

The main principle behind automated metrics is that they should be closely correlated with human judgments, commonly evaluated by human-targeted translation edit rate (HTER): the fewest number of edits a human has to make to the output. In order to do that, they usually make use of the reference translations available for our training data, and compute some sort of similarity score between reference(s) and the system output. In order to choose between metrics, we can break the criteria into two pieces: how well we tune when we use that metric, and how much the evaluation proposed by that metric correlates with HTER. While for evaluation, other proposed metrics have shown higher

correlation with HTER (Macháček and Bojar, 2013), the most commonly used metric for tuning and evaluation is the bilingual evaluation understudy (BLEU) (Papineni et al., 2002a).

BLEU is a precision-oriented metric, and evaluates the output translation h against the reference translation(s) r by considering the number of n-gram matches between them, where n usually runs from 1 to 4. The number of matches is then divided by the total number of possible n-grams in the hypothesis output. To compute the BLEU score of our system's outputs $\{\boldsymbol{y}'\}_1^n$ against the reference translations $\{\boldsymbol{y}\}_1^n$, we first compute n-gram precision matches on a per-sentence basis as follows:

$$p_j = \frac{\sum_{i=1}^{n} \sum_{g \in \text{grams}(j)} \text{count}_{\text{clip}}(g, \boldsymbol{y}'_i)}{\sum_{i=1}^{n} \sum_{g \in \text{grams}(j)} \text{count}(g, \boldsymbol{y}'_i)} \tag{2.17}$$

where $\text{count}(g, \boldsymbol{y}'_i)$ is the count of an n-gram g in hypothesis $\boldsymbol{y}'_i$, and $\text{count}_{\text{clip}}(g, \boldsymbol{y}'_i)$ is the clipped n-gram count, representing the maximum number of times any reference contains g. Clipping is used to not give any hypothesis credit for producing an n-gram more often than it appears in the reference. Combining n-gram precisions p_j of different length using the geometric mean, we can compute the BLEU score as:

$$\log \text{BLEU} = \underbrace{\min\left(1 - \frac{|r|}{|h|}, 0\right)}_{\text{brevity penalty}} + \underbrace{\sum_{i=j}^{4} \frac{1}{4} \log p_j}_{\text{geometric mean}} \tag{2.18}$$

To penalize sentences that are shorter than the reference, a brevity penalty term BP is also included in the metric.

BLEU is a corpus-level metric, meaning that it is meant to be computed as the

aggregate score over the set of sentences in a corpus. However, in learning, we oftentimes need to approximate BLEU on a sentence-level, since we will be evaluating the decision and loss our algorithm makes on each sentence. This poses a problem, since if the number of n-gram matches for any $j \in n$ is 0, which can reasonably happen in practice, BLEU is 0, and if the number of hypothesized n-grams is 0, BLEU is undefined. Furthermore, the BLEU score of a single sentence isolated from the corpus may not be informative of the overall BLEU score of the corpus.

For these reasons, it is necessary to approximate BLEU on a sentence-level. Several proposals have been suggested for dealing with this problem. The simplest is to smooth the score with pseudo counts (Lin and Och, 2004). This has traditionally been done in an *add-1* fashion, or with some decay, such as exponential, for the term to add to higher order n-grams (Li and Khudanpur, 2009):

$$p_j^{\text{smooth}} = \frac{\sum_{i=1}^{n} \frac{1}{2^j} + \sum_{g \in \text{grams}(j)} \text{count}_{\text{clip}}(g, \boldsymbol{y}'_i)}{\sum_{i=1}^{n} \frac{1}{2^j} + \sum_{g \in \text{grams}(j)} \text{count}(g, \boldsymbol{y}'_i)} \tag{2.19}$$

Watanabe et al. (2007) introduced an approximate scoring approach which scores a sentence as part of a pseudo-document of previously translated sentences. Chiang (2012) elaborated on this concept and added an exponential decay to the document, effectively keeping the impact of the new sentence relatively high. We make use of the add-1 smoothing with an exponential decay in Eq. 2.19 along with scoring sentences in the context of previous 1-best translations for our sentence-level approximation.

With the learning paradigm and evaluation metric defined, we will now present the most commonly applied or relevant approaches to learning for SMT, starting with MERT,

then moving to Minimum Risk, PRO, MIRA, and RAMPION.

2.4.5 Minimum Error Rate Training

Minimum-Error Rate Training (MERT) (Och, 2003) is the most popular parameter optimization technique for SMT in use today (Lopez, 2007; Macherey et al., 2008; Kumar et al., 2009). It is the yardstick against which all other methods have been compared, and therefore understanding MERT's strengths and weaknesses provides valuable context for the other methods.

MERT aims to optimize parameters to directly maximize the BLEU score of the maximum weighted derivation (Eq. 2.5), referred to as the 1-best BLEU, over the entire tuning set. Alternatively, we can say that it wants to minimize the error (1-BLEU). The loss function for MERT over the corpus of references $\boldsymbol{y}_i$ and system outputs $\boldsymbol{y}'$ can be expressed as:

$$\ell_{\text{MERT}} = \sum_{i=1}^{n} \Delta\left(\boldsymbol{y}_i, \underset{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i), \boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}')}{\operatorname{argmax}} \text{score}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d})\right) \tag{2.20}$$

Notice that it does not depend on the model score of the maximum derivation; we only use the model score to select the 1-best derivation to compute the cost. As the loss is non-convex and not differentiable for error metrics like BLEU, we cannot rely on gradient-based optimization procedures, such as those used in standard log-linear models (Liu and Nocedal, 1989).

Thus, MERT's optimization routine is gradient-free; coordinate descent with a specialized line search is the core technique behind MERT's parameter optimization, which

aims to directly minimize the corpus-level error on the training data. It does so by constructing a piece-wise linear error surface from the entire tuning set, and performing a line search in each dimension of the parameter vector in order to find the global minimum in that dimension. In practice, many random restarts and directions are used at each iteration (Macherey et al., 2008; Cer et al., 2008).

The error surface can be constructed from a k-best list of candidates from each sentence in the tuning set, as in the original formulation by Och (2003), or from a compact representation of the full hypothesis space, such as a lattice for phrase-based or hypergraph for hierarchical phrase-based translation (Macherey et al., 2008; Kumar et al., 2009).

As its strengths, the gradient-free optimization method is relatively simple to implement, and, when dealing with the standard small set of features, it is highly effective at optimizing parameters even on a small tuning set. Because it is gradient-free, it can be used to optimize non-differentiable task-specific loss functions. Furthermore, since it does not require computation of the features or score of the reference $\boldsymbol{y}_i$ in order to compute the loss, it is free to use the actual reference translation, not requiring the use of surrogate references. As a result, most MT systems are set up to optimize within the MERT framework.

The primary limitation of MERT, which is responsible for its inability to scale, is the unknown direction of the line search. When done in a handful of dimensions, random directions work relatively well; however, as we increase the dimensionality of the feature space this becomes a severe problem.

Foster and Kuhn (2009) and Hopkins and May (2011) have both recently explored

this phenomenon. Hopkins constructed a synthetic data set using a known optimal weight vector, reducing MERT's role to learning the initial weight setting that generated the data. Repeating this experiment with an increasing number of dimensions, they found that as the dimensions grow, MERT quickly is incapable of learning the correct weights. After tens of features the quality of MERT's line search decreases. As we move to take advantage of the high-dimensional feature spaces that are one of the main advantages of discriminatively trained systems, MERT's inability to scale is prohibitive in allowing research into more expressive features.

2.4.6 Minimum Risk

For this reason, while MERT remains the most widely used method, in recent years, there has been a growing trend moving away from MERT to other methods.

Expected BLEU (Rosti et al., 2010), or alternatively, minimum risk (Smith and Eisner, 2006; Li and Khudanpur, 2009), training is a probabilistic method that can be seen as an attempt to address some of these shortcomings. If instead of maximizing the 1-best hypothesis, as we do with MERT, we try to minimize the risk, or expected error, using the model's log-linear probability distribution over *all* the hypotheses, we create a continuous, and therefore differentiable, function for optimization which allows us to use gradient-based methods such as stochastic gradient descent or L-BFGS (Liu and Nocedal, 1989).

The loss can be computed over k-best list approximations of the model search space (Arun et al., 2010) or packed representations (Rosti et al., 2011). Smith and Eis-

ner (2006) present an annealed minimum risk approach which minimizes the expected loss for MT by gradually sharpening the objective to avoid local minima. Rosti et al. (2010) and Li and Khudanpur (2009) developed approaches for optimizing over lattices and hypergraphs, respectively.

Minimum risk training also does not require scoring the reference $\boldsymbol{y}_i$, and unlike conditional likelihood based models can incorporate an arbitrary cost function.

$$\begin{aligned}\ell_{\text{risk}} &= \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i), \boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y})} \Delta(\boldsymbol{y}_i, \boldsymbol{y}) \frac{\exp\left(\text{score}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d})\right)}{\sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i), \boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y})} \exp\left(\text{score}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d})\right)} \\ &= \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i), \boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y})} \Delta(\boldsymbol{y}_i, \boldsymbol{y}) p(\boldsymbol{d}, \boldsymbol{y} | \boldsymbol{x}_i)\end{aligned} \tag{2.21}$$

2.4.7 Pairwise Ranking Optimization

There are a number of other approaches that formulate tuning as a ranking problem (Chen et al., 2009; Haddow et al., 2011; Hopkins and May, 2011; Watanabe, 2012). Instead of focusing on the exact score of a hypothesis, these approaches aim to make sure better hypotheses are ranked higher by the scoring function.

Pairwise ranking optimization (PRO) (Hopkins and May, 2011) has recently come to the forefront of these approaches. In PRO, the task of learning to rank translations is reduced to binary classification between translation pairs using an off-the-shelf classifier. It samples pairs of hypotheses from aggregated k-best approximations of the search space and trains a binary classifier to make pairwise comparisons in accordance with the cost of each hypothesis. The main motivation for PRO was staying as close as possible to the

batch optimization framework established by MERT, but allowing the learning algorithm to scale to higher-dimensional feature spaces.

Since the set of candidate translations is derived from the model's hypothesis space, PRO also does not need to approximate the reference translation. However, as PRO has to compare candidate translations of a given source sentence, it approximates the corpus BLEU score with a sentence-level approximation.

2.4.8 Margin-based Methods

Another group of methods has moved away from MERT and the batch optimization strategies presented above, and turns to online linear large-margin optimization (Chiang et al., 2008a; Watanabe et al., 2007; Arun and Koehn, 2007). Most structured prediction learners, such as CRFs, Max-Margin Markov Networks, and Support Vector Machines (Lafferty et al., 2001; Taskar et al., 2004; Joachims, 1998) were designed as **batch** learners: they consider all examples simultaneously when optimizing the objective. The alternative, an **online** learner, optimizes one example, or a handful of examples, at a time, thus allowing more flexibility with regard to the size of the training set.[19]

Beyond the desire to tune a large number of features, online margin-based methods for SMT also target large-scale discriminative training scenarios where batch optimization is prohibitive or undesirable. While a slew of recent alternative optimization strategies focusing on margin-based methods have been proposed (Watanabe, 2012; Cherry and Foster, 2012; Chiang, 2012; Yu et al., 2013; Tan et al., 2013), here we focus on the most successful thus far.

[19] We will discuss the exact mechanics of different online learning scenarios in Chapter 3.

Liang et al. (2006) was one of the first to apply a structured online linear model, namely the structured perceptron (Collins, 2002), to SMT. This paved the way for many other large-margin linear models for discriminative training: MIRA (Chiang et al., 2009; Watanabe et al., 2007; Watanabe, 2012), batch-MIRA (Cherry and Foster, 2012), Structured SVM (Cherry and Foster, 2012), and RAMPION (Gimpel and Smith, 2012),

The main conceptual difference between these methods and probabilistic ones based on log-linear models is that margin-based methods do not posit a probabilistic view of $\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$. Instead they only care about a **geometric** view of the linear scoring function. Since each hypothesis is associated with a n-dimensional feature vector, composed of numerical attributes, all hypotheses can be mapped to a point in the $n-$dimensional feature space. Hypotheses that are similar to each other are then closer in distance to one another, and vice versa. The parameter vector $\mathbf{w}$ defines a separating hyperplane in n-1 dimensions, and the goal is to create a large separating distance between the correct hypothesis $\boldsymbol{y}_i$ and all others $\boldsymbol{y}'$. The distance between the scores of two hypotheses is the **margin**: $\text{score}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{d}) - \text{score}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d})$.

2.4.8.1 MIRA

Crammer and Singer (2003) first introduced the Margin Infused Relaxed Algorithm (MIRA) as an online large-margin learner for multiclass classification. Taskar et al. (2005) later introduced this to the NLP community as a method for performing large-margin online training for structured prediction problems, which has proven useful for applications such as dependency parsing (McDonald et al., 2005) and MT (Chiang et al.,

2008a). Intuitively, it can be viewed as either introducing a classification margin into the perceptron to reduce generalization error, creating a large-margin perceptron, or alternatively, as an online SVM.

The main motivation for MIRA is that we want our model to enforce a margin between the correct and incorrect outputs of a sentence that agrees with our cost function. This is done by making the smallest update possible to our parameters, $\mathbf{w}$, on every sentence, that will ensure that the difference in model scores $\delta\mathbf{f}_i(\boldsymbol{y}') = \mathbf{w}^\top(\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}'))$ between the correct output $\boldsymbol{y}_i$ and incorrect output $\boldsymbol{y}'$ is at least as large as the cost, $\Delta(\boldsymbol{y}_i, \boldsymbol{y}')$, incurred by predicting the incorrect output. The optimization problem is:

$$\begin{aligned} &\mathbf{w}_{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + C\xi_i \\ \text{s.t.} \quad &\mathbf{w}^\top\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \mathbf{w}^\top\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}') \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}') - \xi_i \quad \forall \boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i) \end{aligned} \tag{2.22}$$

where C is a regularization parameter that controls the size of the update, trading off between margin maximization and constraint violations. The underlying objective of MIRA is the same as that of the margin-rescaled Structural SVM (Tsochantaridis et al., 2004; Martins et al., 2010). Thus, MIRA can be seen as constructing a SVM on each instance, while the norm constraint $||\mathbf{w}||^2$ from SVM is replaced with a proximity, or conservativity, constraint, $||\mathbf{w} - \mathbf{w}_t||^2$ (Crammer et al., 2006). This indicates that we want to update our parameters, but keep them as close as possible to the previous parameter estimates. Since the worse $\boldsymbol{y}'$ is, the more distant its score will be from $\boldsymbol{y}_i$, simply scaling the weights could produce a sufficiently large margin to match any cost. Thus, the objective is to minimize

the distance between successive weight vectors.

In the original formulation for separable classification (Crammer and Singer, 2003), if no constraints are violated, no update occurs. However, when there is a loss, the algorithm updates the parameters to satisfy the constraints. A more general version of the MIRA algorithm was later presented by Crammer et al. (2006), where it was shown to belong to a class of passive-aggressive (PA) algorithms. A PA algorithm will not update the parameters when the loss is less than zero, or, in other words, it will be *passive* when we have correctly scored the output. However, when there is a loss, the weight update is *aggressive*, in that it forces the new weights to correctly score the output, regardless of the size of the necessary update. To allow for noise in the data, i.e. nonseparable instances, a slack variable ξ_i is introduced for each example, and we optimize the margin-rescaled version of the soft-margin (Joachims et al., 2009; Crammer and Singer, 2001), where we are permitted but penalized for not satisfying the constraints.

In order to optimize the objective in Eq. 2.22, we need to solve a quadratic program (QP) with linear constraints, which can be handled by techniques such as Hildreth's algorithm (McDonald et al., 2005), Sequential Minimal Optimization (SMO) (Platt, 1998; Chiang et al., 2008a), or using a cutting-plane (Joachims et al., 2009; Chiang, 2012). However, Crammer et al. (2006) showed that there exists an analytical, or closed-form, update for PA algorithms based on looking at only one constraint. This simplified update forgoes complex machinery, such as the cutting plane, and still guarantees a cumulative loss bound (Crammer et al., 2006). With a PA update, the $\forall \boldsymbol{y}'$ constraint above can be

approximated by selecting the single most violated constraint, which maximizes

$$\boldsymbol{y}^- \leftarrow \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}) + \Delta(\boldsymbol{y}_i, \boldsymbol{y}),$$

since the second term below is an upper bound on the first term:

$$\begin{aligned} \forall \boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i) \quad & - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}) + \Delta(\boldsymbol{y}_i, \boldsymbol{y})\right) \\ & \leq - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}') + \Delta(\boldsymbol{y}_i, \boldsymbol{y}')\right) \end{aligned} \tag{2.23}$$

Selecting $\boldsymbol{y}^-$ based on the model score and cost requires **cost-augmented decoding** (Smith, 2011). Although the exact update step is different from other subgradient optimizers, in essence this reduces to performing a subgradient descent step, where the step size is adjusted based on each example. This update is simple and performs well, reducing the optimization problem in Eq. 2.22 to:

$$\begin{aligned} \mathbf{w}_{t+1} &= \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + C\xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-) \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-) - \xi_i \end{aligned} \tag{2.24}$$

The structured hinge loss, which can be shown to underlie this objective (Martins et al., 2010) can be rewritten as:

$$\ell_{\text{hinge}} = -\mathbf{w}^\top \mathbf{f}(x_i, y_i) + \max_{y' \in \mathcal{Y}(x_i)} \left(\mathbf{w}^\top \mathbf{f}(x_i, y') + \Delta(y_i, y')\right) \tag{2.25}$$

Recently, MIRA has gained prominence for SMT, largely thanks to its ability to

learn models in low- and high-dimensional feature spaces. Arun and Koehn (2007) and Watanabe et al. (2007) were the first to apply a MIRA-based trainer to SMT. Chiang et al. (2008a) then extended this work, and through a series of improvements produced the most successful structured MIRA learner (Chiang et al., 2009; Chiang, 2012).

While previous settings of MIRA in machine learning were able to directly use $\boldsymbol{y}_i$ in the loss, for reasons of (un)reachability described in Section 2.4.3, applying MIRA to SMT necessitates approximating the correct translation with $\boldsymbol{y}^+ \in \mathcal{Y}(\boldsymbol{x}_i)$. We will defer an *in depth* discussion of the implications of this and strategies for selecting $\boldsymbol{y}^+$ to Chapter 3, but here we note that, this will require some form of cost-diminished decoding:

$$\boldsymbol{y}^+ \leftarrow \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}) - \Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

which, along with the presence of latent variables, has the effect of changing the optimization problem for SMT to:

$$\begin{aligned} \mathbf{w}_{t+1} &= \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + C\xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-) \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-) - \Delta(\boldsymbol{y}_i, \boldsymbol{y}^+) - \xi_i \end{aligned} \tag{2.26}$$

with a correspondingly modified loss:

$$\begin{aligned} \ell_{\text{PA}} = &- \max_{(\boldsymbol{y}^+, \boldsymbol{d}^+) \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \Delta(\boldsymbol{y}_i, \boldsymbol{y}^+) \right) \\ &+ \max_{(\boldsymbol{y}^-, \boldsymbol{d}^-) \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-) + \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-) \right) \end{aligned} \tag{2.27}$$

While most previous SMT literature has optimized the loss above and referred to

it as the structured hinge loss, Gimpel and Smith (2012) recently pointed out that we are in fact optimizing losses that are closer to different variants of the structured ramp loss (McAllester and Keshet, 2011). The difference in definition between the two lies in the fact that for the ramp loss, $\boldsymbol{y}_i$ is replaced with $\boldsymbol{y}^+$. The ramp loss objectives are non-convex, and since we are separately computing the $\max$ for both $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$, where the first $\max$ is negated, subgradient methods are no longer guaranteed to be optimizing the desired loss.

However, as with many non-convex optimization problems in NLP, such as those involving latent variables, in practice subgradient-based learning of Eq. 2.27 in this setting behaves quite well.

Table 2.4.8.1 presents a concise comparison between MERT, Minimum Risk, and MIRA training.

	MERT	MIRA	Minimum Risk
Type	1-best	Margin-based	Probabilistic
Objective	Minimize error	Minimize loss augmented score	Minimize expected error
Optimization	Line search	QP/PA	Gradient Based
Limitations	Direction of search unknown	Approximation of reference	Approximate expectation

Table 2.4: Comparison of training regimes for MERT, Minimum risk, and MIRA.

2.4.8.2 RAMPION

RAMPION (Gimpel and Smith, 2012) aims to directly address the disconnect between MT and machine learning discussed above by *actually* optimizing the loss function that MIRA wants to optimize: the structured ramp loss. RAMPION uses CCCP (Yuille and

Rangarajan, 2003), a batch concave-convex procedure also used for solutions to the latent SVM (Yu and Joachims, 2009) to minimize the ramp loss. CCCP can be used to solve loss functions of the form $f(\mathbf{w}) - g(\mathbf{w})$, which are a combination of a convex ($f(\mathbf{w})$) and concave ($g(\mathbf{w})$) term. RAMPION alternates between solving $g(\mathbf{w})$, by first finding the best $\boldsymbol{y}_i^+$ for each sentence in the whole tuning set, and then keeping those fixed, optimizing the loss as if the $\boldsymbol{y}_i^+$ are observed using stochastic subgradient descent. The loss function that RAMPION optimizes is thus:

$$\begin{aligned} \ell_{\text{ramp}} = & \max_{(\boldsymbol{y}^-, \boldsymbol{d}^-) \in \mathcal{Y}(x_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-) + \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-) \right) \\ & - \max_{(\boldsymbol{y}^+, \boldsymbol{d}^+) \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \Delta(\boldsymbol{y}_i, \boldsymbol{y}^+) \right) \end{aligned} \tag{2.28}$$

2.5 Summary

In this chapter we introduced the problem of statistical machine translation, giving a brief overview of the progression of the field from its inception to today. In so doing, we covered modeling, decoding, and learning concerns. In modeling, we define the translation equivalence and latent structure our model uses, along with the feature functions. In decoding, we search for the best translation under our decision rule, usually Eq. 2.5. In learning, we select the optimal $\mathbf{w}$. It would be fair to say that progress in SMT depends on progress in these three issues.

In this thesis, we focus on the learning and modeling issue, introducing advancements to current learning algorithms in Chapters 3, 4 and 6, and new features for the translation model in Chapter 5. Although we concentrate on SMT, the algorithms we

propose in the following chapters can be directly (or with slight modification) applied to other structured prediction tasks.

3 Large-Scale Online Large-Margin Learning

I have no data yet. It is a capital mistake to theorise before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

— Sherlock Holmes

Learning is not child's play; we cannot learn without pain.

— Aristotle

In this chapter we contribute a deeper understanding of online large-margin learning for statistical machine translation. The difficulties of applying large-margin methods in this setting described in Chapter 2 have led to a wide array of implementations and confusion about proper optimization. Thus, we address the question of what to optimize in SMT by proposing a unified form for different *cost-augmented objectives in SMT* as a *family of loss functions* and placing prior work within this setting. We utilize this framework to extensively empirically analyze the optimization performance of different loss functions, parallelization, and updating strategies, first in the standard low-dimensional setting, and then moving to high dimensions.

After we establish the proper loss function to optimize for both learning stability and generalization ability, we move to *large-scale discriminative training*. We develop a decoder-agnostic tool for distributed learning on a MapReduce architecture, and show the practicability and benefits of large-scale training.

The contributions in this chapter are largely empirical, and specifically aimed at providing a practical guide for practitioners of SMT of best practices for large-margin learning.[1]

3.1 Introduction

MIRA (Crammer et al., 2006) is a popular method for online large-margin optimization, which has been shown to perform well for machine translation, as well as other structured prediction tasks (McDonald et al., 2005). This is an attractive method because, as we showed in Chapter 2, it has a simple analytical solution for the optimization problem at each step, which reduces to dual coordinate descent when using 1-best MIRA.

Despite the proven success of MIRA-based large-margin optimization for both small and large numbers of features (Chiang, 2012), these methods have not yielded wide adoption in the community. This is partially due to a perception that these methods are complicated to implement, which has been cited as motivation for other work (Hopkins and May, 2011; Gimpel and Smith, 2012). The complications stem in part from the divergence between the standard application of these methods in machine learning, as discussed in Section 2.4.3, and our application in machine translation, where we have no

[1] This chapter is based on material originally published in Eidelman (2012), Eidelman et al. (2013c), and Eidelman et al. (2013b).

unique correct output and latent structures. As a consequence of the above, there is a lack of understanding in what MIRA optimizes, and the best practices for large-margin learning for SMT, which have resulted in numerous different implementations of MIRA-based optimizers, further adding to the confusion.

This chapter aims to shed light on practical concerns with online large margin training. Specifically, we first unify previous objectives into a single family of losses (§3.2.1), and present the MIRA passive-aggressive update for SMT (§3.3.3), which underlies all MIRA-based training. The resulting algorithm can be used directly for learning in SMT and is simple to implement.

We then empirically analyze several widespread as well as novel optimization strategies that emerge from this framework for large-margin training on Czech-to-English (cs-en) and French-to-English (fr-en) translation across important factors for learning: **loss function** (§3.4.2), **parallelization** (§3.5.1), and **updating strategies** (§3.5.2).

In Section 3.6 we move to large-scale learning in MapReduce and develop tools for highly distributed learning, showing that online large-margin learning is not only capable of handling large feature spaces, but scaling to large data as well.

3.2 Large-Margin Learning

Recall that our training corpus $\mathcal{T} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{T}$ for selecting the parameters $\mathbf{w}$ that optimize our objective consists of input sentences $\boldsymbol{x}_i$ in the source language paired with reference translations $\boldsymbol{y}_i$ in the target language. In Section 2.4.8.1, we saw that the usual presentation of MIRA is given as:

$$\begin{aligned} \mathbf{w}_{t+1} &= \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + C\xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-) \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-) - \xi_i \end{aligned} \tag{3.1}$$

with an underlying structured hinge loss objective function:

$$\ell_{\text{hinge}} = -\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}') + \Delta(\boldsymbol{y}_i, \boldsymbol{y}') \right) \tag{3.2}$$

3.2.1 Hypothesis Selection

Notice that ℓ_{hinge} depends on computing the margin between $\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i)$ and the **correct** output, $\boldsymbol{y}_i$. However, there is no guarantee that $\boldsymbol{y}_i \in \mathcal{Y}(\boldsymbol{x}_i)$ since our decoder is often incapable of producing the reference translation $\boldsymbol{y}_i$. Since we need to have some notion of the correct output in order to compute its feature vector for the margin, in practice we revert to using surrogate references in place of $\boldsymbol{y}_i$. These are often referred to as **oracles**, or **hope** (Chiang, 2012), translations $\boldsymbol{y}^+$, which are selected from the hypothesis space $\mathcal{Y}(x_i)$ of the decoder with cost-diminished decoding.[2]

We are also faced with the standard problem of how best to select the most appropriate $\boldsymbol{y}'$ to shy away from, which we will refer to as $\boldsymbol{y}^-$. These are referred to as **fear** translations (Chiang, 2012). Since optimization will proceed by setting parameters to increase the score of $\boldsymbol{y}^+$, and decrease the score of $\boldsymbol{y}^-$, the selection of these two hypotheses is crucial to success. We present a unified view of the possible loss functions for cost-augmented and diminished learning in Eq. 3.3.

[2] In practice we can define $\mathcal{Y}(\boldsymbol{x}_i)$ to either be the entire hypergraph or k-best output $\mathcal{K}(\boldsymbol{x}_i)$.

$$\ell = - \max_{(\boldsymbol{y}^-, \boldsymbol{d}^-) \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \left(\gamma^+ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{d}^+) - \beta^+ \Delta(\boldsymbol{y}_i, \boldsymbol{y}^+) \right) + \max_{(\boldsymbol{y}^-, \boldsymbol{d}^-) \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \left(\gamma^- \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}^-, \boldsymbol{d}^-) + \beta^- \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-) \right) \tag{3.3}$$

Each setting of $\gamma^\pm$ and $\beta^\pm$ corresponds to optimizing a different loss function. Several definitions of ℓ_r have been explored in the literature, and we discuss them below with corresponding settings of $\gamma^\pm$ and $\beta^\pm$. To our best knowledge, other loss functions explored below are novel.

3.2.1.1 Oracle Selection

In selecting $\boldsymbol{y}^+$, we vary the settings of γ^+ and β^+. Assuming our cost function is based on BLEU, in setting $\beta^+ \to 1$ and $\gamma^+ \to 0$, if $\mathcal{Y}(\boldsymbol{x}_i)$ is taken to be the entire space of possible translations, we are selecting the hypothesis with the highest BLEU overall. This is referred to in past work as max-BLEU (Tillmann and Zhang, 2006) (MB).

$$(\boldsymbol{y}^+, \boldsymbol{d}^+) \leftarrow \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} -\Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

One special case for max-BLEU oracle selection is to update towards the actual reference $\boldsymbol{y}_i$ and simply throw away the training sentence pairs for which the decoder is unable to generate the reference. This approach, referred to as bold or optimistic updating, was first explored by Liang et al. (2006), and then repeated in Blunsom et al. (2008b). A variation on this is performed by Tillmann and Zhang (2006) and Arun and Koehn (2007), where a preprocessing run with a decoder modified to use BLEU against references as its scoring function is used to find the best candidate translation, and these so-called

surrogate references are saved for use in the learning phase. Both Arun and Koehn (2007) and Liang et al. (2006) found that updating conservatively toward a derivation which the decoder produces at each iteration of training performs significantly better than updating optimistically toward the reference.

Another related method is to approximate the search space by restricting $\mathcal{Y}(\boldsymbol{x}_i)$ to a k-best list $\mathcal{K}(\boldsymbol{x}_i)$, and update towards the lowest cost candidate. In this case, we have the so-called local-update (Liang et al., 2006; Arun and Koehn, 2007; Chiang et al., 2009; Watanabe et al., 2007), where we select the highest BLEU candidate from those hypotheses that the model already considers good (LU):

$$(\boldsymbol{y}^+, \boldsymbol{d}^+) \leftarrow \underset{\boldsymbol{y} \in \mathcal{K}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\operatorname{argmax}} -\Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

With increasing k-best size, the max-BLEU and local-update strategies begin to converge.

Setting both $\beta^+ \rightarrow 1$ and $\gamma^+ \rightarrow 1$, we obtain the cost-diminished hypothesis, which considers both the model, i.e. model score, and the cost, and corresponds to the *hope* hypothesis (Chiang et al., 2008a; Chiang, 2012) (M-C):

$$(\boldsymbol{y}^+, \boldsymbol{d}^+) \leftarrow \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\operatorname{argmax}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d}) - \Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

This can be computed over the entire space of hypotheses or a k-best list. In a sense, this is the intuition that local-updating is after, but expressed more directly, since we are explicitly accounting for both the model score and cost. Part of the impetus behind selecting the oracle using this combination is that some translations may have a low cost,

e.g. a high BLEU score, but a distant model score from the area in which most good translations reside, which could cause issues if we are searching for only the lowest cost candidate across the whole search space, or a very large k-best list.

3.2.1.2 Fear Selection

The alternatives for selecting $\boldsymbol{y}^-$ are quite similar. Setting $\beta^- \to 1$ and $\gamma^- \to 0$, we penalize the hypothesis with the highest cost (MC), which to our best knowledge has not been evaluated in SMT:

$$(\boldsymbol{y}^-, \boldsymbol{d}^-) \leftarrow \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\operatorname{argmax}} \Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

Setting $\beta^- \to 0$ and $\gamma^- \to 1$, we select the highest scoring hypothesis according to the model, which corresponds to prediction-based selection (Crammer et al., 2006) (PB):

$$(\boldsymbol{y}^-, \boldsymbol{d}^-) \leftarrow \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\operatorname{argmax}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d})$$

Setting both to 1, we have the cost-augmented hypothesis, which is referred to as the *fear* (Chiang et al., 2008a), and max-loss since it is the maximum violator of the constraints (Crammer et al., 2006) (M+C):

$$(\boldsymbol{y}^-, \boldsymbol{d}^-) \leftarrow \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\operatorname{argmax}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d}) + \Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

This hypothesis is considered the most dangerous because it has a high model score along

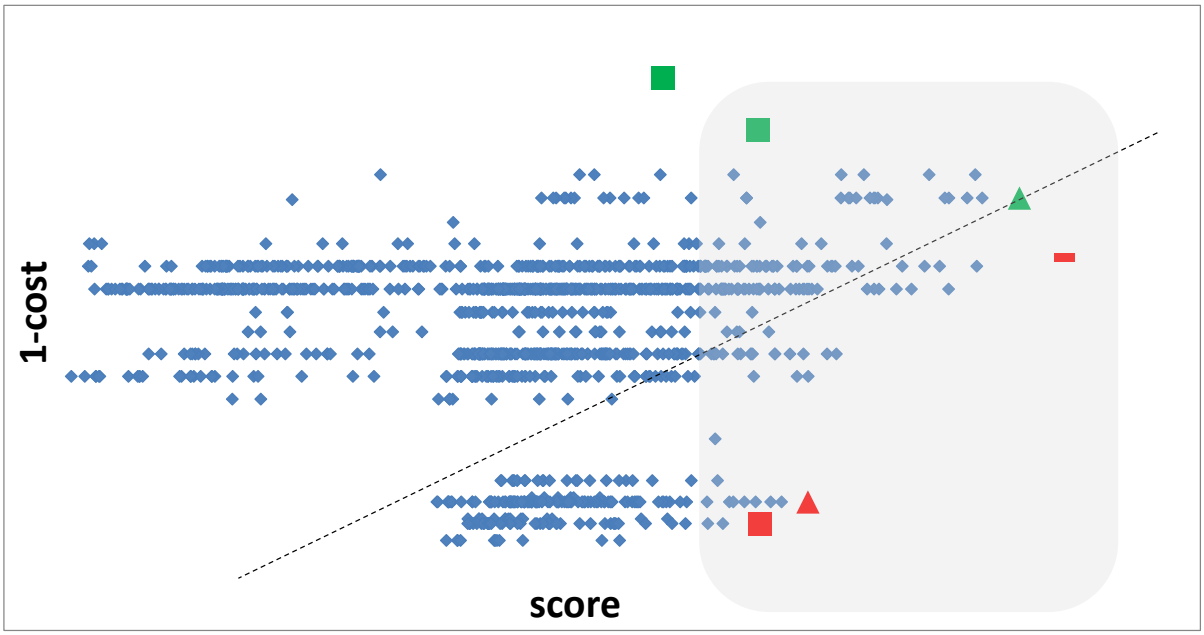


Figure 3.1: Visual representation of hypotheses generated for one instance graphed according to their BLEU and model score. The shaded region represents the k-best output. The green elements represent alternatives for $\boldsymbol{y}^+$ selection: □ outside k-best is max-BLEU, and the same square inside the k-best is the local update, △ is the cost-diminished selection. The red elements represent $\boldsymbol{y}^-$: □ is the max-cost, △ is cost-augmented, and - is prediction-based. The dotted-line represents the point where loss is 0, i.e., margin is equal to cost.

with a high cost. This is the constraint we usually care most about since it directly maximizes the hinge loss: our system believes the hypothesis is a good one, and thus assigns it a high score, while the cost is high, thus proving in fact to be a poor translation.

Figure 3.1 represents the hypothesis space according to cost and model score for one example, showing the visual representation of the different oracle and fear selection strategies and their corresponding location in the search space.

Considering the settings for both parts of Eq. 3.3, γ^+, β^+ and γ^-, β^-, assigning all $\gamma^\pm$ and $\beta^\pm$ to 1 corresponds to the most commonly used loss function in MT (Gimpel and Smith, 2012; Chiang et al., 2009). This is the hope/fear pairing, where we use the

cost-diminished hypothesis $\boldsymbol{y}^+$ and cost-augmented hypothesis $\boldsymbol{y}^-$:

$$\begin{aligned}\ell = &- \max_{(\boldsymbol{y}^+,\boldsymbol{d}^+)\in\mathcal{Y}(\boldsymbol{x}_i),\mathcal{D}(\boldsymbol{x}_i,\boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \Delta(\boldsymbol{y}_i, \boldsymbol{y}^+)\right) \\ &+ \max_{(\boldsymbol{y}^-,\boldsymbol{d}^-)\in\mathcal{Y}(\boldsymbol{x}_i),\mathcal{D}(\boldsymbol{x}_i,\boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-) + \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-)\right)\end{aligned} \tag{3.4}$$

Other loss functions that have been explored are similar to $\gamma^\pm \to 1$, $\beta^+ \to 1$, $\beta^- \to 0$ (Liang et al., 2006):

$$\begin{aligned}\ell = &- \max_{(\boldsymbol{y}^+,\boldsymbol{d}^+)\in\mathcal{Y}(\boldsymbol{x}_i),\mathcal{D}(\boldsymbol{x}_i,\boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(x_i, y^+, \boldsymbol{d}^+) - \Delta(\boldsymbol{y}_i, \boldsymbol{y}^+)\right) \\ &+ \max_{(\boldsymbol{y}^-,\boldsymbol{d}^-)\in\mathcal{Y}(\boldsymbol{x}_i),\mathcal{D}(\boldsymbol{x}_i,\boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-)\right)\end{aligned} \tag{3.5}$$

which does not consider cost when selecting $\boldsymbol{y}^-$. Liang et al. (2006) used this loss with a perceptron using the 1-best derivation to update towards the 1-best oracle. This can be approximated as a special case of 1-best MIRA, which produces the standard perceptron update rule. Another loss used is an approximation of $\gamma^\pm \to 1$, $\beta^+ \to 0$, $\beta^- \to 1$ (Cherry and Foster, 2012):

$$\begin{aligned}\ell = &- \max_{(\boldsymbol{y}^+,\boldsymbol{d}^+)\in\mathcal{Y}(\boldsymbol{x}_i),\mathcal{D}(\boldsymbol{x}_i,\boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+)\right) \\ &+ \max_{(\boldsymbol{y}^-,\boldsymbol{d}^-)\in\mathcal{Y}(\boldsymbol{x}_i),\mathcal{D}(\boldsymbol{x}_i,\boldsymbol{y}_i)} \left(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-) + \Delta(\boldsymbol{y}_i, \boldsymbol{y}^-)\right)\end{aligned} \tag{3.6}$$

which is closer to the usual loss used for max-margin in machine learning.[3]

Since our external metric, BLEU, is a gain, we can think of the first term in Eq. 3.4 as the model score plus the BLEU score, and the second term as the model minus the

[3] As discussed in the Section 2.4.8.1, Gimpel and Smith (2012) have recently pointed out that these types of loss are different variants of the structured ramp loss.

BLEU score. That is, with all $\gamma^{\pm}$ and $\beta^{\pm}$ set to 1, we want $\boldsymbol{y}^{+}$ to be the hypothesis with a high model score, as well as being close to the reference translation, as indicated by a high BLEU score. While for $\boldsymbol{y}^{-}$, we want a high model score, but it should be far away from the reference, as indicated by a low BLEU score. The motivation for choosing $\boldsymbol{y}^{-}$ in this fashion is grounded in the fact that since we are penalized by this term in the loss, we should try to optimize on it directly. In practice, we can compute the cost $\Delta(\boldsymbol{y}_i, \boldsymbol{y})$ for both terms as $1 - \text{BLEU}(\boldsymbol{y}, \boldsymbol{y}_i)$.

3.3 Online Learning

Online learning is one of the oldest machine learning approaches, first gaining prominence with the perceptron (Rosenblatt, 1958). It has nonetheless proven itself to be one of the most successful and popular methods across a variety of tasks due to its speed, simplicity, robustness and scalability (Crammer et al., 2012; Schaul and LeCun, 2013). It offers **fast convergence** to reasonable solutions, and is **memory efficient**, since only one training instance needs to loaded in memory, thus facilitating scaling. It is especially applicable to natural language problems, which deal with large training sets and high-dimensional input representations. Online learners achieve some of the best performance on tasks such as parsing (McDonald et al., 2005), part-of-speech tagging (Shen, 2007), and SMT (Chiang et al., 2008a).

The basic online learning algorithm performs iterative optimization, and is an instance of Stochastic Subgradient Descent (SGD) (Ratliff et al., 2006). It proceeds as in Algorithm 1. We operate on the data in several successive iterations, or rounds. During

each iteration, the algorithm receives a training example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, and must predict an output based on the current parameter settings. The correct answer is revealed immediately after the prediction, and if we predicted incorrectly, we perform an update of the weight vector. We then use the new weight vector to process the next round's instance, and so on, until we reach a predetermined convergence criterion. (The exact form of the update will depend on the algorithm employed, and will be the subject of future sections.) The goal is to answer with the fewest number of mistakes.

After convergence, we may choose to average over the weight vectors to produce a final weight vector. We may average over all weight updates, only the final update from each iteration, or not average at all. The averaging of the weight vector, rather than using the final weight, is done to reduce overfitting, and has proved effective in previous applications (Collins, 2002). Whichever method is used to produce the final weight vector, these weights are then used as the parameters $\mathbf{w}$ of our model.

Batch learners usually make fewer passes through the data, but one pass through the entire data is required to make one update; although the update is better informed about the global loss, we have wasted resources solving for each instance with the same stale parameters. On the other hand, SGD learners make many more smaller updates, which although individually noisier, result in faster convergence and robustness to local optima (Schaul and LeCun, 2013; Bottou and Bousquet, 2011).

3.3.1 Learning Rate

In online learning, since we learn from each instance, the update is based on only that instance's component of the overall loss. This updating method can be shown to be a generalization of gradient descent, which is a method for updating parameters using the entire gradient of the loss function with respect to the parameters (Smith, 2011). Instead, here we only take the subgradient with respect to the active parameters, and step in the direction of the subgradient. The general form of the SGD online update is:

$$\begin{aligned} \mathbf{w} \leftarrow & \mathbf{w} + \eta \nabla \ell_t \\ \nabla \ell_t = & \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-) \end{aligned} \tag{3.7}$$

where $\nabla \ell_t$ is the subgradient of the loss at time t, and η is the learning rate, or **step size**; how much we allow our parameters to change at each update. Determining the proper step size to adjust the parameters for each update is a crucial part of effective online learning. Both small and large step sizes may pose problems. Different learning rate schedules have been suggested which lead to improved performance (Schaul et al., 2013; Schaul and LeCun, 2013). Commonly, these adapt the learning rate by decreasing it throughout learning, adjusting it on a per-instance basis, or even per-parameter basis.

Since the MIRA optimization problem is an instance of a general structured problem with an ℓ_2 norm, the passive-aggressive update we discussed in §2.4.8.1 at each step reduces to dual coordinate descent (Smith, 2011). In our soft-margin setting, this is analogous to the PA-I update of Crammer et al. (2006). Thus, we are performing a subgradient descent step with a single learning rate η for all features, but with an additional α step

size adjusted based on each example:

$$\begin{aligned} \alpha &\leftarrow \min\left(C, \frac{\ell}{\|\nabla \ell_t\|^2}\right) \\ \mathbf{w} &\leftarrow \mathbf{w} + \alpha\eta\nabla\ell_t \end{aligned} \tag{3.8}$$

The step size η along with α limits the amount each feature weight can change at each update. However, since the typical dense features (e.g., language model) are observed far more frequently than sparse features (e.g., rule identity), it has been shown to be advantageous to use an adaptive per-feature learning rate that allows larger steps for features that do not have much support (Green et al., 2013; Duchi et al., 2011). Essentially, in AdaGrad (Duchi et al., 2011) instead of having a single parameter η, we instead have a vector Σ with one entry for each feature weight:

$$\begin{aligned} \Sigma_{t+1}^{-1} &\leftarrow \Sigma_t^{-1} + \lambda \text{diag}\left(\nabla\ell_t \nabla\ell_t^\top\right) \\ &= \sum_{j=1}^{t+1} \lambda \text{diag}\left(\nabla\ell_j \nabla\ell_j^\top\right) \\ \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \alpha\Sigma_{t+1}^{1/2}\nabla\ell_t \end{aligned} \tag{3.9}$$

where for each dimension i in the weight vector at time t the update becomes:

$$\begin{aligned} \nabla_i \ell_t &= \frac{\partial \ell_t}{\partial w_i} = f_i(\boldsymbol{x}, \boldsymbol{y}^+, \boldsymbol{d}^+) - f_i(\boldsymbol{x}, \boldsymbol{y}^-, \boldsymbol{d}^-) \\ \Sigma_{i,t+1}^{-1} &= \sum_{j=1}^{t+1} \nabla_i \ell_j^2 \\ w_{i,t+1} &\leftarrow w_{i,t} + \alpha \frac{\nabla_i \ell_t}{\sqrt{\sum_{j=1}^{t+1} \nabla_i \ell_j^2}} \end{aligned} \tag{3.10}$$

Algorithm 1 Algorithm for Stochastic Gradient Descent

Require: : Training set $\mathcal{T} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{T}$

$\mathbf{w}_{0,0} \leftarrow 0$
for iteration $i \leftarrow 0$ to N **do**
  for example $j \leftarrow 0$ to T **do**
    $\mathcal{Y}(\boldsymbol{x}_i) \leftarrow \text{DECODE}(\boldsymbol{x}_j, \mathbf{w}_{i,j})$
    $\mathbf{w}_{i,j+1} \leftarrow \text{UPDATE}(\mathbf{w}_{i,j})$
  end for
  $\mathbf{w}_{i+1,0} \leftarrow \mathbf{w}_{i,T}$
end for
$\mathbf{w}_a = \frac{1}{NT} \sum^{N} \sum^{T} \mathbf{w}_{i,j}$
$\mathbf{w}_b = \frac{1}{N} \sum^{N} \mathbf{w}_{i,0}$
$\mathbf{w}_f = \mathbf{w}_{N,T}$
return $\mathbf{w}_{a|b|f}$

This means in addition to the $\mathbf{w}$ we need to keep track of a $|\mathbf{w}|$ dimensional vector accumulating the squared subgradients for each feature. This adaptive update is very similar to that of AROW (Crammer et al., 2009a; Chiang, 2012).

3.3.2 Parallelization

A number of improvements to the basic algorithm (Alg. 1) have been presented in order to allow for efficient processing of large training sets.

To make parameter estimation more efficient and scalable, some form of parallelization is necessary, where instead of a single learner, we have multiple concurrent learners. Thus, the first adaptation to the SGD algorithm introduces parallelization in the framework of MapReduce (Dean and Ghemawat, 2008) and parameter mixing (McDonald et al., 2010), where the training data is broken up into shards s, with each shard undergoing SGD training separately, but in parallel, and then having their final weight vectors mixed together by averaging. This is presented in Algorithm 2.

Algorithm 2 Algorithm for Parameter Mixing

Require: : Training set $\mathcal{T} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{T}$

1: $\mathbf{w}_{s,0,0} \leftarrow 0$
2: **for** shard s $\in$1...K **in parallel do**
3: **for** iteration $i \leftarrow$ 0 to N **do**
4: **for** example $j \leftarrow$ 0 to T **do**
5: $\mathcal{Y}(\boldsymbol{x}_i) \leftarrow$DECODE($\boldsymbol{x}_j$,$\mathbf{w}_{s,i,j}$)
6: $\mathbf{w}_{s,i,j+1} \leftarrow$UPDATE($\mathbf{w}_{s,i,j}$)
7: **end for**
8: $\mathbf{w}_{s,i+1,0} \leftarrow \mathbf{w}_{s,i,T}$
9: **end for**
10: **end for**
11: $\mathbf{w}_f = \frac{1}{K} \sum^{K} \mathbf{w}_{a|b|f}$
12: **return** $\mathbf{w}_f$

The second adaptation presented by McDonald et al. (2010) is iterative parameter mixing, where weights are not only mixed at the end of training, but are mixed between shards after each parallel iteration. The same mixed weights are used to initialize each shard for the next iteration. This is shown in Algorithm 3, with the mixing of weights taking place on line 10.

While Chiang (2012) develops a complex parallelization procedure which necessitated passing derivations and updates between learners, performing iterative parameter mixing has been shown to be an effective alternative (Chiang, 2012).

From a multi-task learning perspective (Duh et al., 2010; Simianer et al., 2012), each learner can be seen as optimizing the objective for its own task, and by learning multiple parameter vectors and then mixing them we are creating a set of features and weights that are useful across all of them. Thus, by simultaneously learning toward multiple related tasks, as opposed to each task individually, we achieve greater generalization.

We empirically analyze different parallelization strategies in Section 3.5.1.

Algorithm 3 Algorithm for Iterative Parameter Mixing

Require: : Training set $\mathcal{T} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{T}$

u$\leftarrow 0$
for iteration $i \leftarrow 0$ to N **do**
  for shard s $\in$1...K **in parallel do**
    $\mathbf{w}_{s,i,0} \leftarrow$**u**
    for example $j \leftarrow 0$ to T **do**
      $\mathcal{Y}(\boldsymbol{x}_i) \leftarrow$DECODE($\boldsymbol{x}_j$,$\mathbf{w}_{s,i,j}$)
      $\mathbf{w}_{s,i,j+1} \leftarrow$UPDATE($\mathbf{w}_{s,i,j}$)
    end for
  end for
  u $\leftarrow \frac{1}{K} \sum^{K} \mathbf{w}_{s,i,T}$
end for
return u

3.3.3 Parameter Update

The major practical concern with online large-margin methods for SMT is that oftentimes the implementation aspect is unclear and appears difficult. This is further compounded with a lack of standard practices; both theoretical, such as the objective to optimize, and practical, such as efficient parallelization. The former is a result of the disconnect discussed in §2.4.3 between the standard machine learning setting, which posits reachable references and lack of latent variables, and our own application. The latter is an open engineering problem. Both of these aspects have been receiving recent attention (McAllester et al., 2010; McAllester and Keshet, 2011; Gimpel and Smith, 2012; McDonald et al., 2010), and although certain questions remain as to the exact loss being optimized, we now have a better understanding of the theoretical underpinnings of this method of optimization. We now review the methods that have been suggested for optimizing this criterion.

The first adaptations of MIRA-based learning for structured prediction in NLP uti-

lized a set of k constraints, either for $\boldsymbol{y}^+$, $\boldsymbol{y}^-$, or both. This complicated the optimization by creating a quadratic programming (QP) problem with a set of linear constraints which needed to be solved with either Hildreth's algorithm or SMO style optimization, thereby precluding the possibility of a simple analytical solution. In Chiang et al. (2008a), they use k-best MIRA, where derivations are obtained directly from the hypergraph to generate constraints, and they update toward the 1-oracle. They obtain three different sets of k-best lists. The first is the model k-best, chosen only according to model score, the second is the cost-diminished hypothesis list, the first of which is referred to as hope, and the third is the cost-augmented hypothesis list.

Watanabe et al. (2007) goes further to k-best MIRA with k-oracle by generating a k-best list of oracles $\mathcal{O}(x)$.

The more candidates present, the more constraints on the QP exist, and the more computationally intensive each updating iteration becomes. Thus there is a tradeoff between having enough constraints to achieve good performance and running time. The approaches based on k-constraints use an arbitrary k, usually 10, as this is sufficient to achieve close to best performance. Furthermore, large k have been shown to degrade performance, possibly due to overfitting (McDonald et al., 2005).

Recently, Chiang (2012) introduced a cutting plane algorithm for use in MIRA, like that of Structured SVM's (Tsochantaridis et al., 2004), which optimizes on a small set of active constraints. The algorithm is reproduced in Alg. 5. As opposed to attempting to satisfy the exponentially many constraints, or k constraints simultaneously, the cutting plane algorithm creates successively tighter relaxations of the objective by selecting a growing subset of constraints and satisfying them exactly. This guarantees a solution

which is accurate to within ϵ, meaning that the other exponential number of remaining constraints are not violated by more than ϵ.

While these methods of dealing with structured prediction are in principle able to have tighter error bounds, they come with a higher computational cost. However, the passive-aggressive update shows that satisfying the single most violated margin constraint, commonly referred to as 1-best MIRA, is amenable to a simple analytical solution for the optimization problem at each step (Crammer et al., 2006). Furthermore, the 1-best MIRA update is conceptually and practically much simpler, while retaining most of the optimization power of the more advanced methods.

In fact, this update remains largely intact as the inner core within k-best constraint or cutting plane optimization. Algorithm 4 presents the entire update necessary for each instance in 1-best MIRA training of a machine translation system. When this method replaces Update$(\mathbf{w}_{i,j})$) on line 5 in Alg. 1, we have a single processor MIRA learner. When it replaces Update$(\mathbf{w}_{s,i,j})$ on line 7 of Alg. 3, we have a parallelized MIRA learner. This is the method we use in practice.[4]

As can be seen, the parameter update at step 11 depends on the difference between the features of $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$, where α is the step size, which is controlled by the regularization parameter C; indicating how far we are willing to move at each step. $\mathcal{Y}(\boldsymbol{x}_i)$ may be a k-best list or the entire space of hypotheses.

We empirically compare cutting-plane optimization with passive-aggressive updates in Section 3.5.2.

[4]The algorithm is agnostic as to the underlying hardware: it can be executed using multiple cores on a single machine, or on a cluster.

3.4 Experiments

3.4.1 Setup

To empirically analyze which loss, and thereby which strategy, for selecting $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$ is most appropriate for machine translation, we conducted a series of experiments on Czech-to-English and French-to-English translation. The parallel corpora are taken from the WMT2012 shared translation task, and consist of Europarl data along with the News Commentary corpus (Koehn, 2005). Table 3.1 summarizes the data statistics. All data were tokenized and lowercased, then filtered for length and aligned using the GIZA++ implementation of IBM Model 4 (Och and Ney, 2003) to obtain bidirectional alignments, which were symmetrized using the grow-diag-final-and method (Koehn et al., 2003). Grammars were extracted from the resulting parallel text and used in our hierarchical phrase-based system implemented in `cdec` (Dyer et al., 2010).

All our experiments are carried out within `cdec`, an efficient and modular open source framework for aligning, training, and decoding with a number of different translation models, including SCFGs (Dyer et al., 2010).[5] `cdec`'s modular framework facilitates seamless integration of a translation model with different language models, pruning strategies and inference algorithms. As input, `cdec` expects a string, lattice, or context-free forest, and uses it to generate a hypergraph representation, which represents the full translation forest without any pruning. The forest can now be rescored, by intersecting it with a language model, for instance, to obtain output translations. The above capabilities

[5] http://cdec-decoder.org

of `cdec` allow us to perform the experiments described in this and subsequent chapters, which would otherwise be quite cumbersome to carry out in another system.

We constructed a 5-gram language model from the provided English News monolingual training data as well as the English side of the parallel corpus using the SRI language modeling toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1996). This was binarized using KenLM (Heafield, 2011).

As the tuning set for both language pairs, we used the 2051 sentences in news-test2008 (NT08), and report results on the 2525 sentences of news-test2009 (NT09) and 2489 of news-test2010 (NT10) using the parameters from the best scoring iteration on the tuning set.

Corpus	Sentences	Tokens	
		en	*
cs-en	764K	20.5M	17.5M
fr-en	2M	57M	63M

Table 3.1: Corpus statistics.

We approximate cost-augmented decoding by obtaining a k-best list with k=500 unique best from our decoder at each iteration, and selecting the respective hypotheses for optimization from it. To approximate max-BLEU decoding using a k-best list, we set k=50k unique best hypotheses.[6] As can be seen in Table 3.2, we found this size was sufficient for our purposes as increasing size led to small improvements in oracle BLEU score. We use a learning rate of η=1 and regularization strength of C=0.01. The decoder is

[6] We are theoretically able to extract more constraints from a large list, in the spirit of k-constraints or a cutting plane, but (Chiang, 2012) and ourselves observe that cutting plane performance is approximately 0.2-0.4 BLEU better than a single constraint, so although there is a tradeoff between the simplicity of a single constraint and performance, it is not substantial.

pair	1	500	50k	100k
cs-en	17.9	24.9	29.4	29.7
fr-en	20.25	29.9	33.8	34.1

Table 3.2: Oracle score for model 1-best (baseline) and for k-best of size 500, 50k, and 100k on NT08.

configured to use cube pruning (Huang and Chiang, 2007) with a limit of 200 candidates at each node.

For comparison with MERT, we create a baseline model which uses the small standard set of features described in Section 2.2.4.1.

To optimize the feature weights for our model with MERT, we use Viterbi envelope semiring training (VEST), which is an implementation of MERT over hypergraphs (Dyer et al., 2010; Och, 2003). VEST reinterprets MERT within a semiring framework, which is a useful mathematical abstraction for defining two general operations, addition ($\oplus$) and multiplication ($\otimes$) over a set of values. Formally, a semiring is a 5-tuple $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$, where addition must be communicative and associative, multiplication must be associative and must distribute over addition, and an identity element exists for both. For VEST, having $\mathbb{K}$ be the set of line segments, $\oplus$ be the union of them, and $\otimes$ be Minkowski addition of the lines represented as points in the dual plane, allows us to compute the necessary MERT line search with the INSIDE algorithm.[7]

While BLEU is usually calculated at the corpus level, we need to approximate the metric at the sentence level. In this, we follow a slightly different combination of previous approaches, where in the first iteration through the corpus we use a smoothed sentence

[7]This algorithm is equivalent to the hypergraph MERT algorithm described by Kumar (Kumar et al., 2009).

level BLEU approximation, as given in Eq. 2.19, and in subsequently iterations, the BLEU score is calculated in the context of the previous set of 1-best translations of the entire tuning set.

We use Algorithm 3 to parallelize training, where we divide the tuning set into n shards and distribute them among n learners, along with the parameter vector $\mathbf{w}$. Each learner decodes and updates parameters on its shard of the tuning set, and once all learners are finished, these n parameter vectors are averaged to form the initial parameter vector for the next iteration. Learning is carried out across n=15 learners on our PBS-managed batch cluster unless otherwise noted.

3.4.2 Results

The results of using different optimization strategies for cs-en and fr-en are presented in Tables 3.3 and 3.4 below. For all experiments, all settings are kept exactly the same, with the only variation being the loss dictating the selection of the oracle $\boldsymbol{y}^+$ and prediction $\boldsymbol{y}^-$. The first column in each table indicates the method for selecting the prediction, $\boldsymbol{y}^-$. PB indicates prediction-based, MC is the hypothesis with the highest cost, and M+C is cost-augmented selection. Analogously, the headings across the table indicate oracle selection strategies, with LU indicating local updating, and M-C being cost-diminished selection.

From the cs-en results in Table 3.3, we can see that two settings fare the best: LU oracle selection paired with MC prediction selection (LU/MC), and M-C oracle selection paired with M+C prediction selection (M$\pm$C). On both sets, (M$\pm$C) performs better, but

cs-en	NT09		NT10	
Selection of $\boldsymbol{y}^-$	Selection of $\boldsymbol{y}^+$			
	LU	M-C	LU	M-C
PB	17.71	19.67	18.32	20.55
MC	19.69	17.38	20.34	19.13
M+C	18.92	**20.15**	19.42	**21.06**

Table 3.3: Results with different strategies on cs-en translation. MERT baseline is 19.92 for NT09 and 21.38 for NT10. PB indicates prediction-based, MC is the hypothesis with the highest cost, and M+C is cost-augmented selection. Analogously, the headings across the table indicate oracle selection strategies, with LU indicating local updating, and M-C being cost-diminished selection.

the results are comparable. Pairing M-C with PB is also a viable strategy, while no other pairing is successful for LU.

When comparing with MERT, note that we use a hypergraph based MERT (Kumar et al., 2009), while the MIRA updates are computed from a k-best list. For max-BLEU oracle selection paired with MC, the performance decreases substantially, to 16.91 and 18.26 BLEU on NT09 and NT10, respectively. Using the augmented k-best list did not significantly affect performance for M-C oracle selection.

For fr-en, we see much the same behavior as in cs-en. However, here LU/MC slightly outperforms M$\pm$C. From both tasks, we can see that LU is more sensitive to prediction selection, and can only optimize effectively when paired with MC. M-C on the other hand, is more forgiving, and can make progress with PB and MC, albeit not as effectively as with M+C.

fr-en	NT09		NT10	
Selection of $\boldsymbol{y}^-$	Selection of $\boldsymbol{y}^+$			
	LU	M-C	LU	M-C
PB	21.03	24.93	22.45	26.75
MC	**25.61**	24.78	**27.26**	26.36
M+C	23.77	**25.45**	25.27	**26.97**

Table 3.4: Results with different strategies on fr-en translation. MERT baseline is 26.11 for NT09 and 27.8 for NT10.

3.4.3 Sparse Feature Set

Since one of the primary motivations for large-margin learning is the ability to effectively handle large quantities of features, we further evaluate the ability of the strategies we are investigating by introducing a large number of sparse binary indicator features into our model of the form commonly found in MT research (Chiang et al., 2009; Watanabe et al., 2007). Specifically, we introduce two types of features based on word alignment from hierarchical phrase pairs and a target bigram feature. The first type, a word pair feature, fires for every word pair (e_i, f_j) observed in the phrase pair.

$$f(e_i, f_i) = \begin{cases} 1 & \text{if } f_i = \text{“rappelle” and } e_i = \text{“remind”} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

The second, insertion features, account for spurious words on the target side of a phrase pair by firing for unaligned target words, associating them with every source word, i.e. $(e_i, f_j), (e_i, f_{j+1})$,etc.:

$$f(e_i, f_0 ... f_j) = \begin{cases} 1 & \text{if } e_i = \text{“the” and} (e_i, f_0 ... f_j) \notin \mathbf{a} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The target bigram feature fires for every pair of consecutive words on the target side (e_i, e_{i+1}):

$$f(e_i, e_{i+1}) = \begin{cases} 1 & \text{if } e_i = \text{“forget” and } e_{i+1} = \text{“to”} \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

In all, we introduce 650k features for cs-en, and 1.1M for fr-en. Taking the two best performing strategies from the baseline model, LU/MC and M±C, we compare their performance with the larger feature set in Table 3.5.

	fr-en		cs-en	
	NT09	NT10	NT09	NT10
LU/MC	25.5	27.1	19.62	20.66
M±C	25.51	26.90	20.03	21.24

Table 3.5: Results on cs-en and fr-en with full larger feature set.

Although integrating these features improves the performance on the tuning set, they do not significantly alter the performance on either task on the test sets. This is in line with previous observations that dealt with sparse features on small tuning sets, which usually restrict the number of active features by only using those containing high-frequency words, resulting in approximately 10k features (Hopkins and May, 2011; Gimpel and Smith, 2012). Here, however, we optimize with up to 1 *million*, showing that margin-based optimization is practical, and still finds a comparable solution, even with

several orders of magnitude more features.

To show the utility that correctly selecting sparse features can provide for translation, we follow previous approaches and restrict lexical features according to observed frequency in the training data. Specifically, we replace the word pair feature with context-dependent word pairs for the 300 most frequent aligned word pairs (f,e) in the training corpus, which fire on triples (f,e,f_{+1}) and (f,e,f_{-1}), capturing when we see f aligned to e, with f_{+1} and f_{-1} occurring to the right or left of f, respectively:

$$f(f_{i-1}f_i, e_i) = \begin{cases} 1 & \text{if } f_i = \text{“de” and } e_i = \text{“of” and } f_{i-1} = \text{“politique”} \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

Examples of some of the top context-dependent word pairs are presented in Table 3.6. All other words fall into the default $\langle unk \rangle$ feature bin, which we have found to be an important factor. In addition, we have insertion and deletion features for the 150 most frequently unaligned target and source words, and lexical features on rules indicating the rule identity, which fires on every unique grammar rule, and serves as a discriminative analog to the phrase translation probabilities:

$$f(r) = \begin{cases} 1 & \text{if } r_f = \text{“}X_0 \text{ la position de } X_1\text{”} \text{ and } r_e = \text{“}X_0 \text{ the position of } X_1\text{”} \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

Non-lexical features include structural distortion, which captures the dependence

(de la, the)	(de l’, the)	(la commission, commission)
(, et, and)	(, je, i)	(l’ union, union)
(le président, president)	(que nous, we)	(, nous, we)
(, mais, but)	(à la, the)	(union européenne, european)
(président ,, ,)	(états membres, member)	(dans le, the)
(, la, the)	(pense que, that)	(que la, the)
(, le, the)	(que le, the)	(à l’, the)
(parce que, because)	(d’ un, a)	(que les, the)
(d’ une, a)	(, qui, which)	(l’ ue, eu)
(le parlement, parliament)	(l’ europe, europe)	(les états, states)
(c’ est, is)	(sur la, the)	(n’ est, is)
(, les, the)	(il est, is)	(le conseil, council)
(, il, it)	(sur le, the)	(parlement européen, european)
(du conseil, council)	(que l’, the)	(, comme, as)
(dans les, the)	(dans la, the)	(qu’ il, it)
(la politique, policy)	(est pas, not)	(de cette, this)
(la commission, committee)	(, l’, the)	(du parlement, parliament)
(parlement européen, parliament)	(européenne ., .)	(, dans, in)

Table 3.6: Examples of top context-dependent word pairs for fr-en.

between reordering and the size of a filler (Chiang et al., 2008b):

$$f(r) = \begin{cases} 1 & \text{if reordered and span} = 4 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

and rule shape, which bins grammar rules by their sequence of terminals and nonterminals (Simianer et al., 2012):

$$f(r) = \begin{cases} 1 & \text{if } r_f = \text{“01110” and } r_e = \text{“11010”} \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

All of these features are generated from the translation rules on the fly, and thus do not have to be stored as part of the grammar. These feature templates resulted in millions

	fr	cs
Left context pair	4919	2870
Right context pair	5417	2975
Target bigram	14508	13658
Insertion	132	131
Deletion	144	138
Rule Id	55230	62507
Rule Shape	36	37
Distortion	18	18
Total	80415	82345

Table 3.7: Active sparse features for cs-en and fr-en tuning.

	fr-en		cs-en	
	NT9	NT10	NT09	NT10
LU/MC	25.77	27.69	20.3	21.3
M±C	26.40	27.60	20.92	22.05

Table 3.8: Results on cs-en and fr-en with selected sparse feature set.

of features, of which only a fraction were active for tuning, as shown in Table 3.7.

Table 3.8 shows the substantial gains achieved across all settings when using the selected sparse feature set. The gains are up to 1 BLEU for Czech and French as compared to Tables 3.3 and 3.4, respectively..

3.5 Discussion

Although the performance of the two strategies is competitive on the evaluation sets, this does not fully convey the success of the optimization. For a more complete view of the differences between optimization strategies, we turn to Figures 3.2-3.7. Figure 3.2

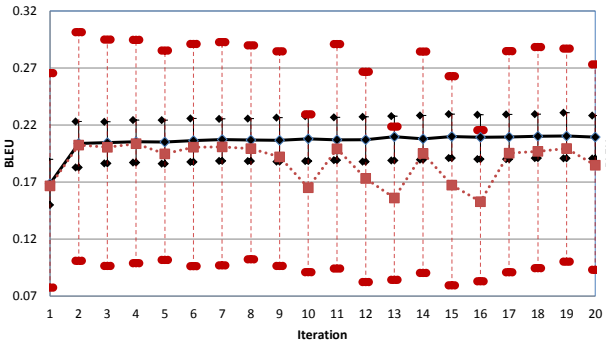


Figure 3.2: Comparison of performance on tuning set for cs-en when using LU/MC and M±C selection.

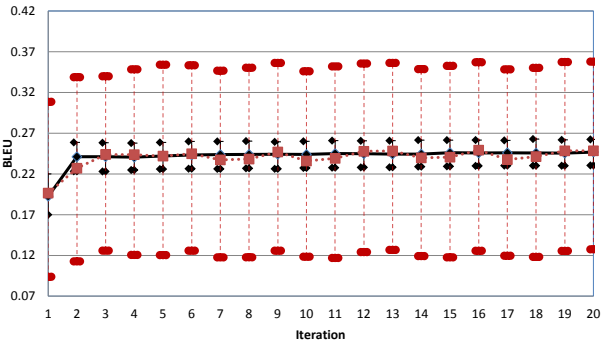


Figure 3.3: Comparison of performance on tuning set for fr-en when using LU/MC and M±C selection.

and 3.3 present the comparison of performance on the NT08 tuning set for cs-en and fr-en, respectively, when using LU/MC to select the oracle and prediction versus M±C selection. M±C is indicated with a solid black line, while LU/MC is a dotted red line. The corpus-level oracle and prediction BLEU scores at each iteration are indicated with error bars around each point, using solid lines for M±C and dotted lines for LU/MC. As can be seen in Figure 3.2, while optimizing with M±C is stable and smooth, and we converge on our optimum after several iterations, optimizing with LU/MC is highly unstable. This is at least in part due to the wide range in BLEU scores for the oracle and prediction, which are in the range of 10 BLEU points higher or lower than the current model best. On the contrary, the range of BLEU scores for the M±C optimizer is on the order of 2 BLEU points, leading to more gradual changes. We see a similar, albeit slightly less pronounced behavior on fr-en in Figure 3.3. M±C optimization is once again smooth, and converges quickly, with a small range for the oracle and prediction scores around the model best. LU/MC remains unstable, oscillating up to 2 BLEU points between iterations. The highly unstable performance of LU/MC on cs-en could be responsible for the subpar

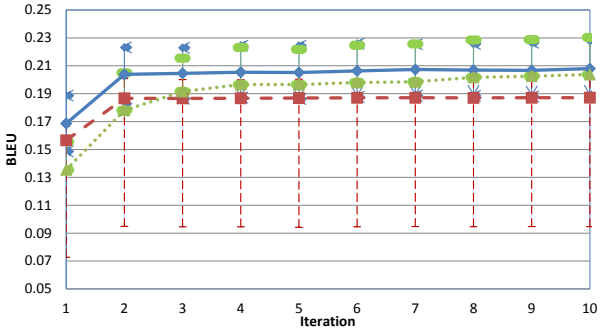


Figure 3.4: Comparison of performance on tuning set for cs-en of the three prediction selection strategies when using M-C selection as oracle.

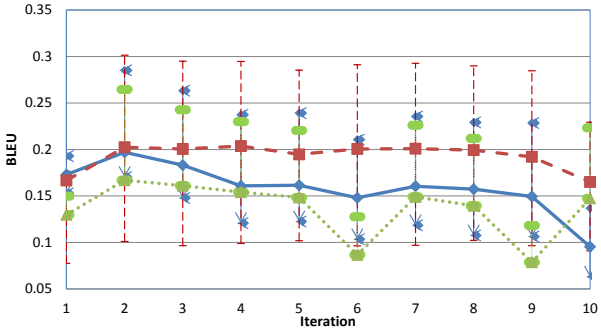


Figure 3.5: Comparison of performance on tuning set for cs-en of the three prediction selection strategies when using LU selection as oracle.

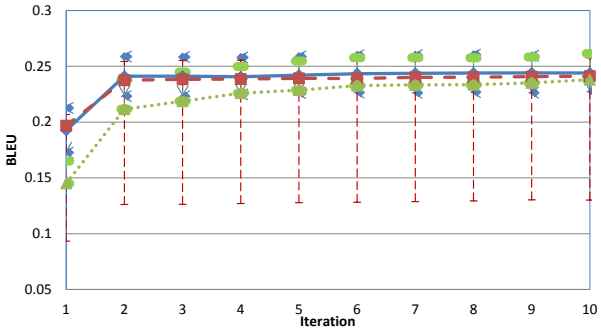


Figure 3.6: Comparison of performance on tuning set for fr-en of the three prediction selection strategies when using M-C selection as oracle.

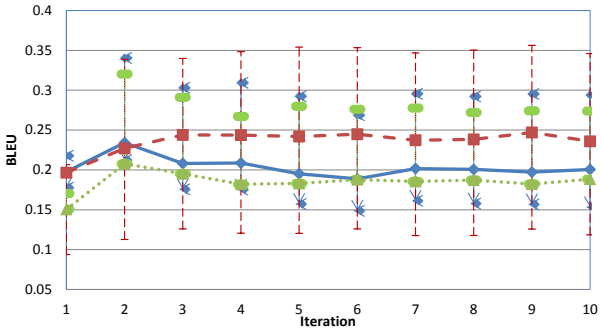


Figure 3.7: Comparison of performance on tuning set for fr-en of the three prediction selection strategies when using LU selection as oracle.

test performance compared to M±C, while the relatively smoother curve of LU/MC for fr-en produces better results.

Figures 3.4- 3.7 compare the different optimization strategies further. In Figures 3.4 and 3.6, we use M-C as the oracle, and show performance on the tuning set while using the three prediction selection strategies, M+C with a solid blue line, PB with a dotted green line, and MC with a dashed red line. Error bars indicate the oracle and prediction BLEU scores for each pairing as before. In all three cases, the oracle BLEU score is in about the same range, as expected, since all are using the same oracle selection strategy.

We can immediately observe that PB has no error bars going down, indicating that the PB method for selecting the prediction keeps pace with the model best at each iteration. On the other hand, MC selection also stands out, since it is the only one with a large drop in prediction BLEU score. Crucially, all learners are stable, and move toward convergence smoothly, which serves to validate our earlier observation that M-C oracle selection can be paired with any prediction selection strategy and optimize effectively. In both cs-en and fr-en, we can observe that M±C performs the best.

In Figures 3.5 and 3.7, we use LU as the oracle, and show performance using the three prediction selection strategies, with each line representing the same strategy as described above. The major difference, which is immediately evident, is that the optimizers are highly unstable. The only pairing which shows some stability is LU/MC, with both the other prediction selection methods, PB and M+C significantly underperforming it.

Based on the translation performance alone of optimizing the loss functions represented by LU/MC and M±C selection, it is hard to distinguish them, as they produce comparable results on the evaluation sets for fr-en and cs-en. However, taking the unstable nature of LU/MC into account, the extent of which may depend on the tuning set, the cost function itself, as well as other factors, for the rest of this thesis we will use the loss function in Eq. 3.4 that is based on selecting the oracle and prediction pair based on M±C.

3.5.1 Parallelization

As another attractive property of online learning is fast convergence and scalability, here we examine parallelization and the effect it has on both the tuning performance, learning time, and final test performance. The results in this section compare Alg. 1, i.e. a single learner, with Alg. 3 using 5 and 15 learners for fr-en. Figure 3.8 plots the performance curve of the four scenarios on the tuning set versus total wallclock time. With 15 learners, we are able to achieve good performance by the second iteration, with a total time of less than 10 minutes, and total time for all 10 iterations of less than 50 minutes. On the other hand, with a single learner we need more than 50 minutes to go through the tuning set once, and double that to achieve comparable performance.

Figure 3.9 presents performance comparison across the learners at each iteration, with error bars marking hope and fear hypothesis scores. We see that on the first iteration, fewer learners leads to slightly better performance, but this lead disappears by the second iteration. We can also notice an interesting side effect in that the spread of the hope and fear hypotheses increases with the number of learners.

Table 3.9 presents results on the evaluation sets. While a single learner performs modestly better, the parallelization does not significantly impact final performance. Thus, although parallelization and weight averaging yields different result from running a single learner over the entire data, the difference is quite small in terms of convergence and quality of tuned weights.

In future chapters, we will use Algorithm 3 to perform efficient parallel training for SMT.

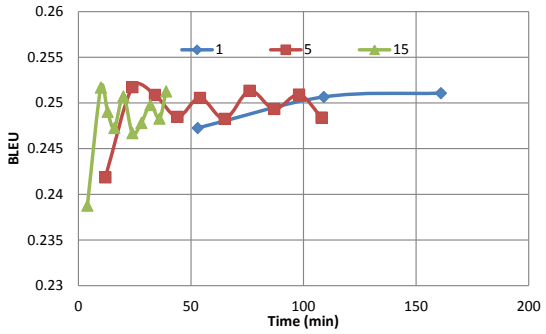


Figure 3.8: Comparison of performance vs. time on the tuning set for fr-en with 1, 5, and 15 parallel learners.

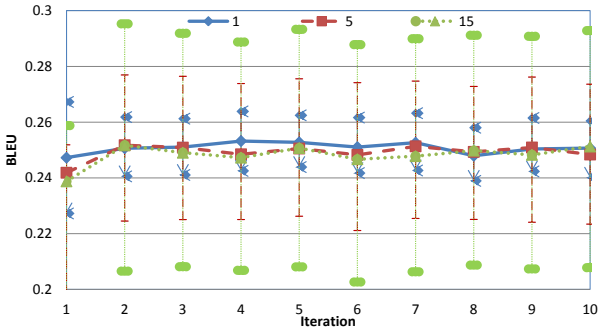


Figure 3.9: Comparison of performance on the tuning set for fr-en with 1, 5, and 15 parallel learners.

	fr-en	
learners	NT09	NT10
1	26.08	27.86
5	25.94	27.28
15	25.95	27.54

Table 3.9: Results on fr-en with different numbers of parallel learners.

3.5.2 Cutting Plane vs. Passive Aggressive Optimization

The cutting-plane method of iteratively finding closer approximations for the solution to the QP allows solving the structured prediction problem efficiently. However, it can require the ability to reweight and generate new hypotheses from the hypergraph, or re-extract the k-best, within one update iteration, which may not be feasible without tight integration with the decoder. On the other hand, the passive-aggressive update is conceptually simpler, and potentially favorable from an implementation standpoint. We thus compare the cutting plane algorithm in Alg. 5 with the passive-aggressive update presented in Alg. 4. Figure 3.10 presents the tuning performance plotted against wall-clock time, and Figure 3.11 presents performance at each iteration. The cutting plane

has slightly longer time per iteration, but not significantly different from the PA method. Furthermore, although the curve is slower to ascend, it has a more stable arc. The evaluation results in Table 3.10 reiterate the fact that the cutting plane finds a better solution, achieving better performance than the single learner above. However, the PA update is quite close, and requires much less computation per example.

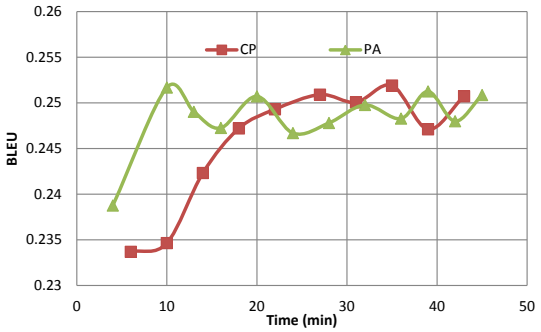


Figure 3.10: Comparison of performance vs. time on the tuning set for fr-en with cutting-plane vs. passive-aggressive learners.

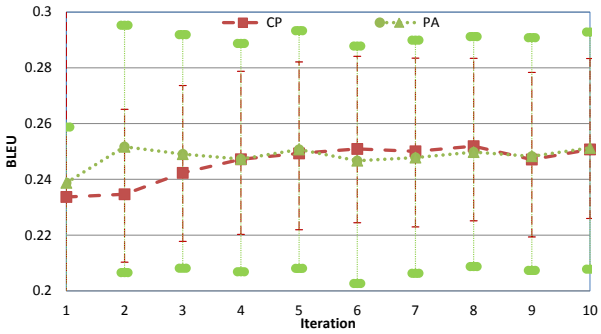


Figure 3.11: Comparison of performance on the tuning set for fr-en with cutting-plane vs. passive-aggressive learners.

	fr-en	
learners	NT09	NT10
Cutting-Plane	26.14	27.90
PA	25.95	27.54

Table 3.10: Results on fr-en with cutting-plane and passive-aggressive optimization.

Algorithm 4 MIRA Passive-Aggressive Update

procedure UPDATE($\mathbf{w}$)
  $(\boldsymbol{y}^+, \boldsymbol{d}^+) \leftarrow$ FINDORACLE($\mathcal{Y}(\boldsymbol{x}_i)$)
  $(\boldsymbol{y}^-, \boldsymbol{d}^-) \leftarrow$ FINDPREDICTION($\mathcal{Y}(\boldsymbol{x}_i)$)
  margin $\leftarrow \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-) - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+)$
  cost $\leftarrow$ BLEU$(\boldsymbol{y}_i, \boldsymbol{y}^+)$ − BLEU$(\boldsymbol{y}_i, \boldsymbol{y}^-)$
  loss = margin + cost
  if loss > 0 **then**
    $\alpha \leftarrow \min\left(C, \frac{\text{loss}}{\|\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-)\|^2}\right)$
    $\mathbf{w} \leftarrow \mathbf{w} + \eta\alpha \left(\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-)\right)$
  end if
  return w
end procedure

procedure FINDORACLE($\mathcal{Y}(\boldsymbol{x}_i)$)
  if γ^+=0 and β^+=1 **then**
    $(\boldsymbol{y}^+, \boldsymbol{d}^+) \leftarrow \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} -\Delta(\boldsymbol{y}_i, \boldsymbol{y})$ ▷ Min cost
  else if $\gamma^+ = \beta^+ = 1$ **then**
    $(\boldsymbol{y}^+, \boldsymbol{d}^+) \leftarrow \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d}) - \Delta(\boldsymbol{y}_i, \boldsymbol{y})$ ▷ Hope
  end if
  return $\boldsymbol{y}^+$
end procedure

procedure FINDPREDICTION($\mathcal{Y}(\boldsymbol{x}_i)$)
  if γ^-=0 and β^-=1 **then**
    $\boldsymbol{y}^- \leftarrow \text{argmax}_{y \in \mathcal{Y}(\boldsymbol{x}_i)}$ cost$(\boldsymbol{y}_i, \boldsymbol{y})$ ▷ Max cost
  else if γ^-=1 and β^-=0 **then**
    $(\boldsymbol{y}^-, \boldsymbol{d}^-) \leftarrow \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y})$ ▷ Prediction Based
  else if $\gamma^- = \beta^- = 1$ **then**
    $(\boldsymbol{y}^-, \boldsymbol{d}^-) \leftarrow \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x_i)} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d}) + \Delta(y_i, y)$ ▷ Max loss/Fear
  end if
  return $\boldsymbol{y}^-$
end procedure

Algorithm 5 MIRA Cutting Plane Update (based on Chiang (2012))

procedure UPDATE($\mathbf{w}$)
  $(\boldsymbol{y}^+, \boldsymbol{d}^+) \leftarrow$ FINDORACLE($\mathcal{Y}(\boldsymbol{x}_i)$)
  $S_i \leftarrow \{(\boldsymbol{y}^+, \boldsymbol{d}^+)\}$
  repeat
    $(\boldsymbol{y}^-, \boldsymbol{d}^-) \leftarrow$ FINDPREDICTION($\mathcal{Y}(\boldsymbol{x}_i)$)
    compute $\xi = \max\left\{0, \max_{(\boldsymbol{y},\boldsymbol{d}) \in S_i} \text{FINDLOSS}((\boldsymbol{y}, \boldsymbol{d}))\right\}$
    if FINDLOSS$((\boldsymbol{y}^-, \boldsymbol{d}^-)) \geq \xi + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{(\boldsymbol{y}^-, \boldsymbol{d}^-)\}$
      OPTIMIZESET($\mathbf{w}$)
    end if
  until S_i does not change
end procedure

procedure OPTIMIZESET($\mathbf{w}$)
  $\alpha_{\boldsymbol{y}} \leftarrow 0$ **for** $(\boldsymbol{y}, \boldsymbol{d}) \in S_i$
  $\alpha_{\boldsymbol{y}_i} \leftarrow C$
  iterations $\leftarrow 0$
  while iterations ≤ 100 **do**
    $(\boldsymbol{y}', \boldsymbol{d}'), (\boldsymbol{y}'', \boldsymbol{d}'') \leftarrow$ SELECTPAIR($\mathbf{w}$)
    if undefined $(\boldsymbol{y}', \boldsymbol{d}'), (\boldsymbol{y}'', \boldsymbol{d}'')$ **then**
      return
    end if
    loss $\leftarrow \Delta(\boldsymbol{y}_i, \boldsymbol{y}') - \Delta(\boldsymbol{y}_i, \boldsymbol{y}'')$
    margin $\leftarrow \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}') - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}'', \boldsymbol{d}'')$
    loss = margin + loss
    $\gamma_\alpha \leftarrow \frac{\text{loss}}{\eta ||\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}') - \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}'', \boldsymbol{d}'')||^2}$
    $\gamma_\alpha \leftarrow \max(-\alpha_{\boldsymbol{y}}, \min(\alpha_{\boldsymbol{y}'}, \gamma_\alpha))$
    $\alpha_{\boldsymbol{y}} \leftarrow \alpha_{\boldsymbol{y}} + \gamma_\alpha$
    $\alpha_{\boldsymbol{y}'} \leftarrow \alpha_{\boldsymbol{y}'} - \gamma_\alpha$
    $\mathbf{w} \leftarrow \mathbf{w} - \gamma_\alpha(\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}') - \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}'', \boldsymbol{d}''))$
  end while
end procedure

procedure SELECTPAIR($\mathbf{w}$)
  for $(\boldsymbol{y}', \boldsymbol{d}') \in S_i$ **do**
    $H_{\text{max}} \leftarrow \max_{\boldsymbol{y}'' \neq \boldsymbol{y}'}$ FINDLOSS$(\mathbf{w}, (\boldsymbol{y}'', \boldsymbol{d}''))$
    if $\alpha_{(\boldsymbol{y}', \boldsymbol{d}')} = 0$ and FINDLOSS $> H_{\text{max}} + \epsilon$ **then**
      if $\exists (\boldsymbol{y}'', \boldsymbol{d}'') \neq (\boldsymbol{y}', \boldsymbol{d}')$ such that $\alpha_{(\boldsymbol{y}'', \boldsymbol{d}'')} > 0$ **then**
        return $(\boldsymbol{y}', \boldsymbol{d}'), (\boldsymbol{y}'', \boldsymbol{d}'')$
      end if
    end if
    if $\alpha_{(\boldsymbol{y}', \boldsymbol{d}')} > 0$ and FINDLOSS$(\mathbf{w}, (\boldsymbol{y}', \boldsymbol{d}')) < H_{\text{max}} - \epsilon$ **then**
      if $\exists (\boldsymbol{y}'', \boldsymbol{d}'') \neq (\boldsymbol{y}', \boldsymbol{d}')$ such that FINDLOSS$(\mathbf{w}, (\boldsymbol{y}'', \boldsymbol{d}'')) >$ FINDLOSS$(\mathbf{w}, (\boldsymbol{y}', \boldsymbol{d}'))$ **then**
        return $(\boldsymbol{y}', \boldsymbol{d}'), (\boldsymbol{y}'', \boldsymbol{d}'')$
      end if
    end if
  end for
  return undefined
end procedure
procedure FINDLOSS$(\mathbf{w}, (\boldsymbol{y}, \boldsymbol{d}))$
  if γ^-=0 and β^-=1 **then**
    $H(\boldsymbol{y}) \leftarrow \text{cost}(y_i, y)$ ▷ Max cost
  else if γ^-=1 and β^-=0 **then**
    $H(\boldsymbol{y}) \leftarrow \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d})$ ▷ Prediction Based
  else if $\gamma^- = \beta^- = 1$ **then**
    $H(\boldsymbol{y}) \leftarrow \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d}) + \Delta(\boldsymbol{y}_i, \boldsymbol{y})$ ▷ Max loss/Fear
  end if
  return $H(\boldsymbol{y})$
end procedure

3.6 Large-Scale Discriminative Learning

In the previous sections, we empirically analyzed the PA algorithms ability to scale to high-dimensional feature spaces for SMT. Although we now possess the ability to easily handle sparse and lexicalized features, we face a new problem. Typically, a handful of parameters for dense features are tuned, along with at most a few thousand others, using a relatively small tuning set, on the order of several thousand sentences (Cherry and Foster, 2012; Chiang et al., 2009). Since these features occur often, and there are far fewer features than examples, there is plenty of support for them in the tuning sentences. However, when dealing with thousands to millions of sparse features that occur very infrequently, i.e. lexical features or rule identity, we may not find sufficient support using only a small tuning set, leading to overfitting. This predicament has driven research into scaling the size of the tuning set (Simianer et al., 2012; Flanigan et al., 2013; Yu et al., 2013). While *more* features tuned on *more* data usually results in better performance for other NLP tasks, this has not necessarily been the case for SMT.

The SGD based methods we have been discussing are especially well suited for large-scale learning, since they scale nicely with both the size of the data and parameters (Bottou and Bousquet, 2011). In the following sections, we extend our previous learner to handle *large-scale training* data in a *highly distributed setting* by developing Mr. MIRA (Eidelman et al., 2013b), an open source decoder agnostic implementation of online large-margin learning in Hadoop MapReduce.[8] Mr. MIRA separates learning from the decoder, allowing the flexibility to specify the desired inference procedure through a

[8] `https://github.com/kho/mr-mira`

simple text communication protocol. The decoder receives input sentences and weight updates from the learner. The learner only requires k-best output with feature vectors, as well as the specification of a cost function. While the current demonstrated application focuses on large-scale discriminative training for machine translation, the learning algorithm is general with respect to the inference algorithm employed. To our best knowledge, this work is the first to achieve practical large-scale large-margin learning with a passive-aggressive algorithm for SMT.[9] In so doing, we focus on improving understanding into the effective use of sparse features, and the benefits and challenges of large-scale discriminative training. Experimental results (§3.8) show that it scales linearly and makes fast parameter tuning on large tuning sets for SMT practical.

3.7 Bitext Tuning

In principle, we may want to train our system completely discriminatively on the parallel data, foregoing the construction of the previously described features in Section 2.2.4.1 based generative models entirely. In practice, however, this remains infeasible. For one, bitexts range on the order of tens of millions of sentences pairs, and tuning with them would require the repeated translation of millions of sentences – quite a task! Furthermore, as we shall see, bitexts are composed of many genres, while the tuning set is selected to match the genre of the data we want to be able to translate. Thus, tuning on the bitext may pose an additional domain mismatch problem.

Thus, the current procedure is to utilize the generative models to align the bitext

[9] There has been related previous work using MapReduce to scale training, for instance Simianer et al. (2012) used MapReduce with a perceptron algorithm for pairwise ranking.

as before, and then simply take a large portion of it to use as a tuning set. This poses another problem, however. Recall that in order to translate, we need to extract translation grammars for each sentences, containing all the translation rules which can be applied to translate the source side of the sentence, and these translation rules are extracted from the aligned parallel corpus. If the portion of the bitext we wish to use for tuning, and thus need to extract translation rules *for*, is included in the parallel corpus we extract translation rules *from*, we face a serious overfitting issue. This is because our translation model would have an abundance of rules to be able to correctly translate most of these sentences, possibly preferring longer more specialized rules that are less able to generalize.[10] While for an unseen sentence, we would have far fewer rules available, and may not have learned proper settings for them.

One solution is to construct a "leave-one-out grammar" (Simianer et al., 2012). Here, the per-sentence grammar for sentence i would only contain rules not extractable from sentence i. While effective, this solution is cumbersome, as it requires modifying how most grammar extractors work. Another simpler solution is to jackknife the data (Flanigan et al., 2013). Here, we divide the bitext into n folds, and withhold sentences from one fold to use for tuning, while aligning and extracting the grammar from the remaining n-1 folds. This results in n systems, of which the best performing one can be taken, or the final weights can be combined.

[10]There would still be issues preventing the perfect translation for all sentences; alignment errors, reordering spans longer than our maximum, etc.

3.7.1 MapReduce

Hadoop MapReduce (Dean and Ghemawat, 2004) is a popular open-source implementation of the MapReduce distributed processing framework. MapReduce has gained widespread adoption, with the advantage of providing scalable parallelization in a manageable framework, taking care of data distribution, synchronization, fault tolerance, as well as other features. This differs from previous parallelization architectures in that there is no need to explicitly manage issues related to data distribution, concurrency, and other system details. Thus, while we could otherwise achieve the same level of parallelization, it would be in a more ad-hoc manner.

A MapReduce program is divided into a `Map` function and a `Reduce` function. This mimics the common processing pipeline of many algorithms: some action is performed on each instance, the results of which are then aggregated. The *mapper* contains the action to be taken, and the *reducer* describes what is to be done with the results.

As part of the MapReduce design, the processing actually takes place over key-value pairs, which define the input and output for mappers and reducers. The entire set of input key-value pairs is split into shards, where the pairs in a shard are sent to a mapper, which process them accordingly and produces intermediate key-value pairs in parallel with other mappers. All intermediate values across mappers are then processed by the reducer according to their key to produce the output key-value pairs. We refer to the reader to Lin and Dyer (2010) for further details.

3.7.2 Algorithm Design

The advantage of online methods lies in their ability to deal with large training sets and high-dimensional input representations while remaining simple and offering fast convergence. We use Hadoop streaming to parallelize the online training process, through which our system can take advantage of commodity clusters to handle parallel large-scale training while also being capable of running on a single machine or PBS-managed batch cluster

Hadoop streaming allows any arbitrary executable to serve as the mapper or reducer, as long as it handles key-value pairs properly.[11]

A single iteration of training is performed as a Hadoop streaming job. Each begins with a *map* phase, with every parallel mapper, which is our learner from earlier, loading the same initial weights and decoding and updating parameters on a shard of the data. This is followed by a *reduce* phase, with a single reducer collecting final weights from all mappers and computing a weighted average to distribute as initial weights for the next iteration.

A straightforward MapReduce implementation of the MIRA learner accepts a single input key-value pair to the Map function. The key is a sentence identifier,[12] and the value consists of one sentence from the training data, along with its reference(s). In each mapper, the current parameter values (i.e., feature weights) are loaded into memory as an initialization step, before processing any key-value pair. After processing each sentence,

[11] By default, each line is treated as a key-value pair encoded in text, where the key and the value are separated by a `<tab>`.

[12] Sentence ids are non-negative.

the mapper will update the parameters based on the computed gradient.

Each mapper loads the same initial weights, processes a single shard of data and produces two types of intermediate key-value pairs: the one-best hypothesis of each sentence along with the sentence ID as the key (non-negative); once there are no more input lines, the final weights are output with a special negative key. In the reduce step, a single reducer collects all key-value pairs, grouped and sorted by keys. The reducer first receives the updated weights from all mappers, all grouped by key -1, from which it computes any feature selection and weighted average of final weights received from all of the mappers as before. The remaining key-value pairs are received in order of their sentence id, and the one-best hypotheses are output to disk in the order they are received, so that the order matches the reference translation set. After the reducer finishes, the averaged weights are used as the initial weights for the next iteration; the emitted hypotheses are scored against the references, which allows us to track the tuning performance and the progress of convergence.

3.7.3 Scalability

In an application such as SMT, the decoder requires access to the translation grammar and language model to produce translation hypotheses. In addition, the mapper also needs to compute the gradient value for learning, and this requires access to the reference translations. For small tuning sets, which have been typical in MT research, having these files transferred across the network to individual servers (which then load the data into memory) is not a problem. However, for even modest input on the order of tens

of thousands of sentences, this creates a challenge. For example, distributing thousands of per-sentence grammar files to all the workers in a Hadoop cluster is time-consuming, especially when this needs to be performed prior to every iteration.

To benefit from MapReduce, it is essential to avoid dependencies on "side data" as much as possible, due to the challenges explained above with data transfer. To efficiently encode the information that the learner and decoder require (source sentence, reference translation, grammar rules) in a manner amenable to MapReduce, i.e. avoiding dependencies on "side data" and large transfers across the network, we append the reference and per-sentence grammar to each input source sentence, creating a very long string as the value object.. Although this file's size is substantial (e.g., 75 gigabytes for 50,000 sentences), it is not an issue since after the initial transfer the data reside on the Hadoop distributed file system and MapReduce optimizes for data locality when scheduling mappers.

Unfortunately, it is much more difficult to obtain per-sentence language models that are small enough to handle in this same manner. Currently, the best solution we have found is to use Hadoop's distributed cache to ship the single large language model to each worker.

3.8 Evaluation

3.8.1 Setup

We evaluated online learning in Hadoop MapReduce by applying it to German-English and Russian-English machine translation, using our hierarchical phrase-based

	Dev	Test	Test2	5k	10k	25k	50k
en	75k	74k	27k	132k	255k	634k	1258k
de	74k	73k	26k	133k	256k	639k	1272k

Table 3.11: Corpus statistics in tokens for German.

	Dev	Test	Test2	15k
ru	46k	24k	24k	350k
en	50k	27k	25k	371k

Table 3.12: Corpus statistics in tokens for Russian.

translation system. The parallel training data for both languages consisted of the Europarl and News Commentary corpora from the WMT12 translation task,[13] For Russian, we additionally used the Common Crawl and Yandex data. The data were lowercased and tokenized, then filtered for length and aligned using the GIZA++ implementation of IBM Model 4 (Och and Ney, 2003) to obtain one-to-many alignments in both directions and symmetrized sing the grow-diag-final-and method (Koehn et al., 2003).

We constructed a 5-gram language model using SRILM (Stolcke, 2002) from the provided English monolingual training data and parallel data with modified Kneser-Ney smoothing Chen and Goodman (1996), which was binarized using KenLM (Heafield, 2011). The sentence-specific translation grammars were extracted using a suffix array rule extractor (Lopez, 2007).

For German, we used the 3,003 sentences in newstest2011 as our tuning set, Dev, and report results on the 3,003 sentences of the newstest2012 Test set using BLEU and TER (Snover et al., 2006). For Russian, we took the first 2,000 sentences of newstest2012 for Dev, and report results on the remaining 1,003. For both languages, we selected 1,000 sentences from the bitext to be used as an additional testing set (Test2).

We experimented with two feature sets: (1) a small set of 16 standard features; and (2) a large sparse feature setting containing the features described in Section 3.4.3.

[13] http://www.statmt.org/wmt12/translation-task.html

All experiments were conducted on a Hadoop cluster (running Cloudera's distribution, CDH 4.2.1) with 16 nodes, each with two quad-core 2.2 GHz Intel Nehalem Processors, 24 GB RAM, and three 2 TB drives. In total, the cluster is configured for a capacity of 128 parallel workers, although we do not have direct control over the number of simultaneous mappers, which depends on the number of input splits. If the number of splits is smaller than 128, then the cluster is under-utilized. To note this, we report the number of splits for each setting in our experimental results.

We ran MIRA on a number of tuning sets, described in Tables 3.11 and 3.12, in order to test the effectiveness and scalability of our system. First, we used the standard tuning set from WMT12, consisting of 3,003 sentences from news domain. In order to show the scaling characteristics of our approach, we then used larger portions of the training bitext directly to tune parameters. Since the bitext is used to learn rules for translation, using the same parallel sentences for grammar extraction as well as for tuning feature weights can lead to severe overfitting (Flanigan et al., 2013). To avoid this issue, we used a jackknifing method to split the training data into $n = 10$ folds, and built a translation system on $n - 1$ folds, while sampling sentences from the News Commentary portion of the held-out fold to obtain tuning sets ranging from 5,000 to 50,000 sentences for German, and 15,000 sentences for Russian.

Set	# features	Tune	Test	
		↑BLEU	↑BLEU	↓TER
de-en	16	22.38	22.69	60.61
+sparse	108k	23.86	**23.01**	**59.89**
ru-en	16	30.18	29.89	49.05
+sparse	77k	32.40	**30.81**	**48.40**

Table 3.13: Results with the addition of sparse features for German and Russian.

3.8.2 Results

3.8.3 Sparse Features

We first perform an evaluation to gauge the effect sparse features have for these language pairs on the typical tuning sets. To allow for memory efficiency while scaling the training data, we hash all the lexical features from their string representation into a 64-bit integer.

3.9 Adaptive Learning Rate

Altogether, these templates result in millions of potential features, thus how to select appropriate features, and how to properly learn their weights can have a large impact on the potential benefit. Figure 3.12 shows tuning performance for sparse models with a single learning rate, and adaptive learning with λ=0.01 and λ=0.1, with associated results on Test in Table 3.14.[14] As can be seen, using a single η produces almost no gain on Dev. However, while both settings using an adaptive rate fare better, the proper setting of λ is

[14] All sparse models are initialized with the same tuned baseline weights. Learning rates are local to each mapper.

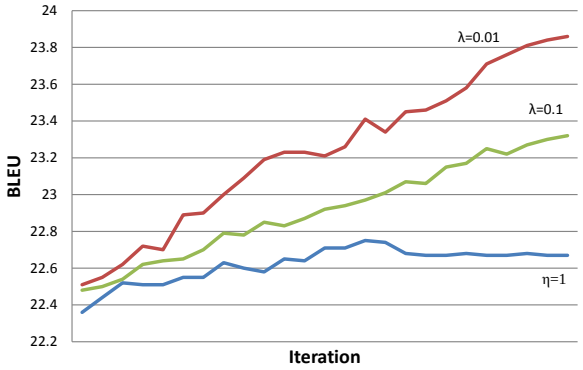


Figure 3.12: Performance on the tuning set when using a single step size (η) versus different per-feature learning rates.

Adaptive	# feat.	Tune	Test	
		↑BLEU	↑BLEU	↓TER
none	74k	22.75	22.87	60.19
λ=0.01	108k	23.86	**23.01**	**59.89**
λ=0.1	62k	23.32	22.92	60.09

Table 3.14: Results with different λ settings for using a per-feature learning rate with sparse features.

important. With λ=0.01 we observe 0.5 BLEU gain over λ=0.1 in tuning, which translates to a small gain on Test. Henceforth, we use an adaptive learning rate with λ=0.01 for all experiments.

Table 3.13 presents baseline results for both languages. With the addition of sparse features, tuning scores increase by 1.5 BLEU for German, leading to a 0.3 BLEU increase on Test, and 2.2 BLEU for Russian, with 1 BLEU increase on Test. The majority of active features for both languages are rule id (74%), followed by target bigrams (14%) and context-dependent word pairs (11%).

3.10 Feature Selection

As the tuning set size increases, so do the number of active features. This may cause practical problems, such as reduced speed of computation and memory issues. Furthermore, while some sparse features will generalize well, others may not, thereby incurring practical costs with no performance benefit. Thus, while utilizing sparse features is a primary motivation for performing large-scale discriminative training, which features to use and how to learn their weights can have a large impact on the potential benefit. To this end, several techniques have recently been explored for improved learning.

Simianer et al. (2012) recently explored ℓ_1/ℓ_2 regularization for joint feature selection for SMT in order to improve efficiency and counter overfitting effects. When performing parallel learning this allows for selecting a reduced set of the top k features at each iteration that are effective across all learners.

Table 3.15 compares selecting the top 200k features versus no selection for a larger German and Russian tuning set. As can be seen, we achieve the same performance with the top 200k features as we do when using double that amount, while the latter becomes increasing cumbersome to manage. Therefore, we use a top 200k selection for the remainder of this work.

3.11 Large-Scale Training

In the previous section, we saw that learning sparse features on the small tuning set leads to substantial gains in performance. Next, we wanted to evaluate if we can obtain

Set	# feat.	Tune	Test	
		↑BLEU	↑BLEU	↓TER
all	510k	32.99	22.36	59.26
top 200k	200k	32.96	22.35	59.29
all	373k	34.26	28.84	49.29
top 200k	200k	34.45	**28.98**	49.30

Table 3.15: Comparison of using all features versus top k selection.

further gains by scaling the tuning data to learn parameters directly on a portion of the training bitext. Results for large-scale training for German are presented in Table 3.16. The second column shows the space each tuning set takes up on disk when we include reference translations and grammar files along with the sentences. The reported tuning BLEU is from the iteration with best performance, and running times are reported from the top-scoring iteration as well.

Although we cannot compare the tuning scores across different size sets, we can see that tuning scores for all sets improve substantially with sparse features. Unfortunately, with increasing tuning set size, we see very little improvement in Test BLEU and TER with either feature set. Similar findings for Russian are presented in Table 3.17. Introducing sparse features improves translation performance on each set, respectively, but tuning on Dev always performs better on Test.

While tuning on Dev data results in better BLEU on Test than when tuning on the larger sets, it is important to note that although we are able to tune more features on the larger bitext tuning sets, they are not composed of the same genre as the Tune and Test sets, resulting in a domain mismatch.

This phenomenon is further evident in German when testing each model on Test2,

which is selected from the bitext, and is thus closer matched to the larger tuning sets, but is separate from both the parallel data used to build the translation model and the tuning sets. Results on Test2 clearly show significant improvement using any of the larger tuning sets versus Dev for both the baseline and sparse features. The 50k sparse setting achieves almost 1 BLEU and 2 TER improvement, showing that there are significant differences between the Dev/Test sets and sets drawn from the bitext.

For Russian, we amplified the effects by selecting Test2 from the portion of the bitext that is separate from the tuning set, but is among the sentences used to create the translation model. The effects of overfitting are markedly more visible here, as there is almost a 7 BLEU difference between tuning on Dev and the 15k set with sparse features. Furthermore, it is interesting to note when looking at Dev that using sparse features has a significant negative impact, as the baseline tuned Dev performs reasonably well, while the introduction of sparse features leads to overfitting the specificities of the Dev/Test genre, which are not present in the bitext.

We attempted two strategies to mitigate this problem: combining the Dev set with the larger bitext tuning set from the beginning, and tuning on a larger set to completion, and then running 2 additional iterations of tuning on the Dev set using the learned model. The resulting model improves somewhat on the other genre and strikes a middle ground, although it is worse on Test than Dev.

Table 3.18 presents results for tuning several additional iterations after learning a model on the larger sets. Although this leads to gains of around 0.5 BLEU on Test, none of the models outperform simply tuning on Dev. Thus, neither of these two strategies seem to help. In future work, we plan to forgo randomly sampling the tuning set from the

bitext, and instead actively select the tuning set based on similarity to the test set.

Tuning set size		Time/iter	# splits	# feat	Tune BLEU	Test		Test2	
(corpus)	(on disk, GB)	(sec)			↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
dev	3.3	119	120	16	22.38	22.69	60.61	29.31	54.26
5k	7.8	289	120	16	32.60	22.14	59.60	29.69	52.96
10k	15.2	432	120	16	33.16	22.06	59.43	29.93	52.37
25k	37.2	942	300	16	32.48	22.21	59.54	30.03	51.71
50k	74.5	1802	600	16	32.21	22.21	59.39	29.94	52.55
dev	3.3	232	120	85k	23.08	23.00	60.19	29.65	53.86
5k	7.8	610	120	159k	33.70	22.26	59.26	30.53	51.84
10k	15.2	1136	120	200k	34.00	22.12	59.24	30.51	51.71
25k	37.2	2395	300	200k	32.96	22.35	59.29	30.39	52.14
50k	74.5	4465	600	200k	32.86	22.40	59.15	30.54	51.88

Table 3.16: Evaluation of our Hadoop implementation of MIRA, showing running time as well as BLEU and TER values for tuning and testing data.

3.11.0.1 Runtime

In terms of running time, we observe that the algorithm scales linearly with respect to input size, regardless of the feature set. With more features, running time increases due to a more complex translation model, as well as larger intermediate output (i.e., amount of information passed from mappers to reducers). The scaling characteristics point out the strength of our system: our scalable MIRA implementation allows one to tackle learning problems where there are many parameters, but also many training instances.

Comparing the wall clock time of parallelization with Hadoop to the standard mode of 10–20 learner parallelization (Haddow et al., 2011; Chiang et al., 2009), for the small 25k feature setting, after one iteration, which takes 4625 seconds using 15 learners on

Tuning	# mappers	# features	Tune	Test		Test2	
			↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
Dev	120	16	30.18	29.89	49.05	57.14	32.56
15k	200	16	34.65	28.60	49.63	59.64	30.65
Dev+15k	200	16	33.97	28.88	49.37	58.24	31.81
Dev	120	77k	32.40	**30.81**	**48.40**	52.90	36.85
15k	200	200k	35.05	28.34	49.69	59.81	30.59
Dev+15k	200	200k	34.45	28.98	49.30	57.61	32.71

Table 3.17: Russian evaluation with large-scale tuning, showing numbers of mappers employed, number of active features for best model, and test scores on Test and bitext Test2 domains.

Tuning	Test	
	↑BLEU	↓TER
5k	22.81	59.90
10k	22.77	59.78
25k	22.88	59.77
50k	22.86	59.76

Table 3.18: Results for German with 2 iterations of tuning on Dev after tuning on larger set.

our PBS cluster, the tuning score is 19.5 BLEU, while in approximately the same time, we can perform five iterations with Hadoop and obtain 30.98 BLEU. While this is not a completely fair comparison, as the two clusters utilize different resources and the number of learners, it suggests the practical benefits that Hadoop can provide. Although increasing the number of learners on our PBS cluster to the number of mappers used in Hadoop would result in roughly equivalent performance, arbitrarily scaling out learners on the PBS cluster to handle larger training sets can be challenging since we would need to manually coordinate the parallel processes in an ad-hoc manner.

3.12 Summary

In this chapter, we extensively empirically analyzed aspects of large-margin structured learning with concrete application to the MT setting. Towards the goal of establishing what to optimize in SMT, we defined a family of objective functions for large-margin learning for structured prediction problems involving latent variables and cost-augmented hypothesis selection. We presented the MIRA passive-aggressive algorithm for optimizing these objectives in SMT. By investigating the optimization performance in standard and high-dimensional feature spaces, we showed that this method can be used directly to effectively tune a statistical MT system with millions of parameters. With extensive empirical evaluation, we evaluated the tradeoffs made by parallelization, cutting-plane versus passive-aggressive optimization, and identified best practices for the cost-augmented and diminished objective function to optimize for SMT.

Extending large-margin learning to large-scale training tasks, we developed an open-source distributed large-margin learner on MapReduce. Parallelizing the learning beyond the usual 15-20 learners, we were able to achieve speedups in convergence and make large-scale learning practical, although without yet being able to improve translation performance.

4 Online Relative Margin Maximization

> *The hypotheses we accept ought to explain phenomena which we have observed. But they ought to do more than this: our hypotheses ought to foretell phenomena which have not yet been observed.*
>
> — William Whewell

In this chapter, we focus on how to improve large-margin optimization by introducing a novel *online learning algorithm for relative margin maximization* suitable for SMT. Recent advances in large-margin learning have shown that better generalization can be achieved by incorporating higher order information into the optimization. However, these solutions are impractical in complex structured prediction problems such as statistical machine translation. We present an online gradient-based algorithm for relative margin maximization, which *bounds the spread* of the projected data while *maximizing the margin*. We evaluate our optimizer on Chinese-English and Arabic-English translation tasks, each with small and large feature sets, and show that our learner is able to achieve significant improvements of 1.2-2 BLEU and 1.7-4.3 TER on average over state-of-the-art optimizers with the large feature set.[1]

[1] This chapter is based on material originally published in Eidelman et al. (2013a).

4.1 Introduction

In every SMT system, and in machine learning in general, the goal of learning is to find a model that generalizes well, i.e. one that will yield good translations for previously unseen sentences. However, while in Chapter 3 we showed that we can incorporate high-dimensional sparse features, as the dimension of the feature space increases, generalization becomes increasingly difficult. Since only a small portion of all (sparse) features may be observed in a relatively small fixed set of instances during tuning, we are prone to overfit the training data. An alternative approach for solving this problem is estimating discriminative feature weights directly on the training bitext (Tillmann and Zhang, 2006; Blunsom et al., 2008b; Simianer et al., 2012), which is usually substantially larger than the tuning set. While we have also examined this approach in Section 3.6, that is complementary to our goal here of better generalization given a fixed size tuning set.

In order to achieve that goal, we need to carefully choose what objective to optimize, and how to perform parameter estimation of $\mathbf{w}$ for this objective. We have already established the first part in Chapter 3, and will continue using our chosen large-margin criterion as the objective, since this criterion performs well in practice for SMT at finding a linear separator using the passive-aggressive algorithm in high-dimensional feature spaces. Thus, now we focus on the second part of how to best optimize that objective. While MIRA improves the feature capabilities over MERT, its generalization capabilities are still less than optimal. The theoretical generalization ability of these learners is also related to some measure of the *spread* of the data, a notion that has been receiving increasing attention.

Recent advances in machine learning have shown that the generalization ability of these learners can be improved by utilizing second order information, as in the Second Order Perceptron (Cesa-Bianchi et al., 2005), Gaussian Margin Machines (Crammer et al., 2009b), confidence-weighted learning (Dredze and Crammer, 2008), and AROW (Crammer et al., 2009a; Chiang, 2012).

Motivated by improvements of higher-order information, Relative Margin Machines (RMM) (Shivaswamy and Jebara, 2009b) were introduced as an effective and less computationally expensive way to incorporate the *spread* of the data – information about the distance between hypotheses when projected onto the line defined by the weight vector $\mathbf{w}$. The motivation for this work has been that the margin should not be learned in isolation, but instead with relation to the spread of the data.

In essence, structured prediction with relative margin is a generalization of Structured SVM (Tsochantaridis et al., 2004), which shares its underlying objective with MIRA (Martins et al., 2010). Therefore, our intuition is that SMT should benefit from relative margin maximization as well, compared with current large-margin-based tuning methods.

Unfortunately, not all advances in machine learning are easy to apply to structured prediction problems such as SMT; the latter often involve latent variables and surrogate references, resulting in loss functions that have not been well explored in machine learning (McAllester and Keshet, 2011; Gimpel and Smith, 2012). Although Shivaswamy and Jebara (2009a) extended RMM to handle sequence labeling, their batch approach to quadratic optimization, using existing off-the-shelf QP solvers, does not provide a practical solution: as Taskar et al. (2006) observe, "off-the-shelf QP solvers tend to scale poorly with problem and training sample size" for structured prediction problems. This motivates

an online gradient-based optimization approach—an approach that is particularly attractive because its simple update is well suited for efficiently processing structured objects with sparse features (Crammer et al., 2012).

The contributions of this chapter include **(1)** introduction of a loss function for structured RMM in the SMT setting, with surrogate reference translations and latent variables; **(2)** an online gradient-based solver, RM, with a closed-form parameter update to optimize the relative margin loss; and **(3)** an efficient implementation that integrates well with the open source cdec SMT system (Dyer et al., 2010).[2] In addition, **(4)** as our solution is not dependent on any specific QP solver, it can be easily incorporated into practically any gradient-based learning algorithm.

First, we introduce RMM (§4.2.1) and propose a latent structured relative margin objective which incorporates cost-augmented hypothesis selection and latent variables. Then, we derive a simple closed-form online update necessary to create a large margin solution while simultaneously bounding the spread of the projection of the data (§4.2.2). Chinese-English translation experiments show that our algorithm, RM, significantly outperforms strong state-of-the-art optimizers, in both a basic feature setting and high-dimensional (sparse) feature space (§4.3). Additional Arabic-English experiments further validate these results, even where previously MERT was shown to be advantageous (§4.4). Finally, we discuss the spread and other key issues of RM (§4.5), and conclude with discussion of future work (§4.6).

[2] https://github.com/veidel/cdec

4.2 The Relative Margin Machine in SMT

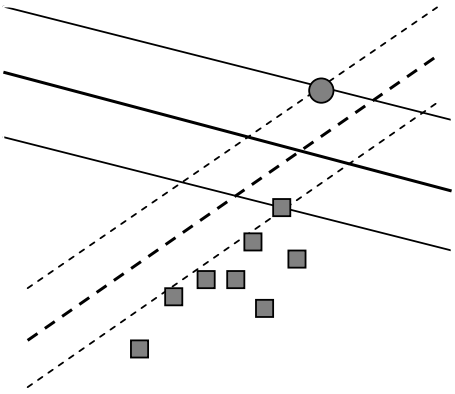
4.2.1 Relative Margin Machine

The margin, the distance between the correct hypothesis and incorrect one, is defined by score$(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+)$ and score$(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}^-)$. It is maximized by minimizing the norm in SVM, or analogously, the proximity constraint in MIRA: $\text{argmin}_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w_t}||^2$. However, theoretical results supporting large-margin learning, such as the VC-dimension (Vapnik, 1995) or the Rademacher bound (Bartlett and Mendelson, 2003) consider measures of complexity, in addition to the empirical performance, when describing future predictive ability. The measures of complexity usually take the form of some value on the radius of the data, such as the ratio of the radius of the data to the margin (Shivaswamy and Jebara, 2009a). This shows that there is a dependence of generalization error not just on margin, but on the radius of the data. As radius is a way of measuring spread in any projection direction, here we will specifically be interested in the the spread of the data as measured after the projection defined by the learned model $\mathbf{w}$.

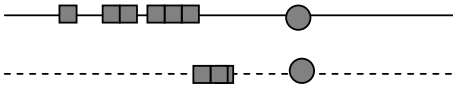
More formally, the *spread* is the distance between $\boldsymbol{y}^+$, and the worst candidate

$$(\boldsymbol{y}^w, \boldsymbol{d}^w) \leftarrow \underset{(\boldsymbol{y},\boldsymbol{d}) \in \mathcal{Y}(\boldsymbol{x}_i), \mathcal{D}(\boldsymbol{x}_i)}{\text{argmin}} \text{score}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d}),$$

after projecting both onto the line defined by the weight vector $\mathbf{w}$. For each $\boldsymbol{y}'$, this projection is conveniently given by score$(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d})$, thus the spread is calculated as δscore$(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y}^w)$, where $\delta\text{score}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y}') = \delta\text{s}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y}') = \text{score}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+) - \text{score}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}')$.



(a) RM and large margin linear separators for two-dimensional example. The RM solution is shown with a darker dotted line, while the large margin solution is shown with a darker solid line. Parallel lighter lines indicate margins.



(b) Spread of the projections given by the large-margin and RM solutions above.

Figure 4.1: (a) RM and large margin solution comparison and (b) the spread of the projections given by each.

Since generalization bounds for future performance of a large-margin classifier include the ratio of the radius of the data to the margin, RMM was introduced as a generalization over SVM that incorporates both the margin constraint and information regarding the spread of the data. This is still a large margin solution, but now focused on creating the optimal margin in the *proper* direction. The relative margin is the ratio of the absolute, or maximum margin, to the spread of the projected data. Thus, the RMM learns a large margin solution relative to the spread of the data, or in other words, geometrically creates a max margin while simultaneously bounding the spread of the projected data.

As a concrete example, consider the plot shown in Figure 4.1(a), with hypotheses

represented by two-dimensional feature vectors. The point marked with a circle in the upper right represents $\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+ \boldsymbol{d}^+)$, while all other squares represent alternative incorrect hypotheses $\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d})$. The large margin decision boundary is shown with a darker solid line, while the relative margin solution is shown with a darker dotted line. The lighter lines parallel to each define the margins, with the square at the intersection being $\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-)$. The bottom portion of Figure 4.1(b) presents an alternative view of each solution, showing the projections of the hypotheses given the learned model for the large margin on top, and relative margin on the bottom. Notice that with a large margin solution, although the distance between $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$ is greater, the points are highly spread, extending far to the left of the decision boundary.

In contrast, with a relative margin, although we have a smaller absolute margin, the spread is smaller, all points being within a smaller distance ϵ of the decision boundary. Graphically, when all data points are projected onto a line, the relative margin solution bounds how far away the leftmost and rightmost projected points are from one another.[3] The higher the spread of the projection, the higher the variance of the projected points, and the greater the likelihood that we will mislabel a new instance, since the high variance projections may cross the learned decision boundary. In higher dimensions, accounting for the spread becomes even more crucial, as will be discussed in Section 4.5.1.

[3]The motivation for controlling spread in RM is related to linear discriminant analysis (LDA) (Bishop, 2006), where LDA tries to create a large inter-class separation between classes, but decrease the intra-class spread, the distance between instances of the same class. While we can see in Fig. 4.1(b) that the RM solution decreases the overall spread, without explicitly trying to decrease intra-class spread, when taking the bounding and margin constraints together, we can see that RM achieves substantially the same effect as LDA. Considering all incorrect hypotheses as one class, and the correct hypothesis as the other, the margin constraint creates a large inter-class separating distance between the two classes, while the bounding constraint decreases the intra-class spread of the instances of the incorrect class by pushing them together as a consequence of decreasing the overall spread. The motivation of confidence-weighted estimation Dredze and Crammer (2008) and AROW Crammer et al. (2009a) is also related in spirit. They use second-order information in the form of a distribution over weights to change the maximum margin solution.

Although RMM is theoretically well-founded and improves practical performance over large-margin learning in the settings where it was introduced, it is unsuitable for most complex structured prediction in NLP. Large-margin learners in SMT have not previously accounted for the spread of the data, and only optimize the large margin criterion. Furthermore, online optimization may be preferable from the computational perspective, as we have shown that a simple update is well suited for processing structured objects with sparse features efficiently. To take advantage of the information RMM utilizes for increased generalizability in SMT, we need a computationally efficient optimization procedure that does not require batch training or an off-the-shelf QP solver.

4.2.2 RM Algorithm

We address the above-mentioned limitations by introducing a novel online learning algorithm for relative margin maximization, RM. The relative margin solution is obtained by maximizing the same margin as Eq. 2.26, but now with respect to the distance between $\boldsymbol{y}^+$, and the worst candidate $\boldsymbol{y}^w$. Thus, the relative margin dictates trading-off between a large margin as before, and a small spread of the projection, in other words, bounding the distance between $\boldsymbol{y}^+$ and $\boldsymbol{y}^w$. The additional computation required, namely, obtaining $\boldsymbol{y}^w$, is efficient to perform, and has likely already happened while obtaining the k-best derivations necessary for the margin update. The online latent structured soft relative

margin optimization problem is then:

$$\begin{aligned}
\mathbf{w_{t+1}} &= \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w} - \mathbf{w_t}||^2 + C\xi_i + D\tau_i \\
\text{s.t.:}\ \ & \delta\text{score}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y}^-) \geq \Delta_i(\boldsymbol{y}^-) - \Delta_i(\boldsymbol{y}^+) - \xi_i \\
& -B - \tau_i \leq \delta\text{score}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y}^w) \leq B + \tau_i
\end{aligned} \tag{4.1}$$

where $\Delta_i(\boldsymbol{y})$ is the cost $\Delta(\boldsymbol{y}_i, \boldsymbol{y})$, and additional bounding constraints are added to the usual margin constraints in order to contain the spread by bounding the difference in projections. B is an additional parameter; it controls the spread, trading off between margin maximization and spread minimization. Notice that when $B \to \infty$, the bounding constraints disappear, and we are left with the original problem in Eq. 2.26. D, which plays an analogous role to C, allows penalized violations of the bounding constraints.

The dual of Equation (4.1) can be derived as:

$$\begin{aligned}
\max_{\alpha,\beta,\beta^*} \mathcal{L} = & \sum_{y \in \mathcal{Y}(x_i)} \alpha_y - B \sum_{y \in \mathcal{Y}(x_i)} \beta_y - B \sum_{y \in \mathcal{Y}(x_i)} \beta^*_y \\
& -\frac{1}{2}\Big\langle \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \alpha_{\boldsymbol{y}} \omega_i(\boldsymbol{y}^+, \boldsymbol{y}) - \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \beta_{\boldsymbol{y}} \omega_i(\boldsymbol{y}^+, \boldsymbol{y}) + \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \beta^*_{\boldsymbol{y}} \omega_i(\boldsymbol{y}^+, \boldsymbol{y}), \\
& \quad \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_j)} \alpha_{\boldsymbol{y}'} \omega_j(\boldsymbol{y}^+, \boldsymbol{y}') - \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_j)} \beta_{\boldsymbol{y}'} \omega_j(\boldsymbol{y}^+, \boldsymbol{y}') + \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_j)} \beta^*_{\boldsymbol{y}'} \omega_j(\boldsymbol{y}^+, \boldsymbol{y}') \Big\rangle
\end{aligned} \tag{4.2}$$

where the α Lagrange multiplier corresponds to the standard margin constraint, while β and β^* each correspond to a bounding constraint, and $\omega_i(\boldsymbol{y}^+, \boldsymbol{y}')$ corresponds to the difference of $\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}^+)$ and $\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}')$. The weight update can then be obtained

from the dual variables:

$$\sum \alpha_{\boldsymbol{y}} \omega_i(\boldsymbol{y}^+, \boldsymbol{y}) - \sum \beta_{\boldsymbol{y}} \omega_i(\boldsymbol{y}^+, \boldsymbol{y}) + \sum \beta^*_{\boldsymbol{y}} \omega_i(\boldsymbol{y}^+, \boldsymbol{y}) \tag{4.3}$$

The dual in Equation (4.2) can be optimized using a cutting plane algorithm similar to the one described in Section 3.3.3, an effective method for solving a relaxed optimization problem in the dual, used in Structured SVM, MIRA, and RMM (Tsochantaridis et al., 2004; Chiang, 2012; Shivaswamy and Jebara, 2009a). The cutting plane presented in Alg. 6 decomposes the overall problem into subproblems which are solved independently by creating working sets S_i^j, which correspond to the largest violations of either the margin constraint, or bounding constraints, and iteratively satisfying the constraints in each set.

The cutting plane in Alg. 6 makes use of the the closed-form gradient-based updates we derived for RM presented in Alg. 7. The updates amount to performing a subgradient descent step to update $\mathbf{w}$ in accordance with the constraints. Since the constraint matrix of the dual program is not strictly decomposable across constraint types, we are in effect solving an approximation of the original problem.

Alternatively, we could utilize a passive-aggressive updating strategy (Crammer et al., 2006), which would simply bypass the cutting plane and select the most violated constraint for each set, if there is one, and perform the corresponding parameter updates in Alg. 7. We refer to the resulting passive-aggressive algorithm as RM-PA, and the cutting plane version as RM-CP. Preliminary experiments showed that RM-PA performs on par with RM-CP, thus RM-PA is the one used in the empirical evaluation below.

Algorithm 6 RM Cutting Plane Algorithm (adapted from Shivaswamy and Jebara (2009a))

Require: i^{th} training example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, weight $\mathbf{w}$, margin reg. C, bound B, bound reg. D, ϵ, ϵ_B

$S_i^1 \leftarrow \{\boldsymbol{y}^+\}$, $S_i^2 \leftarrow \{\boldsymbol{y}^+\}$, $S_i^3 \leftarrow \{\boldsymbol{y}^+\}$
repeat
  $H(\boldsymbol{y}) := \Delta_i(\boldsymbol{y}) - \Delta_i(\boldsymbol{y}^+) - \delta\mathrm{s}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y})$
  $\boldsymbol{y}_1 \leftarrow \mathrm{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} H(\boldsymbol{y})$
  $\boldsymbol{y}_2 \leftarrow \mathrm{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} G(\boldsymbol{y}) := \delta\mathrm{s}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y})$
  $\boldsymbol{y}_3 \leftarrow \mathrm{argmin}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} -G(\boldsymbol{y})$
  $\xi \leftarrow \max\{0, \max_{y \in S_i} H(y)\}$
  $V_1 \leftarrow H(\boldsymbol{y}_1) - \xi - \epsilon$
  $V_2 \leftarrow G(\boldsymbol{y}_2) - B - \epsilon_B$
  $V_3 \leftarrow -G(\boldsymbol{y}_3) - B - \epsilon_B$
  $j \leftarrow \arg\max_{j' \in \{1,2,3\}} V_{j'}$
  if $V_j > 0$ **then**
    $S_i^j \leftarrow S_i^j \cup \{\boldsymbol{y}_j\}$
    OPTIMIZE($\mathbf{w}, S_i^1, S_i^2, S_i^3, C, B$) ▷ see Alg. 7
  end if
until S_i^1, S_i^2, S_i^3 do not change

Furthermore, as all the computation necessary for the second set of bounding constraints is carried out for the normal margin constraints, it has the same complexity per instance as the normal large margin solution.

A graphical depiction of the passive-aggressive RM update is presented in Figure 4.2. The upper right circle represents $\boldsymbol{y}^+$, while all other squares represent alternative hypotheses $\boldsymbol{y}'$. As in the standard MIRA solution, we select the maximum margin constraint violator, $\boldsymbol{y}^-$, shown as the triangle, and update such that the margin is greater than the cost. Additionally, we select the maximum bounding constraint violator, $\boldsymbol{y}^w$, shown as the upside-down triangle, and update so the distance from $\boldsymbol{y}^+$ is no greater than B.

Algorithm 7 RM update with α, β, β^*

procedure OPTIMIZE($\mathbf{w}, S_i^1, S_i^2, S_i^3, C, B$)
  while $\mathbf{w}$ changes **do**
    if $\left|S_i^1\right| > 1$ **then**
      UPDATEMARGIN($\mathbf{w}, S_i^1, C$)
    end if
    if $\left|S_i^2\right| > 1$ **then**
      UPDATEUPPERBOUND($\mathbf{w}, S_i^2, B$)
    end if
    if $\left|S_i^3\right| > 1$ **then**
      UPDATELOWERBOUND($\mathbf{w}, S_i^3, B$)
    end if
  end while
end procedure
procedure UPDATEMARGIN($\mathbf{w}, S_i^1, C$)
  $\alpha_{\boldsymbol{y}} \leftarrow 0$ **for all** $\boldsymbol{y} \in S_i^1$
  $\alpha_{\boldsymbol{y}_i^+} \leftarrow C$
  for $n \leftarrow 1...MaxIter$ **do**
    Select two constraints $\boldsymbol{y}, \boldsymbol{y}'$ from S_i^1
    $\gamma_\alpha \leftarrow \frac{\Delta_i(\boldsymbol{y}') - \Delta_i(\boldsymbol{y}) - \delta s(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{y}')}{||\omega(\boldsymbol{y}, \boldsymbol{y}')||^2}$
    $\gamma_\alpha \leftarrow \max(-\alpha_{\boldsymbol{y}}, \min(\alpha_{\boldsymbol{y}'}, \gamma_\alpha))$
    $\alpha_{\boldsymbol{y}} \leftarrow \alpha_{\boldsymbol{y}} + \gamma_\alpha \; ; \quad \alpha'_{\boldsymbol{y}} \leftarrow \alpha'_{\boldsymbol{y}} - \gamma_\alpha$
    $\mathbf{w} \leftarrow \mathbf{w} + \gamma_\alpha(\omega(\boldsymbol{y}, \boldsymbol{y}'))$
  end for
end procedure
procedure UPDATEUPPERBOUND($\mathbf{w}, S_i^2, B$)
  $\beta_{\boldsymbol{y}} \leftarrow 0$ **for all** $\boldsymbol{y} \in S_i^2$
  for $n \leftarrow 1...MaxIter$ **do**
    Select one constraint $\boldsymbol{y}$ from S_i^2
    $\gamma_\beta \leftarrow \max(0, \frac{B - \delta s(x_i, y^+, y)}{||\omega(y^+, y)||^2})$
    $\beta_{\boldsymbol{y}} \leftarrow \beta_{\boldsymbol{y}} + \gamma_\beta$
    $\mathbf{w} \leftarrow \mathbf{w} - \gamma_\beta(\omega(\boldsymbol{y}^+, \boldsymbol{y}))$
  end for
end procedure
procedure UPDATELOWERBOUND($\mathbf{w}, S_i^3, B$)
  $\beta^*_{\boldsymbol{y}} \leftarrow 0$ **for all** $\boldsymbol{y} \in S_i^3$
  for $n \leftarrow 1...MaxIter$ **do**
    Select one constraint $\boldsymbol{y}$ from S_i^3
    $\gamma_{\beta^*} \leftarrow \max(0, \frac{-B - \delta s(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{y})}{||\omega(\boldsymbol{y}^+, \boldsymbol{y})||^2})$
    $\beta^*_{\boldsymbol{y}} \leftarrow \beta^*_{\boldsymbol{y}} + \gamma_{\beta^*}$
    $\mathbf{w} \leftarrow \mathbf{w} + \gamma_{\beta^*}(\omega(\boldsymbol{y}^+, \boldsymbol{y}))$
  end for
end procedure

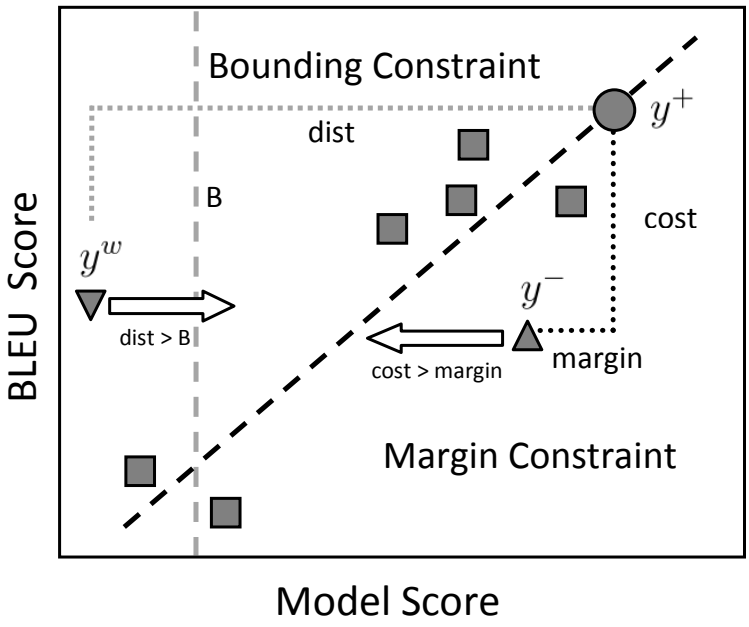


Figure 4.2: RM update with margin and bounding constraints. The diagonal dotted line depicts cost–margin equilibrium. The vertical gray dotted line depicts the bound B. White arrows indicate updates triggered by constraint violations. Squares are data points in the k-best list not selected for update in this round.

4.3 Experiments

4.3.1 Setup

To evaluate the advantage of explicitly accounting for the spread of the data, we conducted several experiments on two Chinese-English translation test sets, using two different feature sets in each. For training we used the non-UN and non-HK Hansards portions of the NIST training corpora, which was segmented using the Stanford segmenter (Tseng et al., 2005). The data statistics are summarized in the top half of Table 4.1. The English data was lowercased, tokenized and aligned using GIZA++ (Och and Ney, 2003) to obtain bidirectional alignments, which were symmetrized using the `grow-diag-final-and` method (Koehn et al., 2003). We trained a 4-gram LM on

task	Corpus	Sentences	Tokens	
			En	Zh/Ar
Zh-En	training	1.6M	44.4M	40.4M
	tune (MT06)	1664	48k	39k
	MT03	919	28k	24k
	MT05	1082	35k	33k
Ar-En	training	1M	23.7M	22.8M
	tune (MT06)	1797	55k	49k
	MT05	1056	36k	33k
	MT08	1360	51k	45k
	4-gram LM	24M	600M	–

Table 4.1: Corpus statistics.

the English side of the corpus with additional words from non-NYT and non-LAT, randomly selected portions of the Gigaword v4 corpus, using modified Kneser-Ney smoothing (Chen and Goodman, 1996). We used cdec as our hierarchical phrase-based decoder, and tuned the parameters of the system to optimize BLEU on the NIST MT06 corpus.

We applied several competitive optimizers as baselines: hypergraph-based MERT (Kumar et al., 2009), k-best variants of MIRA (Crammer et al., 2006; Chiang et al., 2009), PRO (Hopkins and May, 2011), and RAMPION (Gimpel and Smith, 2012). The size of the k-best list was set to 500 for RAMPION, MIRA and RM, and 1500 for PRO, with both PRO and RAMPION utilizing k-best aggregation across iterations. RAMPION settings were as described in Gimpel and Smith (2012), and PRO settings as described in Hopkins and May (2011), with PRO requiring regularization tuning in order to be competitive with the other optimizers. MIRA and RM were run with 15 parallel learners using Alg. 3. All optimizers were implemented in cdec and use the same system configuration, thus the only independent variable is the optimizer itself. We set C to 0.01, and $MaxIter$ to 100. We selected the bound step size D, based on performance on a held-out dev set, to be

0.01 for the basic feature set and 0.1 for the sparse feature set. The bound constraint B was set to 1.[4] The approximate sentence-level BLEU cost Δ_i is computed in a manner similar to Chiang et al. (2009), namely, in the context of previous 1-best translations of the tuning set. All results are averaged over 3 runs. We use MultEval (Clark et al., 2011) to perform a permutation test to estimate statistical significance.

4.3.2 Feature Sets

We experimented with a small (basic) feature set, and a large (sparse) feature set. For the small feature set, we use 14 features, including a language model, 5 translation model features, penalties for unknown words, the glue rule, and rule arity. For experiments with a larger feature set, we introduced additional lexical and non-lexical sparse Boolean features of the form commonly found in the literature and described in Section 3.4.3 (Chiang et al., 2009; Watanabe et al., 2007; Simianer et al., 2012).

These feature templates resulted in a total of 3.4 million possible features, of which only a fraction were active for the respective tuning set and optimizer, as shown in Table 4.2.

4.3.3 Results

As can be seen from the results in Table 4.3, our RM method was the best performer in all Chinese-English tests according to all measures – up to 1.9 BLEU and 6.6 TER over

[4] We also conducted an investigation into the setting of the B parameter. We explored alternative values for B, as well as scaling it by the current candidate's cost, and found that the optimizer is fairly insensitive to these changes, resulting in only minor differences in BLEU.

Optimizer	Zh	Ar
MIRA	35k	37k
PRO	95k	115k
RAMPION	22k	24k
RM	30k	32k
Active+Inactive	3.4M	4.9M

Table 4.2: Active sparse feature templates

Optimizer	Small (basic) feature set					Large (sparse) feature set				
	Tune	MT03		MT05		Tune	MT03		MT05	
	↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
MERT	35.4	35.8	60.8	32.4	63.9	-	-	-	-	-
MIRA	35.5	35.8	61.1	32.1	64.6	36.6	35.9	60.6	32.1	64.1
PRO	34.1	36.0	60.2	31.7	63.4	35.7	34.8	56.1	31.4	59.1
RAMPION	35.1	36.5	58.6	33.0	61.3	36.7	36.9	57.7	33.3	60.6
RM	31.3	**36.5**$^{R^-}$	**56.4**	**33.6**$^{R^a}$	**59.3**	33.2	**37.5**	**54.6**	**34.0**	**57.5**

Table 4.3: Performance on Zh-En with basic (left) and sparse (right) feature sets on MT03 and MT05. All RM improvements are significant at the p <0.001 level unless otherwise indicated: ($^-$) indicates significance at $p < 0.05$ and (^{a}) at $p < 0.1$. These are combined with RAMPION (^{R}) to indicate baseline for comparison.

MIRA – even though we only optimized for BLEU.[5] Surprisingly, it seems that MIRA did not benefit as much from the sparse features as RM. The results are especially notable for the basic feature setting – up to 1.2 BLEU and 4.6 TER improvement over MERT – since MERT has been shown to be competitive with small numbers of features compared to high-dimensional optimizers such as MIRA (Chiang et al., 2008b).

For the tuning set, the decoder performance was consistently the *lowest* with RM, compared to the other optimizers. We believe this is due to the RM bounding constraint being more resistant to overfitting the training data, and thus allowing for improved generalization. Conversely, while PRO had the second lowest tuning scores, it seemed to display signs of underfitting in the basic and large feature settings.

4.4 Additional Experiments

In order to explore the applicability of our approach to a wider range of languages, we also evaluated its performance on Arabic-English translation. All experimental details were the same as above, except those noted below.

For training, we used the non-UN portion of the NIST training corpora, which was segmented using an HMM segmenter (Lee et al., 2003). Dataset statistics are given in the bottom part of Table 4.1. The sparse feature templates resulted here in a total of 4.9 million possible features, of which again only a fraction were active, as shown in Table 4.2.

As can be seen in Table 4.4, in the smaller feature set, RM and MERT were the

[5] In the small feature set RAMPION yielded similar best BLEU scores, but worse TER. In preliminary experiments with a smaller trigram LM, our RM method consistently yielded the highest scores in all Chinese-English tests – up to 1.6 BLEU and 6.4 TER from MIRA, the second best performer.

Optimizer	Small (basic) feature set					Large (sparse) feature set				
	Tune	MT05		MT08		Tune	MT05		MT08	
	↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
MERT	43.8	**53.3**	**40.2**	**41.0**	50.7	-	-	-	-	-
MIRA	43.0	52.8	40.8	**41.3**	50.6	44.4	53.4	40.1	**41.8**	50.2
PRO	41.5	51.3	41.5	39.4	51.5	46.8	53.2	40.0	41.4	49.7
RAMPION	42.4	52.0	40.8	40.0	50.8	44.6	52.9	40.4	41.0	50.4
RM	38.5	**53.3**$^{E^n}_{M^-}$	**39.8**$^{E^n}$	40.6	**49.7**	43.0	**55.3**	**37.5**	**41.8**$^{P^-}_{M^-}$	**48.4**

Table 4.4: Performance on Ar-En with basic (left) and sparse (right) feature sets on MT05 and MT08. All RM improvements are significant at the p <0.001 level unless otherwise indicated: ($^-$) indicates significance at $p < 0.05$ and (^{n}) indicates insignificant improvement at $p > 0.1$. These are combined with MIRA (^{M}), PRO (^{P}) and MERT (^{E}) to indicate baseline for comparison.

best performers, with the exception that on MT08, MIRA yielded somewhat better (+0.7) BLEU but a somewhat worse (-0.9) TER score than RM.

On the large feature set, RM is again the best performer, except, perhaps, a tied BLEU score with MIRA on MT08, but with a clear 1.8 TER gain. In both Arabic-English feature sets, MIRA seems to take the second place, while RAMPION lags behind, unlike in Chinese-English (§4.3).[6]

Interestingly, RM achieved substantially higher BLEU precision scores in all tests for both language pairs. However, this was also usually coupled had a higher brevity penalty (BP) than MIRA, with the BP increasing slightly when moving to the sparse setting.

[6]In our preliminary experiments with the smaller trigram LM, MERT did better on MT05 in the smaller feature set, and MIRA had a small advantage in two cases. RAMPION performed similarly to RM on the smaller feature set. RM's loss was only up to 0.8 BLEU (0.7 TER) from MERT or MIRA, while its gains were up to 1.7 BLEU and 2.1 TER over MIRA.

Optimizer	Small set BLEU	TER	Large set BLEU	TER
MERT	0.4	2.6	-	-
MIRA	0.5	3.0	1.4	4.3
PRO	1.4	2.9	2.0	1.7
RAMPION	0.6	1.6	1.2	2.8

Table 4.5: RM gain over other optimizers averaged over all test sets.

4.5 Discussion

The trend of the results, summarized as RM gain over other optimizers averaged over all test sets, is presented in Table 4.5. RM shows clear advantage in both basic and sparse feature sets, over all other state-of-the-art optimizers. The RM gains are notably higher in the large feature set, which we take as an indication for the importance of bounding the spread.

Figures 4.3 and 4.4 present a comparison of the performance on the Chinese tuning set between MIRA and RM. The time per iteration is slightly higher with RM, but both finish 20 iterations in under an hour using 15 learners. RM's scores show the effect that the bounding constraint has on optimization, introducing larger deviations between iterations early on, but eventually converging to a more stable performance.

4.5.1 Spread Analysis

For RM, the average spread of the projected data in the Chinese-English small feature set was 0.9 ± 3.6 for all tuning iterations, and 0.7 ± 2.9 for the iteration with the highest decoder performance. In comparison, the spread of the data for MIRA was 5.9 ± 20.5 for

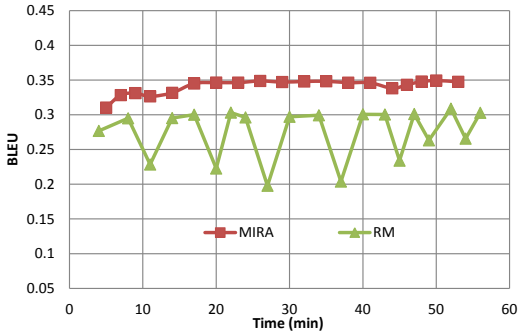


Figure 4.3: Comparison of performance vs. time on the tuning set for MIRA and RM. The average time per iteration for MIRA is 2.7±0.7 minutes, and for RM is 3.0±0.9 minutes.

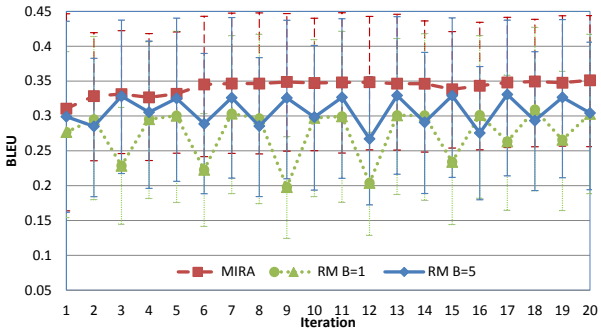


Figure 4.4: Comparison of learning performance with hope and fear scores for MIRA and RM on the tuning set.

the best iteration. In the sparse setting, RM had an average spread of 0.9±2.4 for the best iteration, while MIRA had a spread of 14.0±31.1. Similarly, on Arabic-English, RM had a spread of 0.7±2.4 in the small setting, and 0.82±1.4 in the sparse setting, while MIRA's spread was 9.4±26.8 and 11.4±22.1, for the small and sparse settings, respectively. Notice that the average spread for RM stays about the same when moving to higher dimensions, with the variance decreasing in both cases. For MIRA, however, the average spread increases in both cases, with the variance being much higher than RM. For instance, observe that the spread of MIRA on Chinese grows from 5.9 to 14.0 in the sparse feature setting. While bounding the spread is useful in the low-dimensional setting (0.7-1.5 BLEU gain with RM over MIRA as shown in Table 4.3), accounting for the spread is even more crucial with sparse features, where MIRA gains only up to 0.1 BLEU, while RM gains 1 BLEU. These results support the claim that our imposed bound B indeed helps decrease the spread, and that, in turn, lower spread yields better generalization

performance.[7]

4.5.2 Error Analysis

The inconclusive advantage of RM over MIRA (in BLEU vs. TER scores) on Arabic-English MT08 calls for a closer look. Therefore we conducted a coarse error analysis on 15 randomly selected sentences from MERT, RMM and MIRA, with basic and sparse feature settings for the latter two. This sample yielded 450 data points for analysis: output of the 5 conditions on 15 sentences scored in 6 violation categories. The categories were: function word drop, content word drop, syntactic error (with a reasonable meaning), semantic error (regardless of syntax), word order issues, and function word mistranslation and "hallucination". The purpose of this analysis was to get a *qualitative* feel for the output of each model, and a better idea as to why we obtained performance improvements. RM noticeably had more word order and excess/wrong function word issues in the basic feature setting than any optimizer. However, RM seemed to benefit the most from the sparse features, as its bad word order rate dropped close to MIRA, and its excess/wrong function word rate dropped below that of MIRA with sparse features (MIRA's rate actually doubled from its basic feature set). We conjecture both these issues will be ameliorated with syntactic features such as those in Chiang et al. (2008b).

This correlates with our observation that RM's overall BLEU score is negatively impacted by the BP, as the BLEU precision scores are noticeably higher. Table 4.6 shows BLEU n-grams precisions and BP for MIRA and RM on the Chinese MT03 test set. RM

[7] Although we focused on incorporating a bounding constraint into a ramp-like loss, it as been suggested that the bound may behave like a form of regularization, and thus we believe can be applied to other losses, such as the log-loss.

Optimizer	features	1-gram	2-gram	3-gram	4-gram	BP
MIRA	base	76.7	45.9	27.6	16.8	1.0
RM	base	79.5	47.4	28.6	17.4	0.99
MIRA	sparse	77.1	46.0	27.9	17.0	1.0
RM	sparse	80.3	48.5	29.7	18.4	0.98

Table 4.6: Comparison of RM and MIRA BLEU precisions and penalties on MT03.

with base features has better precisions than MIRA with sparse features, while the BP penalty increases with sparse features for RM.

4.5.3 Parameter Settings

RM is potentially more sensitive to the size and order of the k-best list. While MIRA is only concerned with the margin between $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$, RM also accounts for the distance between $\boldsymbol{y}^+$ and $\boldsymbol{y}^w$. Thus, it might be the case that a larger k-best, or revisiting previous strategies for $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$ selection discussed in Section 3.2.1 might have a greater impact. There are also many settings for B, where recall that as $B \to \infty$ we return to the original large-margin solution. This introduces a continuum of RM solutions that trade off between margin and spread in different ways. Table 4.7 presents a comparison between performance using our original 500-best list, and increasing it by an order of magnitude to 5000. As can be seen, the tuning BLEU decreases significantly, and test performance degrades by 1 point, while TER stays approximately the same. When increasing B from 1 to 5, we see a 2 point increase in the tuning BLEU, performing midway between the RM and MIRA tuning scores from Table 4.3. However, we only see a slight change in test performance.

bound	k-best	Tune	MT03		MT05	
		↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
1	500	31.3	36.5	54.7	33.5	57.0
5	500	33.3	36.7	54.7	33.6	57.5
1	5000	29.3	35.4	54.6	32.2	56.8

Table 4.7: Comparison of RM performance under different bound and k-best settings.

4.5.4 Active Features

Perhaps contrary to expectation, we did not see evidence of a correlation between the number of active features and optimizer performance. RAMPION, with the fewest features, is the closest performer to RM in Chinese, while MIRA, with a greater number, is the closest on Arabic. We also notice that while PRO had the lowest BLEU scores in Chinese, it was competitive in Arabic with the highest number of features.

4.5.5 Model Comparison

Aside from producing quantitatively different solutions in terms of performance, the parameters found by RM versus MIRA are also quite distinct. Table 4.8 presents a comparison of the best features for the baseline setting of Chinese-English. We can immediately notice that the RM solution heavily downweighs most parameter values, including the language model and translation models.

Feature	Weight
PassThrough	-9.910
SourceWordPenalty	-1.000
WordPenalty	-0.700
PhraseModel_0	-0.276
Arity_0	0.232
Arity_2	0.232
PhraseModel_1	-0.212
LanguageModel	0.173
PhraseModel_2	-0.167
Arity_1	0.124
PhraseModel_4	-0.095
PhraseModel_3	0.065
Glue	-0.026

Feature	Weight
WordPenalty	-10.007
PassThrough	-10.001
LanguageModel	2.206
PhraseModel_1	-2.023
PhraseModel_2	-1.732
PhraseModel_0	-1.257
Glue	-1.138
PhraseModel_4	1.032
PhraseModel_3	-1.010
SourceWordPenalty	-1.000
Arity_1	0.168
Arity_0	0.013
Arity_2	0.013

Table 4.8: Features and weights for RM and MIRA solutions arranged in decreasing order of importance.

4.6 Future Work

In achieving sparse feature improvements with RM we made use of standard, relatively small tuning sets, contrasted with improvements involving sparse features obtained using much larger tuning sets, on the order of hundreds of thousands of sentences (Liang et al., 2006; Tillmann and Zhang, 2006; Blunsom et al., 2008b; Simianer et al., 2012). Since our approach is complementary to scaling up the tuning data, combining these two methods should yield further improvements. In future work we also intend to explore using additional sparse features that are known to be useful in translation, e.g. syntactic features explored by Chiang et al. (2008b).

Finally, although motivated by statistical machine translation, RM is a gradient-based method that can easily be applied to other problems. We plan to investigate its utility elsewhere in NLP (e.g. for parsing) as well as in other domains involving high-

dimensional structured prediction.

4.7 Summary

In this chapter, we focused on how to find a large-margin solution that generalizes better by introducing a loss for structured relative margin with latent variables and cost-augmented hypothesis selection. We developed RM, a novel online margin-based algorithm designed for optimizing high-dimensional feature spaces, which introduces constraints into a large-margin optimizer that bound the spread of the projection of the data while maximizing the margin. The closed-form online update for our relative margin solution accounts for surrogate references and latent variables.

Empirical evaluation in statistical MT yielded significant improvements over several other state-of-the-art optimizers, especially in a high-dimensional feature space (up to 2 BLEU and 4.3 TER on average). Overall, RM achieves the best or comparable performance according to two scoring methods in two language pairs, with two test sets each, in small and large feature settings. Moreover, across conditions, RM always yielded the best combined TER-BLEU score.[8]

[8] We and other researchers often use $\frac{1}{2}(\text{TER} - \text{BLEU})$ as a combined SMT quality metric.

5 Adaptation with Topic Models

We see only what we know.

— Johann Wolfgang von Goethe

In the previous chapters, we have developed and extended our learning algorithm to effectively optimize large numbers of parameters. As discriminatively informative features allow our system to translate mode accurately, we now turn our attention to taking advantage of our ability to learn in high-dimensions by defining new feature functions. Thus, this chapter presents our contribution to the problem of how to represent the input by developing novel features for translation modeling.

In accordance with improving the generalization ability of our learner in Chapter 4, we develop features which help our model generalize to new **domains**. While domain adaptation has typically been done with manually defined domains and corpora, we employ topic models to perform *unsupervised domain induction*; this can be thought of as inducing subcorpora for adaptation without any human annotation. By introducing adaptation features based on probabilistic domain membership, we are able to *dynamically*

bias our translation model toward relevant translations based on topic-specific contexts.[1]

5.1 Introduction

As in all statistical learning, we make the assumption in SMT that what we will see in the future looks like what we saw in the past. Thus, the performance of a statistical machine translation system on a translation task depends largely on the quantity and suitability of the available parallel training data. Recall from Chapter 2 that our translation rules are extracted for a given source sentence based on patterns we have observed in our bitext data. If a sentence presents completely new lexical items or phrases, then we will not be able to translate those at all. However, even when the sentence contains words that we have observed, they may still be *polysemous* or *homonymous*, i.e. they are either words which have multiple meanings, or different words sharing the same orthographic form, respectively. The problem of word sense disambiguation (Jurafsky and Martin, 2008) is a difficult one monolingually, however, it is even further exacerbated in a multilingual setting. The same polysemous and homonymous words occurring in different contexts usually have **alternate meanings**, and will likely result in different translations.

For instance, a *tennis court* and *supreme court* refer to different types of courts in English, one for playing sports, the other for conducting judicial proceedings. While they share the same form in English, they will most often translate to two different words in another language; but how is our translation model supposed to make the distinction between senses?

It is commonplace in SMT to define the notation of a **domain**, and assume that data

[1] This chapter is based on material originally published in Eidelman et al. (2012).

coming from that domain will share certain characteristics, such as lexical preferences, stylistic tendencies, and so on. A domain may refer to a collection of text of a specific type, commonly referred to as a genre; common ones include newswire, broadcast news, and blogs. Domain may also refer to a more fine-grained distinction between the specific sources of text, such as different news outlets, or assimilation organizations that collect and process text; such as Xinhua news, LDC, or UN.

Domains may vary widely in their lexical choices and stylistic preferences, and what may be preferable in a general setting, or in one domain, is not necessarily preferable in another domain. Indeed, sometimes the domain can change the meaning of a phrase entirely. In a food related context, the Chinese sentence "粉丝很多 " ("fěnsī hěnduō") would mean "They have a lot of vermicelli"; however, in an informal Internet conversation, this sentence would mean "They have a lot of fans". Without the broader context, it is impossible to determine the correct translation in otherwise identical sentences.

Thus, a translation model that is built on newswire training data will be a poor choice when tasked with the translation of medical documents. This problem has led to a substantial amount of recent work in trying to adapt the translation model (TM) toward particular domains of interest (Axelrod et al., 2011; Foster et al., 2010).[2] Assuming our training corpus is composed of several different domains, we might reasonably assume that domain gives us the indication of translation meaning that we were looking for. In our earlier example, for instance, knowing *tennis court* is coming from ESPN, or *supreme court* is coming from the National Law Journal, would be a very strong indication to resolve the translation ambiguity.

[2]Language model adaptation is also prevalent but is not the focus of this work.

Domain adaptation is the task of biasing our existing model to account for changes in the domain. The intuition behind TM adaptation is to increase the likelihood of selecting relevant phrases for translation.

Notice that this problem setting makes the assumption that we know what the domain we are trying to adapt to is. This is a fairly common assumption, and results in the **cross-domain adaptation** setting. However, here we are interested in **dynamic adaptation**, where we do not know beforehand the domain of the unseen data. We will elaborate on this issue further in Section 5.2.

The common thread throughout prior work is the concept of a *domain*. Inevitably domain is a manually defined distinction of **provenance**: where the data is from. A domain is typically an externally imposed, hand labeled, hard constraint, such as genre or corpus collection. For example, a sentence either comes from newswire, or weblog, but not both. However, this poses several problems.

First, many domain adaptation approaches focus on dividing the training data by domain and learning different parameters for each one (Foster et al., 2010; Chiang et al., 2011). Since in this setting a sentence contributes its counts only to the translation table for the domain it came from, many word pairs will be unobserved for a given domains rule table. This sparsity necessitates some form of smoothing. Second, and more important, we may not actually know the (sub)corpora our training data comes from; and even if we do, *subcorpus* may not be the most useful notion of domain to help translation quality.

In this chapter, we take advantage of our learning algorithms that allow efficient high-dimensional optimization and take a finer-grained, flexible, unsupervised approach to the idea of lexical weighting by domain. We induce domains in an unsupervised fash-

ion from large corpora, and we incorporate soft, probabilistic domain membership into a translation model. Unsupervised modeling of the training data produces naturally occurring subcorpora, generalizing beyond corpus and genre. Depending on the model used to select subcorpora, we can dynamically bias our translation toward any arbitrary distinction. This reduces the problem to identifying what automatically defined subsets of the training corpus may be beneficial for translation.

We consider the underlying **latent topics** of the documents (Blei et al., 2003). In our case, by building a topic distribution for the source side of the training data, we abstract the notion of domain to include automatically derived subcorpora with probabilistic membership. This topic model is used to infer the topic distribution of a test set and bias sentence translations toward appropriate topics. We do this by introducing topic dependent lexical probabilities directly as features in the translation model, and interpolating them linearly with our other features, thus allowing us to discriminatively optimize their weights on an arbitrary objective function. Incorporating these features into our hierarchical phrase-based translation system significantly improved translation performance, by up to 1 BLEU and 3 TER over a strong Chinese to English baseline.

5.2 Domain Adaptation

The problem of domain adaptation is an active area of research in SMT (Daumé and Marcu, 2006; Koehn and Schroeder, 2007; Foster et al., 2010; Axelrod et al., 2011; Carpuat et al., 2013). From one perspective, the fact that training a system on one domain leads to such poor performance on another can be viewed as a problem of overfitting. We

learn parameters in such a way as to best fit our current (training) domain with all of its tendencies and quirks, and then are unable to subsequently produce correct translations for a different domain. The question then becomes how to tune our system in order to improve its performance either on a specific domain, or on different domains in general. The first problem is an instance of cross-domain adaptation, while the second is dynamic adaptation. Since, as we described in Chapter 2, translation systems are built from a number of heterogeneous steps, there is potential to adapt each component: the alignment, the internal parameters of the translation and language models, and the linear weighted combination.

Ideally, if we know the required domain ahead of time, and have some amount of data from it, we should train our system on the corresponding data. This scenario is known as cross-domain adaptation. However, this could still be problematic if the domain does not offer a sufficient amount of parallel data to estimate a translation model. It would be unwise to volitionally discard available data, even if it does not exactly match our domain. Instead, we should combine the data sources in such a way as to leverage the small domain-specific data to the fullest extent, while also taking into account the larger domain. In this way, we can have a translation model that is capable of general translation, but its parameters settings have be biased with domain-specific information.

Most domain adaptation work falls into this category. Specifically, they assume one domain as the *in*-domain, or target for adaptation, and everything else is the *out*-of-domain, or general-domain. It can be viewed as supervised adaptation, as the in domain provides guidance as to the best setting of parameters. As many successful techniques have been proposed for this setting, we will briefly review the relevant ones here.

These fall into two categories: those that modify the training data, for instance by weighting different domains differently, or filtering domains, and training a single translation model, and those that train separate translation models for different domains and combine them. While we could simply combine the in and out with uniform weight and train a single translation model, since the general domain corpus is usually much larger, this does not take full advantage of the in domain data (Koehn and Schroeder, 2007).

Better results can be achieved using the latter approach and training separate translation models (Koehn and Schroeder, 2007). This approach can be seen as an instance of mixture modeling (Hastie et al., 2001). Foster and Kuhn (2007) trained a separate translation model on each domain of interest, and then interpolated the translation tables with either log-linear or linear mixture weights. For the log-linear mixture, the translation models were each represented as a feature in the global scoring function, and thus combined directly with the other features. For the linear mixture, the mixture weights were learned on a separate held-out set based on distance from the current sentence to the different domains, and then the combined mixture probability was represented in the global score as one feature. Thus, the difference becomes whether the mixing weights are set in conjunction with all other features to optimize the overall loss, or separately for another given optimization criterion. Surprisingly, they found that even a linear mixture with uniform weights produces a gain.

Matsoukas et al. (2009) focus on estimating the quality of each individual sentence in the bitext in order to down-weight the importance of out-of-domain segments. They introduced assigning a pair of binary features to each training sentence, indicating sen-

tences' *genre and collection* as a way to capture domains. They then learn a mapping from these features to sentence weights, and use the sentence weights to bias the model probability estimates by computing weighted counts for a single translation model, and subsequently learn the model weights. While they do not have to indicate what is in or out, the working assumption is that the domain distribution on the tuning set will be similar to the test.

Foster et al. (2010) extend this approach and perform it on a more granular level: individual phrase pairs. Instead of focusing on domain indicator features, they build on work by Daumé and Marcu (2006) and Daumé (2007) to capture whether language is general or specific. Foster et al. (2010) use a larger set of features than Matsoukas et al. (2009) to capture whether a phrase pair is more similar to the in domain or general domain. Weights for out domain phrases are learned using a log-linear model and influence the relative frequency estimates of phrase pairs coming from the out domain, which are then combined linearly with in domain probabilities.

As Matsoukas et al. (2009) found sentence weights to be most beneficial for lexical weighting, Chiang et al. (2011) extends the same notion of conditioning on provenance (i.e. the origin of the text) by removing the separate mapping step of finding sentence weights, instead directly optimizing the weight of the genre and collection features in the linear model. They then compute a separate word translation table for each feature, estimated from only those sentences that comprise that genre or collection, which will be used to compute the lexical smoothing probabilities.

An alternative adaptation setting is that of dynamic adaption, where we have no information about a specific domain; we must simply select the best possible translation

for each sentence during decoding (Foster and Kuhn, 2007).[3] This amounts to performing adaptation in an unsupervised fashion. This scenario is less commonly examined.

Even if we do not have a marked *in* domain, but only a general corpus, we may assume some parts of it are more applicable to a given sentence. For instance, to perform dynamic adaptation, Foster and Kuhn (2007) constructed a multi-domain tuning set and set linear mixture parameters as a function of distance of the different domains to the current sentence. Finch and Sumita (2008) use a log-linear model to determine the probability a sentence comes from the question or declaration domain before decoding, and train translation models for each domain. They control the contribution of each model dynamically through interpolation weights which are set to be the probability assigned by the classifier.

5.3 Topic Models

Topic models are a type of unsupervised Bayesian generative model for analyzing and describing collections of text. They can be seen as a tool for dimensionality reduction whereby the space of all words in the collection is reduced to a smaller low-dimensional space of automatically fitted themes that occur in the data. These themes, or topics, as they are usually referred to, are sets of words selected from the data that share some similarity. Specifically, its posited that there is a **latent dimension of the data** which relates these words to each other.

The reason why biasing toward certain genres and corpus collections helps trans-

[3]These two are not necessarily mutually exclusive, as we could have a cross-domain setting with some in domain data, and then also make adjustments on a sentence basis.

lation is because these domains usually share certain characteristics and preferences for lexical choice that help resolve translation ambiguity. As the goal of topic models is to automatically find meaningful *similarities* between possibly disparate documents, our hypothesis is that perhaps this unsupervised approach, unbiased by distinctions of genre and collection, can find other similarities that can be beneficial, perhaps ever more so, for translation.

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) has emerged as one of the most popular methods for building topic models, and is available in multiple implementations. The main driving intuition behind LDA is that a single document is composed of a number of different topics. In describing LDA, as with most generative models, such as the noisy-channel model in Chapter 2, it is typical to refer to the generative story of how the model assumes the document was created.

LDA extends the multinomial mixture modeling approach described above by adding an additional generation step into hierarchy. Formally, each document in a collection $\mathcal{T}$ with vocabulary V is modeled as a multinomial over K unknown topics, where each topic is a multinomial distribution over the vocabulary. All the documents in $\mathcal{T}$ share the same set of K topics, but each document is allowed to have its own distribution over those topics.

Instead of assuming that each document is generated by choosing a class, and then choosing words form a class conditional density, the first step in generating a document is choosing a multinomial *distribution* over topics (i.e. classes). Then, each word in the document will be chosen from one of the topics. For each word, we first choose a topic z according the the distribution over topics we chose for this document, and then, from the

distribution over words that this topic defines, we choose a word.

The generative story of LDA proceeds as follows for each document:

1. Choose per-document topic distribution θ
2. Choose per-topic word distribution for each topic ϕ_k
3. For each word position w_n:
   (a) Choose topic z_n from topic distribution according to θ
   (b) Choose word w_n according to topic-word probability ϕ_{z_n}

The documents and words within are observed, but the topics, document-topic distributions and the topic-word distributions are latent, and need to be computed from the data. The best model is the one whose latent structure best explains the document. The explanation is a form of thematic annotation.

Figure 5.1 presents the topic distributions obtained by a three topic topic model over the corpus of three documents shown. The bars for each document represent the probability of that topic in the document. For instance, we would expect *tennis court* to occur in a document with sports related content, so the probability of the sports topic, Topic 1, is highest. Likewise, *supreme court* comes from Topic 2, the legal topic, and *food court*, comes from Topic 3, the food topic. By looking at such a graphical representation, we are able to easily summarize the main elements of each document.

There has been some exploration of topic modeling for use in SMT. For instance Bilingual LSA adaptation (Tam et al., 2007), which uses source side topic information on the target side to adapt the language model, and the BiTAM model (Zhao and Xing,

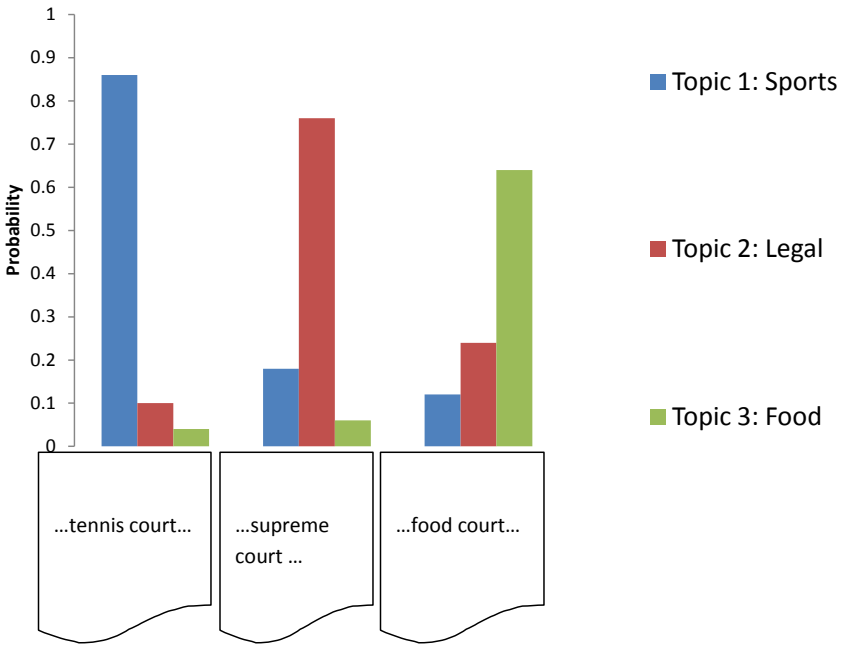


Figure 5.1: Example topic distribution over a collection of three documents with three possible topics.

2006), which uses a bilingual topic model for learning alignment. Recently, Xiao et al. (2012) proposed a topic similarity model which assigns each rule in the grammar a topic distribution, and compute a topic similarity feature between a rule and document based on their topic distributions. They also include a topic sensitivity model so as not to penalize generic rules which have low topic similarity. Furthermore, they independently model the source and target side of rules, and compare the target similarity during decoding by projecting the target distribution into the source space. Hasler et al. (2012) use source side topic assignment in combination with sparse word and phrase pair features. They use Hidden Topic Markov Models (HTMM) (Gruber et al., 2007) which model documents as a Markov chain and assign a single topic to the whole sentence, rather than a distribution of topics. Su et al. (2012) trained a HTMM on in-domain monolingual data and the source-side of the out-of-domain bitext, where they estimate the probability of a topic

given source from the former, and probability of a phrase pair from the latter, combining the two by mapping the in-domain topic distribution into the out domain.

5.4 Lexical Weighting

We briefly introduced lexical weighting features in Section 2.2.4.1, but as we will be adapting them, we now give a more thorough explanation. Lexical weighting features were introduced by Koehn et al. (2003) as a way estimate the quality of a phrase pair by combining the lexical translation probabilities of the words in the phrase. In a hierarchical system, these correspond to translation rules. In other words, instead of computing the frequency with which we see a whole phrase:

$$p(\overline{e}|\overline{f}) = \frac{\text{count}(\overline{e}, \overline{f})}{\sum_{\overline{e}} \text{count}(\overline{e}, \overline{f})} \qquad p(\overline{f}|\overline{e}) = \frac{\text{count}(\overline{e}, \overline{f})}{\sum_{\overline{f}} \text{count}(\overline{e}, \overline{f})} \tag{5.1}$$

we see how well the individual words in the phrase translate to each by relying on the word translation probabilities computed from the aligned parallel data. Given a source sentence $\boldsymbol{f} = f_1 ... f_n$ and target sentence $\boldsymbol{e} = e_1 ... e_m$, with alignment $\boldsymbol{a}$, we compute this as described in Chiang et al. (2011):

$$\text{count}(f_j, e_i) \leftarrow \text{count}(f_j, e_i) + \frac{1}{|a_i|} \text{ for } j \in a_i \tag{5.2}$$

$$\text{count}(\text{NULL}, e_i) \leftarrow \text{count}(\text{NULL}, e_i) + 1 \text{ if } |a_i| = 0 \tag{5.3}$$

Then, we compute the lexical conditional probabilities $w(e|f)$ with maximum like-

lihood estimates from relative frequencies:

$$w(e|f) = \frac{\text{count}(f,e)}{\sum_e \text{count}(f,e)} \qquad w(f|e) = \frac{\text{count}(f,e)}{\sum_f \text{count}(f,e)} \tag{5.4}$$

The lexical weight $p_{\text{lex}}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$ and $p_{\text{lex}}(\overline{\boldsymbol{f}}|\overline{\boldsymbol{e}})$ of a phrase pair is computed as described in Koehn et al. (2003):

$$p_{\text{lex}}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}}) = \prod_{i=1}^{|\overline{e}|} \begin{cases} \frac{1}{|a_i|} \sum_{j \in a_i} w(e_i|f_j) & \text{if} |a_i| > 0 \\ w(e_i|\text{NULL}) & \text{otherwise} \end{cases} \tag{5.5}$$

This can be referred to as smoothing the phrase relative frequencies we obtain in Eq. 5.1, as we are basically backing off to a probability which we can estimate more reliably.

Chiang et al. (2011) showed that is it beneficial to condition the lexical weighting features on provenance by assigning each sentence pair a set of features, $f_s(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$, for each domain s, which compute a new word translation table $w_s(e|f)$ estimated from only those sentences which belong to s:

$$\frac{\text{count}_s(f,e)}{\sum_e \text{count}_s(f,e)} \tag{5.6}$$

where $\text{count}_s(\cdot)$ is the number of occurrences of the word pair in s.

5.5 Topic Modeling for MT

To generalize the notion of conditioning lexical weighting on provenance, we extend provenance to cover a set of automatically generated topics z_n. Given a parallel training corpus $\mathcal{T}$ composed of documents d_i, we build a source side topic model over $\mathcal{T}$,

which provides a topic distribution $p(z_n|d_i)$ for $z_n = \{1, \ldots, K\}$ over each document, using LDA. Then, we assign $p(z_n|d_i)$ to be the topic distribution for every sentence $\boldsymbol{x}_j \in d_i$, thus enforcing topic sharing across sentence pairs in the same document instead of treating them as unrelated. Computing the topic distribution over a document and assigning it to the sentences serves to tie the sentences together in the document context.

In many cases, document delineations may not be readily available for the training corpus. Furthermore, a document may be too broad, covering too many disparate topics, to effectively bias the weights on a phrase level. For this case, we also propose a local LDA model, which treats each sentence in the training corpus as a separate document.

To obtain the lexical probability conditioned on topic distribution, we augment our rule extraction to accept a topic distribution for each sentence, and imbue the topic distribution of a sentence to each rule extracted from that sentence. Then, to compute word translation probabilities, we first compute the expected count $e_{z_n}(e, f)$ of a word pair under topic $z_n = \{1, \ldots, K\}$:

$$e_{z_n}(e, f) = \sum_{d_i \in \mathcal{T}} p(z_n|d_i) \sum_{\boldsymbol{x}_j \in d_i} \text{count}_j(e, f) \tag{5.7}$$

where $\text{count}_j(\cdot)$ denotes the number of occurrences of the word pair in sentence x_j, and then compute the topic-dependent word probabilities in both directions:

$$w_{z_n}(e|f) = \frac{e_{z_n}(e, f)}{\sum_e e_{z_n}(e, f)} \qquad w_{z_n}(f|e) = \frac{e_{z_n}(e, f)}{\sum_f e_{z_n}(e, f)} \tag{5.8}$$

$w_{z_n}(e|f)$ tells us how likely we are to translate f into e when the topic is z_n. This is

in effect introducing K new word translation tables, one for each $w_{z_n}(e|f)$ and $w_{z_n}(f|e)$, and $2\dot{K}$ new corresponding features $f_{z_n}(\overline{e}|\overline{f})$ and $f_{z_n}(\overline{f}|\overline{e})$, where we compute the lexical weighting from the topic-dependent word probabilities as described in Eq. 5.5.

Our features will compute the similarity between each rule, which now has a topic distribution based on the training corpus, and the topic distribution of the sentence we are currently translating. Thus, although the word translation probabilities are computed offline and are static, the actual feature values we compute will depend on the topic distribution of the document we are translating. For a test document V, we infer topic assignments on V, $p(z_n|V)$, using the previously built topic model, thus keeping the topics found from $\mathcal{T}$ fixed. The feature values conditioned on topic then become:

$$f_{z_n}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}}) = -\log\left\{p_{z_n}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}}) \cdot p(z_n|V)\right\} f_{z_n}(\overline{\boldsymbol{f}}|\overline{\boldsymbol{e}}) = -\log\left\{p_{z_n}(\overline{\boldsymbol{f}}|\overline{\boldsymbol{e}}) \cdot p(z_n|V)\right\}, \quad (5.9)$$

a combination of the topic dependent lexical weight and the topic distribution of the sentence from which we are extracting the phrase. To optimize the weights of these features we combine them in our linear model with the other features when computing the model score for each phrase pair:[4]

$$\underbrace{\sum_p \mathrm{w}_p h_p(\overline{\boldsymbol{e}}, \overline{\boldsymbol{f}})}_{\text{unadapted features}} + \underbrace{\sum_{z_n} \mathrm{w}_{z_n} f_{z_n}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})}_{\text{adapted features}} \quad (5.10)$$

Combining the topic conditioned lexical weights $p_{z_n}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$ computed from the training corpus with the topic distribution $p(z_n|V)$ of the test sentence being translated provides

[4]The unadapted lexical weight $p(\overline{e}|\overline{f})$ is included in the model features.

a probability on how relevant that translation is to the sentence based on the topic. This allows us to bias the translation toward the topic of the sentence. For example, if topic k is dominant in $\mathcal{T}$, $p_k(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$ may be quite large, but if $p(k|V)$ is very small, then we should steer away from this phrase pair and select a competing phrase pair which may have a lower probability in $\mathcal{T}$, but which is more relevant to the test sentence at hand.

Also note, that while Chiang (Chiang et al., 2011) has to *explicitly* smooth the resulting $p_s(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$, since many word pairs will be unseen for a given domain s, we are already performing an *implicit* form of smoothing (when computing the expected counts), since each document has a distribution over all topics, and therefore we have some probability of observing each word pair in every topic.

5.6 Feature Representation

After obtaining the topic conditional features, there are two possible ways we could present them to the model. Which one we select depends on how we want our system to interpret $f_z(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}})$.

The two questions they could answer are: 1) F_1: What is the probability under topic 1 of this sentence, topic 2? ... topic z?, or 2) F_2: What is the probability under the most probable topic of this sentence? Second most? etc.

A model using F_1 learns whether a *specific* topic is useful for translation, since feature f_1 would be

$$f_1 = p_{z=1}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}}) \cdot p(z = 1|V) \tag{5.11}$$

feature f_2 would be

$$f_2 = p_{z=2}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}}) \cdot p(z = 2|V) \tag{5.12}$$

and so on. Thus, f_1 is telling us about the translation probability under the topic that the topic model identified as topic 1, f_2 refers to the topic models topic 2, etc. Using F_1, we can restrict our topics to have a one-to-one mapping with genre/collection, which is trivial to do by having as many topics as genres/collections and setting $p(z_n|d_i)$ to 1 for every sentence in the collection and 0 to everything else. In this case, we see that our method fully recovers Chiang et al. (2011).

F_1 is appropriate for *cross-domain* adaptation when we have advance knowledge that the distribution of the tuning data will match the test data, as in Chiang et al. (2011), where they assume one genre as the in domain, eg. web, and then tune their system and test on that domain. If we know that our translation task data will be drawn from the same document-topic distribution, then this approach will also work well for unsupervised domain induction.

If however, as in the more general case, we do not know what our data will be, this approach will have the effect of overfitting the tuning set. For instance, if our tuning data consisted of the sports document in Figure 5.1, then f_1 would be the probability of translation under the sports topics, and we would learn that f_1 is informative. If our test data is all from the sports topic as well, then this will fine. If our test data is from the medical topic, however, we still have weight settings biasing us toward Topic 1, the sports topic, and we would not have learned that f_3 is important, and thus will not have adjusted our weights accordingly. Thus, we can see that F_1 is not the form of feature we would

like, since we do want to learn to bias our system toward a certain topic distribution.

On the other hand, with F_2, we are learning how useful knowledge of the topic distribution is in **general**, since f_1 is set to be the probability of translation under the most likely topic:

$$f_1 = p_{z=(\text{argmax}_{z_n}(p(z_n|V)))}(\overline{\boldsymbol{e}}|\overline{\boldsymbol{f}}) \cdot p(z = \underset{z_n}{\text{argmax}}(p(z_n|V))|V) \tag{5.13}$$

and so on. The key distinction is that F_2 does not does not correlate f_1 with the topic the topic model identified as topic 1. Rather, f_1 represents the translation probability under the dominant topic of the sentence, no matter what the identity of that topic is. F_2 is intuitively what we want, since we do not want to bias our system toward a specific distribution, but rather learn to utilize information from **any** topic distribution if it helps us create relevant translations. F_2 is useful for *dynamic* adaptation, where the adapted feature weight changes based on the source sentence.

In our running example, with this approach in the same scenario where we tune on the sports document, we would once again learn that f_1, representing the sports topic, is useful. The difference now lies in what happens when we apply it to unseen data. Upon receiving the medical document, f_1 would now represent the translation probability under topic 3, the medical topic, since that is the most prominent topic in the medical document; it would not be represented by f_3, as with F_1. Thus, by learning f_1 is helpful in the tuning data, when we apply our translation system to any new document, regardless of its topic distribution, we can leverage that information.

Thus, we do not condition the feature weights on the topic identity, but rather topic

prominence. If topic information is useful for translation, we should learn to favor hypotheses that have higher lexical probability from the most prominent topic of a document, regardless of what specific topic it is.

This adaptation is dynamic because the weight given to each topic as defined by the topic model changes from sentence to sentence, according to the topic distribution of the source sentence. The weight for topic z on sentence n could be different from the weight for topic z on sentence $n+1$, since the z that maximizes $(\text{argmax}_{z_n}(p(z_n|V)))$ can change, but the weight on whichever z does remains constant. Each feature can be thought of as a separate translation model, and the contribution of each of these translation model is determined dynamically through the weights. It is also worth noting that this approach should generalize to different topic distributions over the training data as well, not only over the test sentences given the model built on the training data, a point we examine in Section 5.7.4.

To summarize, if we know the data we want to translate after tuning will always consist of predominantly of some fixed topic distribution, we can achieve better translations for that specific topic by using features of the form F_1. If we do not, then F_2 is the appropriate choice. Thus, F_2 is the approach we utilize in our work, which allows us to tune our system weights toward having topic information be useful, not toward a specific distribution.

5.7 Experiments

5.7.1 Setup

To evaluate our approach, we performed experiments on Chinese to English MT in two settings. In the first setting we will assume the bitext has marked document boundaries in order to evaluate document level topic models versus modeling at the sentence level, while in the second, we will work in the more typical MT scenario of only having sentence boundaries.

For the first setup, we use the FBIS corpus as our training bitext for MT and training corpus for building the topic model. Since FBIS has document delineations, we can compare modeling at the document level (GTM) with local topic modeling (LTM). In the former, GTM, we compute the topic distribution of the document, and distribute that distribution to each sentence. In the latter, LTM, we disregard the document boundaries and infer a topic distribution on each sentence separately.

The second setting utilizes the non-UN and non-HK Hansards portions of the NIST training corpora for MT and building the LTM only. Table 5.1 summarizes the data statistics. For both settings, the data was lowercased, tokenized and aligned using GIZA++ (Och and Ney, 2003) to obtain bidirectional alignments, which were symmetrized using the `grow-diag-final-and` method (Koehn et al., 2003). The Chinese data were segmented using the Stanford segmenter. We trained a trigram LM on the English side of the corpus with an additional 150M words randomly selected from the non-NYT and non-LAT portions of the Gigaword v4 corpus using modified Kneser-Ney smoothing (Chen

Corpus	Sentences	Tokens	
		En	Zh
FBIS	269K	10.3M	7.9M
NIST	1.6M	44.4M	40.4M

Table 5.1: Corpus statistics.

and Goodman, 1996).

We use `cdec` as our decoder, and tune the parameters of the system to optimize BLEU on the NIST MT06 tuning corpus. Topic modeling was performed with Mallet (Mccallum, 2002), a standard implementation of LDA, using a Chinese stoplist and setting the hyperparameter $\alpha = 0.01$. This setting of α was chosen to encourage sparse topic assignments, which make induced subdomains consistent within a document.[5] All results are averaged over 3 runs. We use MultEval to perform a permutation test to estimate statistical significance.

5.7.2 Results

Results for the first setting, using the smaller document-delineated FBIS corpus, are shown in Table 5.2. GTM models the latent topics at the document level, while LTM models each sentence as a separate document. To evaluate the effect topic granularity would have on translation, we varied the number of latent topics in each model to be 5, 10, and 20.[6] On FBIS, we can see that both models achieve moderate but consistent gains over the MIRA baseline with no topic-conditioned features using both BLEU and TER.

[5] We explored a number of alternative hyperparameter settings, and found that having a peaked distribution for each sentence was important for translation.

[6] We explored the inclusion of more topics, but as performance plateaued after 20, we only include these settings in our results.

The best model, LTM-10, achieves a gain of about 0.5 and 0.6 BLEU and 2 TER. Although the performance on BLEU for both the 20 topic models LTM-20 and GTM-20 is suboptimal, the TER improvement is better. Interestingly, the difference in translation quality between capturing document coherence in GTM and modeling purely on the sentence level is not substantial.[7] In fact, the opposite is true, with the LTM models achieving better performance.[8]

Table 5.3 presents results on the NIST corpus. LTM-10 again achieves the best gain of approximately 1 BLEU and up to 3 TER. LTM performs on par with or better than GTM, and provides significant gains even in the NIST data setting, showing that this method can be effectively applied directly on the sentence level to large training corpora which have no document markings. Depending on the diversity of training corpus, a varying number of underlying topics may be appropriate. However, in both settings, 10 topics were optimal, suggesting that may be a good initial point. This is also the setting we will use for further experiments.

We further evaluated our model on our full BOLT system, comprised of 5 million training sentences taken from LDC, GALE, and BOLT releases and using a 5-gram language model trained on over 1 billion words. We built a 10 topic model on the entire BOLT training corpus, and then extracted topic-dependent lexical translation features as described above. The first column of results in Table 5.4 are the MIRA tuned baseline, and the second column are with LTM-10 features. While the gains are more modest than in the smaller NIST setting in Table 5.2, we still see general improvement across 5 different

[7] An avenue of future work would condition the sentence topic distribution on a document distribution over topics (Teh et al., 2006).

[8] As an empirical validation of our earlier intuition regarding feature representation, presenting the features in the form of F_1 caused the performance to remain virtually unchanged from the baseline model.

Model	MT03		MT05	
	↑BLEU	↓TER	↑BLEU	↓TER
BL	28.72	65.96	27.71	67.58
GTM-5	28.95^n	65.45	27.98^n	67.38^s
GTM-10	29.22	64.47	**28.19**	66.15
GTM-20	29.19	**63.41**	28.00^n	**64.89**
LTM-5	29.23	64.57	**28.19**	66.30
LTM-10	**29.29**	63.98	**28.18**	65.56
LTM-20	29.09^n	63.57	27.90^n	65.17

Table 5.2: Performance using FBIS training corpus. Improvements are significant at the p <0.05 level, except where indicated (^{n}).

test sets.

5.7.3 RM Optimization

After establishing that topic adaptation features improve translation, we evaluate whether combining these features with our RM optimizer can yield further improvements. Table 5.5 compares MIRA and RM using the 10 topic LTM model on the NIST setting. MIRA baseline results, and the best performing model with topics, LTM, are presented first, with RM with a bound B setting of 1, 5, and 10 on the bottom. As we observed in Chapter 4, minimizing the spread concurrently with maximizing the margin has the effect of lowering the tuning score on MT06 dramatically, while preserving the gains on the two test sets. As we decrease the aggressivity of the bound constraint B from 1 to 5 to 10, we see the RM solution coming closer to the LTM model. RM-10 produces better BLEU and TER scores, while RM-5 improves TER substantially while underperforming

Model	MT03		MT05	
	↑BLEU	↓TER	↑BLEU	↓TER
BL	34.31	61.14	30.63	65.10
MERT	34.60	60.66	30.53	64.56
LTM-5	35.21	59.48	31.47	62.34
LTM-10	**35.32**	**59.16**	**31.56**	**62.01**
LTM-20	33.90^{n}	60.89^{n}	30.12^{n}	63.87

Table 5.3: Performance using NIST corpus. Improvements are significant at the p <0.05 level, except where indicated (^{n}).

Test Set	Baseline		LTM-10	
	↑BLEU	↓TER	↑BLEU	↓TER
Test1	20.06	59.72	20.34	59.72
Test2	18.66	59.67	19.00	59.74
Test3	28.37	55.66	28.78	55.63
Test4	15.68	62.53	15.85	62.39
Test5	14.70	59.06	14.77	58.82

Table 5.4: Performance on BOLT test sets.

on BLEU. RM-1 is too aggressive, and is not able to perform on par with the others, giving an example where the B setting is too limiting.

5.7.4 Topic Distributions

To empirically verify our earlier point regarding the parameters from one distribution generalizing to different topic distributions over the training data, we compare using a learned model from one topic distribution directly on another. First, we used our previously built topic model and corresponding inferred topics on the tuning and test sets to

Model	Tune	MT03		MT05	
	↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
MIRA	34.31	35.12	59.99	31.56	63.05
LTM	34.03	35.83	58.67	32.30	61.40
RM-1	25.93	30.16	56.48	27.49	58.85
RM-5	28.44	35.23	55.60	32.50	58.29
RM-10	30.90	36.10	57.71	32.51	60.56

Table 5.5: Performance using RM learning with B set to 1, 5 and 10 compared to best performing MIRA learner with 10 topics, LTM. All RM results are with the 10 topic LTM model.

Model	MT03		MT05	
	↑BLEU	↓TER	↑BLEU	↓TER
Old	35.83	58.67	32.30	61.40
New	35.88	58.50	32.61	60.89

Table 5.6: Performance using weights trained on one 10 topic distribution to decode test sets with new 10 topic distribution.

tune parameters; presented as Old in Table 5.6. Then for New, we built a second topic model on the training data, and infer topics with it on the tuning and test sets as before. Table 5.8 shows the same 5 randomly selected sentences with topic distributions according to the Old and New models. It is clear that the sentences have different topic distributions. However, we do not retune the parameters using the new topic features. Instead, we decode using the parameters learned from Old. As the results show, the parameters learned from the old topic distribution are equally valuable on the new distribution, without having to retune the model weights, even though the topic distributions are obviously different.

Table 5.7 presents a comparison of the baseline, unadapted MIRA system from

Model	Tune	MT03		MT05	
	↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
BL	34.31	35.12	59.99	31.56	63.05
Shuffle Topic	33.63	34.41	61.07	31.19	64.00
Random	33.97	35.12	59.56	31.50	62.54

Table 5.7: Performance using random topic assignments. Shuffle uses same topic distribution assigned to each sentence with topic model, but randomly reassigns the topic index. Random generates random topic distributions for each sentence.

Table 5.5 with topic adaptation using random topic assignments. In the first case, Shuffle Topic, we retain the same topic distributions used in the LTM model for the training corpus and test sets, but randomly reassign the topic indices on each sentence. In the second case, Random, we generate random topic distributions for each sentence. In both cases, we can see that random topic assignment does not improve upon the baseline of having no topic adaptation, thereby showing that the improvement we observe is not simply an artifact of having additional attributes, but rather that the method of inducing topic assignments matters.

5.8 Discussion

Looking at the topic distributions inferred on our training data, we found topics which corresponding more to the notion of topic prevalent in topic modeling. Naturally, since certain genre talk about certain things more often, topics may in general correlate somewhat with genre/domain, however the interaction seems to be minimal here, as most of the training data is of similar genre. It is also worth noting that we chose Chinese due to its topic-comment structure, which may play a role in allowing topic modeling to

6	0.9987	9	1.43E-04	8	1.4E-04	7	1.4E-04	5	1.4E-04	4	1.4E-04	3	1.4E-04	2	1.4E-04	1	1.4E-04	0	1.4E-04
7	0.6218	8	0.3775	9	6.6E-05	6	6.6E-05	5	6.6E-05	4	6.6E-05	3	6.6E-05	2	6.6E-05	1	6.6E-05	0	6.6E-05
7	0.9743	8	0.0247	9	1.1E-04	6	1.1E-04	5	1.1E-04	4	1.1E-04	3	1.1E-04	2	1.1E-04	1	1.1E-04	0	1.1E-04
8	0.7494	2	0.1203	4	0.0648	5	0.0370	3	0.0185	9	0.0093	7	8.3E-05	6	8.3E-05	1	8.3E-05	0	8.3E-05
7	0.9991	9	9.9E-05	8	9.9E-05	6	9.9E-05	5	9.9E-05	4	9.9E-05	3	9.9E-05	2	9.9E-05	1	9.9E-05	0	9.9E-05

8	0.9987	9	1.43E-04	7	1.4E-04	6	1.4E-04	5	1.4E-04	4	1.4E-04	3	1.4E-04	2	1.4E-04	1	1.4E-04	0	1.4E-04
5	0.9994	9	6.66E-05	8	6.6E-05	7	6.6E-05	6	6.6E-05	4	6.6E-05	3	6.6E-05	2	6.6E-05	1	6.6E-05	0	6.6E-05
8	0.9743	2	0.0124	3	0.01244	9	1.1E-04	7	1.1E-04	6	1.1E-04	5	1.1E-04	4	1.1E-04	1	1.1E-04	0	1.1E-04
1	0.6384	0	0.3331	4	0.02783	9	8.3E-05	8	8.3E-05	7	8.3E-05	6	8.3E-05	5	8.3E-05	3	8.3E-05	2	8.3E-05
0	0.5551	7	0.2221	8	0.2221	9	9.9E-05	6	9.9E-05	5	9.9E-05	4	9.9E-05	3	9.9E-05	2	9.9E-05	1	9.9E-05

Table 5.8: The same 5 sentences with topic distributions from the old (top) and new (bottom) topic models. The topic assignments themselves are different in both, as are the distributions. Furthermore, the dominant topics across sentences do not seem to correlate from old to new.

influence translation performance. It would be interesting to examine whether this holds for other languages.

Inducing domains with topic models presents several advantages over existing approaches. We can construct a topic model once on the training data, and use it infer topics on any test set to adapt the lexical weighting features. We can also incorporate additional parallel, or monolingual data in the source language, to infer better latent topic distributions without relying on collection or genre distinctions.

5.9 Future Work

There are several possible avenues of exploration in future work. First, we could expand the topic modeling to multilingual topic models, with the hope of capturing topics of higher relevance to both the source and target language (Boyd-Graber and Resnik, 2010), and thus better suited for translation. We could then use the same technique presented above to integrate topic information onto the source side grammar, and simply take advantage of better topic assignments, or we could additionally include target side topic information. If bilingual topic modeling is beneficial, there are several interesting questions, such as how the data is aligned - does the data have to be comparable, parallel, etc.? Second, we could use big data for topic model, either monolingually or in combination with multilingual modeling, and assess whether we can achieve better translation with larger topic models. Furthermore, as sentences contain a mix of topics, we could also integrate topic modeling with instance weighting (Foster et al., 2010) to distribute topic information at a more granular level.

5.10 Summary

Whenever we apply SMT to different domains, it is beneficial to introduce domain knowledge. In this chapter, we abstracted the usual notion of a domain to finer-grained topic distributions induced in an unsupervised fashion and introduced a novel method for dynamic translation model adaptation. Utilizing our ability to optimize many features with the learning algorithms presented in Chapters 2 and 4, we showed that incorporating lexical weighting features conditioned on soft domain membership directly into our model is an effective strategy for dynamically biasing SMT towards relevant translations, as evidenced by significant performance gains in small to large settings. By applying the RM algorithm we were able to tune these features toward a relative margin, showing further gains on top of the large-margin model.

Feature	Weight
WordPenalty	-10.787
PassThrough	-9.997
LanguageModel	2.873
count($\boldsymbol{f}, \boldsymbol{e}$)	-2.014
count($\boldsymbol{f}$)	-1.654
count($\boldsymbol{f}, \boldsymbol{e}$)=1	-1.345
$p_{z=0}(\boldsymbol{f}\|\boldsymbol{e})$	-1.229
$p(\boldsymbol{e}\|\boldsymbol{f})$	-1.095
SourceWordPenalty	-1.000
count($\boldsymbol{f}$)=1	-0.964
$p_{z=0}(\boldsymbol{e}\|\boldsymbol{f})$	-0.859
$p_{lex}(\boldsymbol{e}\|\boldsymbol{f})$	0.848
Glue	-0.741
$p_{z=6}(\boldsymbol{e}\|\boldsymbol{f})$	-0.678
$p_{lex}(\boldsymbol{f}\|\boldsymbol{e})$	-0.664
Arity_0	0.583
Arity_2	0.583
$p_{z=6}(\boldsymbol{f}\|\boldsymbol{e})$	-0.497
Arity_1	-0.457
$p_{z=7}(\boldsymbol{e}\|\boldsymbol{f})$	-0.401
$p_{z=7}(\boldsymbol{f}\|\boldsymbol{e})$	-0.398
$p_{z=3}(\boldsymbol{f}\|\boldsymbol{e})$	-0.240
$p_{z=8}(\boldsymbol{e}\|\boldsymbol{f})$	0.226
$p_{z=4}(\boldsymbol{e}\|\boldsymbol{f})$	-0.197
$p_{z=1}(\boldsymbol{e}\|\boldsymbol{f})$	-0.175
$p_{z=9}(\boldsymbol{e}\|\boldsymbol{f})$	0.161
$p_{z=3}(\boldsymbol{e}\|\boldsymbol{f})$	-0.156
$p_{z=5}(\boldsymbol{e}\|\boldsymbol{f})$	0.150
$p_{z=2}(\boldsymbol{f}\|\boldsymbol{e})$	-0.136
$p_{z=1}(\boldsymbol{f}\|\boldsymbol{e})$	-0.109
$p_{z=9}(\boldsymbol{f}\|\boldsymbol{e})$	0.103
$p_{z=2}(\boldsymbol{e}\|\boldsymbol{f})$	0.071
$p_{z=5}(\boldsymbol{f}\|\boldsymbol{e})$	-0.064
$p_{z=4}(\boldsymbol{f}\|\boldsymbol{e})$	-0.009
$p_{z=8}(\boldsymbol{f}\|\boldsymbol{e})$	-0.008

Feature	Weight
PassThrough	-9.897
WordPenalty	-3.206
count($\boldsymbol{f}$)	-1.269
SourceWordPenalty	-1.000
$p_{lex}(\boldsymbol{e}\|\boldsymbol{f})$	0.695
LanguageModel	0.634
count($\boldsymbol{f}$)=1	-0.599
$p(\boldsymbol{e}\|\boldsymbol{f})$	-0.547
$p_{z=0}(\boldsymbol{e}\|\boldsymbol{f})$	-0.495
Arity_0	0.478
Arity_2	0.478
Glue	-0.445
count($\boldsymbol{f}, \boldsymbol{e}$)=1	-0.443
$p_{z=0}(\boldsymbol{f}\|\boldsymbol{e})$	-0.410
count($\boldsymbol{f}, \boldsymbol{e}$)	-0.347
$p_{z=6}(\boldsymbol{e}\|\boldsymbol{f})$	-0.306
$p_{z=7}(\boldsymbol{e}\|\boldsymbol{f})$	-0.302
Arity_1	-0.289
$p_{z=1}(\boldsymbol{e}\|\boldsymbol{f})$	-0.140
$p_{z=9}(\boldsymbol{e}\|\boldsymbol{f})$	0.129
$p_{z=3}(\boldsymbol{e}\|\boldsymbol{f})$	-0.120
$p_{z=9}(\boldsymbol{f}\|\boldsymbol{e})$	-0.098
$p_{z=5}(\boldsymbol{f}\|\boldsymbol{e})$	-0.089
$p_{z=6}(\boldsymbol{f}\|\boldsymbol{e})$	-0.088
$p_{z=3}(\boldsymbol{f}\|\boldsymbol{e})$	-0.087
$p_{lex}(\boldsymbol{f}\|\boldsymbol{e})$	-0.078
$p_{z=7}(\boldsymbol{f}\|\boldsymbol{e})$	-0.069
$p_{z=2}(\boldsymbol{f}\|\boldsymbol{e})$	-0.067
$p_{z=8}(\boldsymbol{e}\|\boldsymbol{f})$	-0.066
$p_{z=4}(\boldsymbol{e}\|\boldsymbol{f})$	-0.033
$p_{z=8}(\boldsymbol{f}\|\boldsymbol{e})$	0.032
$p_{z=4}(\boldsymbol{f}\|\boldsymbol{e})$	-0.027
$p_{z=1}(\boldsymbol{f}\|\boldsymbol{e})$	0.014
$p_{z=5}(\boldsymbol{e}\|\boldsymbol{f})$	0.005
$p_{z=2}(\boldsymbol{e}\|\boldsymbol{f})$	-0.005

Table 5.9: Features and weights for LTM-10 (left) and RM-10 (right) arranged in decreasing order of importance.

6 Latent Large-Margin Learning

> *You have your way. I have my way. As for the right way, the correct way, and the only way, it does not exist.*
>
> — Friedrich Nietzsche

> *When it is not in our power to determine what is true, we ought to follow what is most probable.*
>
> — Rene Descartes

The previous chapters discussed the problem of determining the appropriate objective to optimize (§3), and how to efficiently and effectively optimize the chosen objective (§4). However, consistent with previous literature, we have been intentionally loose in our interchangeable use of derivations and translations. In this chapter we present our final contribution to optimization, by introducing a general framework for latent variable models which explicitly accounts for derivations in both learning and decoding. We define a unified representation of a *family of latent structured losses* and show that various previously defined losses emerge as special cases. We then present a *novel loss* function from this family for large-margin learning in the latent variable setting, and develop a suitable optimization algorithm for *maximum probability translation learning* and inference

in SMT.

6.1 Introduction

As discussed in Chapter 2, discriminative training algorithms for SMT must contend with several issues that are not routinely encountered in standard structure prediction settings in machine learning. First, the assumption that the correct output is reachable is violated, since our decoder is often incapable of producing the reference translation. This issue has received considerable attention recently (Liang et al., 2006; Chiang, 2012), with the introduction of various forms of surrogate references obtained through cost-augmented hypothesis selection becoming prevalent.

The second issue, that a latent structure, or derivation, is constructed during the translation process, will be tackled in this chapter. As briefly discussed in the introduction, we can obtain many possible translations $\boldsymbol{y}$ from a source sentence $\boldsymbol{x}_i$, $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)$, and for each $\boldsymbol{y}$, we may have many derivations $\boldsymbol{d}$ that produce the same $\boldsymbol{y}$, $\boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}, \boldsymbol{y})$. Recall that derivations $\boldsymbol{d}$ comprise the set of rules used in the production of $\boldsymbol{y}$, but since we only observe translations $\boldsymbol{y}_i$, which may have many possible derivations $\boldsymbol{d}_j$, we model the derivations as a latent, or hidden, variable.

This leads to a well-known issue of **derivational** ambiguity, where there may be multiple, exponentially many, in fact, ways to produce the same target string. This problem is related to **spurious** ambiguity, which is sometimes defined narrowly as multiple *distinct* derivations (trees or segmentations) with the *same* set of features and output string. Whereas, derivational ambiguity broadens the criteria to encompass derivations

that can have *different* feature vectors but also produce the same target string. Derivational and spurious ambiguity have sometimes been used interchangeably in the literature, and here we focus on the broader problem of derivational ambiguity.

The problem is illustrated in Figure 6.1. The figure presents a simple grammar and hypergraph that, using only the 5 rules in the grammar, encodes 4 different derivations that translate *ein kleines haus* into *a little house*. Unfortunately, there is an exponential relationship between the sentence length and the number of derivations (Blunsom et al., 2008b), thus for a real sentence, we will have to deal with many more derivations.

$$
\begin{array}{lccc}
\mathrm{X} \longrightarrow & \langle\ \text{ein} & , & \text{a}\ \rangle \\
\mathrm{X} \longrightarrow & \langle\ \text{kleines} & , & \text{little}\rangle \\
\mathrm{X} \longrightarrow & \langle\text{ein kleines} & , & \text{a little}\rangle \\
\mathrm{X} \longrightarrow & \langle\text{kleines haus} & , & \text{little house}\rangle \\
\mathrm{X} \longrightarrow & \langle\text{haus} & , & \text{house}\ \rangle \\
\mathrm{S} \longrightarrow & \langle\ \mathrm{S}_0\ \mathrm{X}_1 & , & \mathrm{S}_0\ \mathrm{X}_1\rangle
\end{array}
$$

While the goodness of a translation should be the total probability of all its derivations, most past work in SMT optimization does not consider all possible derivations of a sentence. With some notable exceptions discussed below, it is standard practice in SMT to decode and optimize toward the 1-best, or maximum Viterbi derivation (Koehn et al., 2003), disregarding the multitude of other possibilities. Taking this shortcut is in effect treating derivations as the translation, and optimizing towards the best derivation, *not* the best translation.

Decoding and optimizing derivations is especially ubiquitous in linear and large-margin models. However, it seems that for translation, where the ambiguities of not one, but two languages are exposed, and where we are already limiting ourselves to building a

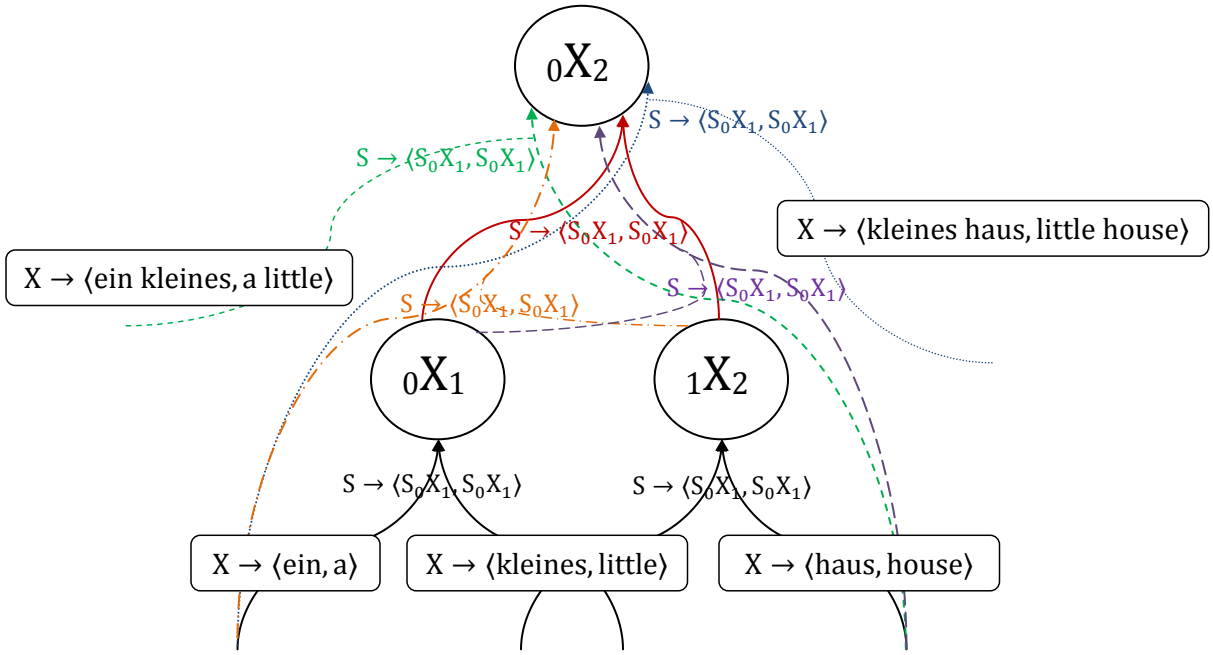


Figure 6.1: Hypergraph representing different derivations that translate *ein kleines haus* into *a little house*.

statistical system based on a cost computed on one to a handful of reference translations, limiting ourselves even further by optimizing toward a *single derivation* of that *single reference* is unwise.

Intuitively, we would like to account for all the possible derivations of all possible translations. Since features are constructed over derivations, many derivations can actually contribute to a particular translation string $\boldsymbol{y}$'s score. Accounting for all derivations should further allow our learned model parameter vector $\mathbf{w}$ to generalize better, since we are more likely to have observed a particular set of features.

We can separate this out into two separate but related problems: the decoding problem, where we decide what derivation and translation pair to produce, and the training problem, where we decide what derivation(s) to optimize for.

In the decoding problem introduced in Chapter 2, finding the maximum probability

translation necessitates marginalizing over all the derivations of all translations, and is thus computationally intractable. Sima'an (1996) showed that it is an NP-hard problem. For this reason, most systems address the decoding problem by approximating the maximum translation with the maximum derivation. However, ideally we would like to decode to find the maximum translation itself.

In the training problem, in the simplest case, large-margin learners choose two derivation and translation pairs, one for the surrogate reference, the hope, and another for the worst violator, the fear, where each pair is described by one feature vector. However, we would like to simultaneously tune our system toward *all* the derivations of a translation, replacing the feature vector with **feature expectations** by marginalizing over all the possible derivations. For this, we will take advantage of the fact that marginalization over derivations for a particular $\boldsymbol{y}$ *is* tractable.

In this chapter we accomplish this by introducing a new way of combining a probabilistic framework for dealing with latent variables with a geometric model that performs margin maximization. To date, margin and perceptron style methods have been used to optimize derivations. To the best of our knowledge, a latent variable model has not been used in coordination with a large-margin learner in SMT.

We define a unified representation of a family of latent structured losses (§6.3.3) and show that various previously defined losses emerge as special cases. We then present a novel loss function from this family for large-margin learning with latent variables (§6.3.3), and develop a suitable optimization algorithm for maximum probability translation learning and inference in SMT (§6.3.4). We *explicitly* account for latent variables in the translation process, instead of simply noting their presence but not taking advantage

of them. By marginalizing over derivation paths in the training and decoding process, we are able to combine maximum probability translation decoding with a latent variable large margin learner.

6.2 Decoding Problem

In the decoding problem, we need to decide what derivation and translation pair to produce. Since the maximum weighted derivation

$$(\boldsymbol{y}^*, \boldsymbol{d}^*) = \underset{(\boldsymbol{y},\boldsymbol{d})\in\mathcal{Y}(\boldsymbol{x}),\mathcal{D}(\boldsymbol{x},\boldsymbol{y})}{\operatorname{argmax}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d}) \tag{6.1}$$

can be efficiently extracted using a dynamic programming technique for hypergraphs (Dyer, 2010a), this is the most common approximation for the model's best output (Arun et al., 2009; Li et al., 2009). However, since the maximum derivation is only informed by one out of an exponential number of translation paths, it may be a bad approximation of the model's true maximum. The question for us then becomes, given a source sentence, how to find the maximum probability translation in Eq. 6.2 efficiently?

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})}{\operatorname{argmax}} \sum_{\boldsymbol{d}\in\mathcal{D}(\boldsymbol{x},\boldsymbol{y})} \exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})\right\} \tag{6.2}$$

6.2.1 Maximum Translation Sampling

Several alternative strategies for considering all derivations in the hypergraph during decoding have been explored in the literature. Li et al. (2009) develops a variational

approximation to the MAP estimate. Using n-gram models to construct a second distribution q that approximates $p(\boldsymbol{y}|\boldsymbol{x})$ by collapsing the latent derivations and giving a distribution over outputs for a given input, this scheme allows tractably finding the best translation.

Blunsom et al. (2008b) presented a beam-search approximation to the sum over all derivations. In a manner similar to cube-pruning for n-gram language model integration into a hypergraph (Huang and Chiang, 2007), Blunsom uses a beam search to prune the derivation forest when looking for the maximum probability translation, which results in a potentially biased approximation.

Thinking in a probabilistic framework, the problem we are faced with is approximate inference of the posterior distribution produced by a complex model. Recently, we have seen many general marginalization techniques (Li et al., 2009; Arun et al., 2009), and particularly sampling methods, being used in NLP for approximate inference. Arun et al. (2009) present a Markov Chain Monte Carlo technique that samples derivations from the posterior distribution of a phrase-based model. Their Gibbs sampler produces samples from the entire search space by drawing derivation and translation output pairs from $p(\boldsymbol{d}, \boldsymbol{y}|\boldsymbol{x})$. This can be used to determine the maximum derivation, or, by marginalizing over the derivations, an estimate of $p(\boldsymbol{y}|\boldsymbol{x})$ to obtain the maximum translation.

Although sampling the entire translation hypothesis space in this style is possible for a phrase-based model, it becomes prohibitively expensive for our hierarchical phrase-based model. Blunsom and Osborne (2008) presented another Monte-Carlo sampling technique which can sample derivations from the probability distribution defined by the SCFG model. Recall that a hypergraph encodes exponentially many trees, or derivations,

in a compact structure. Since each derivation has a score associated with it based on its features and weights, the hypergraph defines a probability distribution over derivations $p(\boldsymbol{d}, \boldsymbol{y}|\boldsymbol{x})$. Furthermore, it defines an implicit distribution over translations $p(\boldsymbol{y}|\boldsymbol{x})$. Blunsom and Osborne (2008) adapted INSIDE-OUTSIDE sampling (Goodman, 1998), a monolingual sampling technique designed for PCFG, to the SCFG translation model.

Using the INSIDE algorithm, we can compute the scores that define a multinomial distribution over all partial derivations available at a given rule. Processing top-down, we can draw derivations from the distribution defined by the inside scores. Here, we draw derivation samples from the LM pruned hypergraph, according to their probability. For each sample we can simply discard the actual derivation $\boldsymbol{d}$, keeping only the surface string $\boldsymbol{y}$. We can create a histogram of surface string occurrences, and assume the string with the highest count is the maximum probability translation. Blunsom and Osborne (2008) compare this method to the beam approximation in Blunsom et al. (2008b), and although it does not result in significantly higher BLEU scores, it is theoretically more justified. Thus, we utilize this sampling method for obtaining maximum probability translations as part of our new latent large-margin learning technique.

6.2.2 Cost-Augmented Inference

Recall from Chapter 3 that the hope and fear candidates are obtained from the k-best list by cost-augmented rescoring. In order to be able to sample the maximum probability translation for $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$ candidates from the hypergraph as just described, we need to rescore the hypergraph to include the cost on each rule, or hyperedge. Each edge repre-

sents an application of a translation rule, containing the features and associated feature values that are active when this rule is applied. In addition to the standard features described in previous chapters, we can think of the BLEU statistics and approximated score being additional features on the edge. However, these "features" always have a weight of 1 and are never optimized.

Since our cost function, BLEU, does not linearly decompose onto hyperedges like the other features, we need to make several minor approximations (Li and Khudanpur, 2009; Chiang, 2012). First, when computing the normal BLEU score, we clip the maximum number of times the hypothesis can get credit for a given n-gram to be the number of times that n-gram occurs in the reference. But since we are computing an approximate BLEU score based on a partial representation of the hypothesis as we build the hypergraph, accounting for how many times we have scored a given n-gram becomes intractable (Chiang et al., 2008a). Thus, we compute the *unclipped* n-gram match count.

Second, we follow the procedure of Li and Khudanpur (2009), which is similar to Dreyer et al. (2007) and Chiang et al. (2008a), and we rescore the hypergraph with the cost using a dynamic program in a manner similar to language model integration with cube pruning. In our setting, the major difference from LM integration is that we also need to maintain the BLEU statistics, i.e. the hypothesized and matched n-gram counts on each hyperedge. This additional information does not impose additional cost on the inference, as the dynamic program states depend only on the partial hypothesis length and left/right states, and not on the sufficient BLEU statistics associated with each node (Li and Khudanpur, 2009).

Putting this together, to get the necessary hope and fear candidates for maximum

translation optimization, we first construct the hypergraph for the source sentence in the standard manner. Then, we rescore the hypergraph using the reference translations in order to place an additional "feature", the cost, on each hyperedge in the hypergraph. We then extract the hope with a negative weight on the cost feature, and the fear with a positive weight on the cost feature. Theoretically, we could forgo cost-augmented inference and extract the candidates from a k-best list as in previous chapters. However, we believe this approach, which takes advantage of the whole space of hypotheses, is more principled and better suited for learning with latent variables.

6.3 Training Problem

6.3.1 Latent Variable Model

Previous work in SMT using a latent variable model has utilized Minimum Risk or expected BLEU (Rosti et al., 2010; Li and Eisner, 2009), and log-linear models (Blunsom et al., 2008b; Dyer and Resnik, 2010). Li and Eisner (2009) perform minimum risk training over hypergraphs by introducing a novel second-order expectation semiring. While we utilize a first-order semiring to compute feature expectations, which can also be used to compute the risk objective, computing the gradient of the risk objective requires a specialized second-order semiring.

Blunsom et al. (2008b) were the first to introduce a discriminative latent variable model for SMT by explicitly modeling derivations as a latent variable, and marginalizing over them during training and decoding to obtain the best translation. They maximized the regularized conditional marginal log-likelihood of the parallel training data

$\mathcal{T} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_1^n$ in a CRF model, where the conditional probability of a target translation and derivation pair is given by a log-linear model:

$$p(\boldsymbol{d}, \boldsymbol{y}|\boldsymbol{x}) = \frac{\exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})\right\}}{Z(\boldsymbol{x})} \tag{6.3}$$

The normalizing partition function $Z(x)$ sums over every derivation and translation:

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \sum_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}, \boldsymbol{y})} \exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})\right\} \tag{6.4}$$

The log-likelihood objective and corresponding gradient are:

$$\ell = \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}} \log p(\boldsymbol{y}|\boldsymbol{x}) \tag{6.5}$$

$$\frac{\partial \ell}{\partial \mathrm{w}_k} = \mathbb{E}_{p(\boldsymbol{d}|\boldsymbol{x}, \boldsymbol{y})}\left[f_k(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})\right] - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}\left[f_k(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})\right] \tag{6.6}$$

To compute the gradient, they need to obtain two separate hypergraphs.[1] The first represents the complete derivation forest given a source sentence. They perform unconstrained decoding of $\boldsymbol{x}$ to obtain a hypergraph representing all the possible resulting translations y' along with their derivations. By marginalizing over all the paths in this forest using the INSIDE-OUTSIDE algorithm, they obtain the feature expectations of the model given the source sentence, $\mathbb{E}_{p(\boldsymbol{d}|\boldsymbol{y}, \boldsymbol{x})}$, the first term in Eq. 6.6.

The second hypergraph contains only the derivations which produce the reference translation from the source. They perform constrained decoding, by constraining the

[1] Blunsom et al. (2008b) refer to them as packed charts.

target side $\boldsymbol{y}'$ to only produce the reference $\boldsymbol{y}$. The resulting hypergraph is a subset of the first one, constrained on both sides (source and reference), whereas the first hypergraph is only constrained on one side (source only). It is used to find the *reference* or *correct* expected feature values. By marginalizing over the paths in this hypergraph they are able to obtain the feature expectations of the model when constrained on both sides by the source and target reference, i.e. $\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}$. These feature expectations can be considered the empirical feature expectations with regard to the standard gradient of log-linear models.

In short, Blunsom et al. (2008b) want to maximize the likelihood of the training data given the model, and the likelihood can be expressed in such a way that when you differentiate with respect to a particular feature in the model, you get the difference in expectations of that feature value for the model given a source sentence, versus for the model when constrained on both sides by the source sentence and reference. This eliminates the need to have a k-best list of derivations against which to optimize, since instead of choosing just one derivation and placing our hopes on it, we distribute probability mass throughout the full derivation forest, placing more mass on the good paths at the expense of the bad paths.

This objective precludes the inclusion of an extrinsic loss, thus they were unable to target BLEU, or any other evaluation score normally used in SMT. Like Dyer and Resnik (2010), who also makes use of a log-linear model to incorporate latent variables, to be competitive with state-of-the-art SMT models optimized toward an error metric, they had to collapse any sparse features into dense features, and perform several optimization rounds of MERT.

We will similarly utilize constrained and unconstrained decoding to allow us to

update towards the maximum translation of the hope, and away from the maximum translation of the fear candidate, rather than optimizing toward a single derivation. An issue that arises here is that in standard large-margin approaches, where derivations are treated as translations, each derivation has a feature vector associated with it. One can simply use the feature vector that corresponds to the decoder's best derivation, and the feature vector for the correct translation, and compute the difference. With the latent variable model, however, we have a forest of derivations, where each derivation has its own set of features, and a probability associated with that derivation - a *distribution* over feature vectors for both the model and the reference.

We address this issue by creating feature vectors comprising the feature expectations computed by the INSIDE-OUTSIDE algorithm on the two hypergraphs resulting from the hope and fear translations. Instead of defining

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d}) \tag{6.7}$$

we replace the feature vector $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})$ with its expectation $\mathbb{E}\left[\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})\right]$ to get

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{w}^\top \mathbb{E}\left[\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})\right] = \mathbb{E}\left[\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})\right] \tag{6.8}$$

by linearity of $\mathbb{E}$. Thus, using the latent variable model, we can compute $\mathbb{E}\left[\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}^+)\right]$ and $\mathbb{E}\left[\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}^-)\right]$, and update $\mathbf{w}$ directly towards the maximum translation.

We will describe how to obtain the feature expectations and compute the update with them below.

6.3.2 Loss Functions

We begin our discussion of large-margin learning with latent variables by going back to a setting without latent variables. As we discussed in Section 2.4.2, in several popular methods, including CRFs and max-margin models, learning takes place by the minimization of a regularized loss function. As has been previously shown (Martins et al., 2010; Pletscher et al., 2010), the loss function for max-margin learning:

$$\ell_{\text{MM}} = -\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \left\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}) + \Delta_i(\boldsymbol{y}) \right\} \tag{6.9}$$

and CRF log-loss:

$$\ell_{\text{LL}} = -\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \exp \left\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}) \right\} \tag{6.10}$$

can be unified by introducing an inverse temperature β as:

$$\ell = -\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \frac{1}{\beta} \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \exp \left[\beta \left\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}) + \gamma \Delta_i(\boldsymbol{y}) \right\} \right] \tag{6.11}$$

This generalizes the loss of both regimes. It is easy to see that setting γ=0 and β=1 produces ℓ_{LL}. Following Theorem 8.1 (Wainwright and Jordan, 2008), we can produce ℓ_{MM} as the limiting case when $\beta \to \infty$ and γ=1. And as the bridge between the two, we can produce the softmax-margin ℓ_{SM} (Gimpel and Smith, 2010; Pletscher et al., 2010) with β=γ=1.

$$\ell_{\text{SM}} = -\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}_i)} \exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}) + \Delta_i(\boldsymbol{y})\right\} \tag{6.12}$$

This setting combines the benefits of probabilistic modeling with maximum entropy and margin maximization. As the equations above are presented in the standard form that posits reachable outputs, they need to be adapted accordingly for SMT.

6.3.3 Learning with Latent Variables

Now, we will examine how to include all the derivations that produce a single translation string in our optimization, with the goal that by marginalizing over these latent derivations, we can compute the margin between the expected feature values, not just the feature vector from a single derivation. More formally, the standard formulation of the constraints for a structured large-margin learner, as presented in Chapter 3 is:

$$\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{d}) - \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}) \geq \Delta_i(\boldsymbol{y}') - \xi_i \tag{6.13}$$

By marginalizing over the latent variables, we can obtain the expected feature function $\mathbb{E}\left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y})\right]$, and our constraints become:

$$\mathbf{w}^\top \mathbb{E}\left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i)\right] - \mathbf{w}^\top \mathbb{E}\left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}')\right] \geq \Delta_i(\boldsymbol{y}') - \xi_i \tag{6.14}$$

Crucially, what we care about is that the gradient of the objective function with respect to the weights, $\nabla \ell$, incorporate these expectations, since the gradient will directly

be used in the weight update at each iteration. Thus, the gradient takes the form

$$\nabla \ell = \mathbb{E}\left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i)\right] - \mathbb{E}\left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}')\right]$$

.

Here we introduce a unified representation of a **family of structured latent loss functions** which generalizes the max-margin latent variable formulation just described. This formulation is analogous to the unifying loss function in the case with no latent variables in Eq. 6.11 in the previous section. Instead of just a single inverse temperature β, we introduce two temperature variables, η_d, which only applies to the latent part of the function, $\boldsymbol{d}$, and β_y, which only applies to the observed part of the function, $\boldsymbol{y}$.

$$\begin{aligned} \ell_{\text{LV}} = -\frac{1}{\eta_d} \log \sum_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}, \boldsymbol{y}_i)} \exp\Big[\eta_d \Big\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{d}) \Big\}\Big] + \\ \frac{1}{\beta_y} \frac{1}{\eta_d} \log \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x}_i)} \sum_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}')} \exp\Big[\beta_y \eta_d \Big\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}) + \gamma \Delta_i(y') \Big\}\Big] \end{aligned} \tag{6.15}$$

Several well-known losses can be shown to emerge as special cases of this family. Setting γ=0 and β_y=η_d=1, we obtain the Hidden CRF (Quattoni et al., 2007; Blunsom et al., 2008b):

$$\ell_{\text{HCRF}} = -\log \sum_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{d})\right\} + \log \sum_{y' \in \mathcal{Y}(\boldsymbol{x}_i)} \sum_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}')} \exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d})\right\} \tag{6.16}$$

Setting γ=1 and β_y=$\eta_d \to \infty$, we recover the Latent SVM (Yu and Joachims, 2009):

$$\ell_{\text{LSVM}} = -\max_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{d}) + \max_{y' \in \mathcal{Y}(\boldsymbol{x}_i)} \max_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}')} \left\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}) + \Delta_i(y') \right\} \tag{6.17}$$

Setting γ=β_y=η_d=1 produces the latent variable softmax-margin (Gimpel and Smith, 2010):

$$\ell_{\text{LSMM}} = -\log \sum_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \exp\left\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{d}) \right\} + \log \sum_{y' \in \mathcal{Y}(\boldsymbol{x}_i)} \sum_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}')} \exp\Big\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}) + \Delta_i(y') \Big\} \tag{6.18}$$

Finally, as the unified view of the family above allows us to clearly see what has not yet been explored, here we introduce the **latent max-margin** formulation, with $\gamma = \eta_d = 1$ and $\beta_y \to \infty$

$$\ell_{\text{LMM}} = -\log \sum_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}_i)} \exp\left\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{d}) \right\} + \max_{y' \in \mathcal{Y}(\boldsymbol{x}_i)} \log \sum_{d \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}')} \exp\Big\{ \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}', \boldsymbol{d}) + \Delta_i(y') \Big\} \tag{6.19}$$

Eqs. 6.16- 6.19 are presented for clarity in a form that assumes that $\boldsymbol{y}_i$ is reachable, i.e. $\boldsymbol{y}_i \in \mathcal{Y}(\boldsymbol{x}_i)$. However, as we will be using surrogate references through cost-augmented and diminished decoding, the loss function employed will have another $\max$ operator on the first term of each equation to find $\boldsymbol{y}^+$. The SMT version of l_{LMM} is presented in Eq. 6.20.

$$\ell_{\text{LMM}_2} = -\max_{\boldsymbol{y}^+ \in \mathcal{Y}(\boldsymbol{x}_i)} \log \sum_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}^+)} \exp\Big\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^+, \boldsymbol{d}) - \Delta_i(\boldsymbol{y}^+)\Big\} + \max_{\boldsymbol{y}^- \in \mathcal{Y}(\boldsymbol{x}_i)} \log \sum_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{x}_i, \boldsymbol{y}^-)} \exp\Big\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}^-, \boldsymbol{d}) + \Delta_i(\boldsymbol{y}^-)\Big\} \tag{6.20}$$

ℓ_{LMM} is related to the first three objectives, but has a few important differences. The Latent SVM, ℓ_{LSVM}, is a well known model explicitly dealing with the introduction of latent variables into a large-margin model that was developed as an extension of structural SVMs to support latent variables. Notice that when the latent variable is trivial, i.e. $\mathcal{D}(\boldsymbol{x}, \boldsymbol{y})$ contains a single member, these losses reduce to their non-latent versions. In this case, ℓ_{LMM} and ℓ_{LSVM} become identical. This elucidates the major difference between the two in the latent setting as well: ℓ_{LSVM} and ℓ_{LMM} differ in their treatment of derivations, with ℓ_{LSVM} maximizing over the latent derivations by taking into account only the maximum scoring latent variable, whereas the other approaches marginalize over all of them. Thus, it does not use feature expectations, but rather the exact feature function $\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{d})$ for the maximum scoring $(\boldsymbol{y}, \boldsymbol{d})$ pair. Like ℓ_{LMM}, the margin in ℓ_{LSVM} is computed between two labels, $\boldsymbol{y}_i$, and $\boldsymbol{y}'$. ℓ_{LSMM} extends ℓ_{HCRF} with an additional cost, $\Delta_i(\boldsymbol{y}')$, and both compute a softmax margin, between $\boldsymbol{y}_i$ and a sum over all $\boldsymbol{y}'$.

To ensure that ℓ_{LMM} has feature expectations in the gradient as we desired, we take the partial derivative of ℓ_{LMM} with respect to the weights , and obtain:

$$\nabla \ell_{\text{LMM}} = \mathbb{E}_{p(\boldsymbol{d}|\boldsymbol{x}_i, \boldsymbol{y}_i)}\left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i)\right] - \mathbb{E}_{p(\boldsymbol{d}|\boldsymbol{x}_i, \boldsymbol{y}')}\left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}')\right] \tag{6.21}$$

In contrast, the gradient of ℓ_{LSMM} and ℓ_{HCRF} is

$$\nabla \ell_{\text{LSMM}} = \nabla \ell_{\text{HCRF}} = \mathbb{E}_{p(\boldsymbol{d}|\boldsymbol{x}_i, \boldsymbol{y}_i)} \left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) \right] - \mathbb{E}_{p(\boldsymbol{d}, \boldsymbol{y}'|\boldsymbol{x}_i)} \left[\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{y}') \right] \tag{6.22}$$

All three are similar in that we want to learn parameters to produce a score for the correct output that is better than some function of the other scores. The difference between $\nabla \ell_{\text{LMM}}$ and the other two, $\nabla \ell_{\text{LSMM}}$ and $\nabla \ell_{\text{HCRF}}$, lies in the probability distribution being used to calculate the second expectation. For $\nabla \ell_{\text{LSMM}}$ and $\nabla \ell_{\text{HCRF}}$, we are maximizing the difference between the expected feature values of the correct output and *all possible* outputs. In contrast, for $\nabla \ell_{\text{LMM}}$, we are maximizing the difference between the expected feature values of the correct output and the *incorrect* outputs, exemplified by the output $\boldsymbol{y}^-$ that violates the constraint the most.

6.3.4 Expected Feature Computation

For ℓ_{LMM}, we can compute expectations by first computing marginal probabilities using the INSIDE-OUTSIDE algorithm over the two forests of derivations, as in Blunsom et al. (2008b).

Recall from Chapter 2 that synchronous parsing is the procedure utilized in decoding using a SCFG translation model. Dyer (2010b) presented an alternative view of the synchronous parsing algorithm which decomposed it into two successive monolingual parsing operations. The first parse can be seen as computing the composition of the source sentence, represented as an FST, with our SCFG translation model. The composi-

tion produces a hypergraph that can only generate $\boldsymbol{x}$ on the source side, but can generate all possible translations of $\boldsymbol{x}$ on the target side that can be obtained with our translation model. Composing this hypergraph again with an FST representing the target sentence $\boldsymbol{y}$ results in a new hypergraph that can only produce $\boldsymbol{x}$ on the source side and $\boldsymbol{y}$ on the target side.

We will need to construct three hypergraphs per sentence: by performing unconstrained decoding once, and constrained twice. The first decoding will be unconstrained to produce the hypergraph representing the whole translation hypothesis space. From this we obtain $\boldsymbol{y}'$, and through cost-augmented inference, the fear translation $\boldsymbol{y}^-$, and hope $\boldsymbol{y}^+$. The second decoding will be constrained to get the expected feature values for the source and hope pair, and can be viewed as a composition operation of the full hypergraph from above with the hope sentence, constraining the forest to only those derivations that produce the hope sentence on the target side. Similarly, we will compose the full grammar with the fear sentence to get the expected feature values for the source and fear pair. These constrained decoding runs give us the expectations under the model. With these expectations calculated, we can perform the standard large-margin update, as described in Chapter 3.

Both in training and evaluation, we can obtain the desired hypotheses using maximum translation decoding via the sampling described above, or standard maximum derivation decoding.

6.4 Evaluation

6.4.1 Setup

To evaluate the advantage of explicitly accounting for latent variables in the optimization and performing maximum probability translation decoding, we conduct experiments on Chinese-English and Arabic-English translation.

As our training data we utilize the non-UN and non-HK Hansards portions of the NIST training corpora, amounting to 1.6 million sentence pairs for Chinese and 1 million sentence pairs for Arabic. The data was lowercased, tokenized and aligned using GIZA++ (Och and Ney, 2003) to obtain bidirectional alignments, which were symmetrized using the `grow-diag-final-and` method (Koehn et al., 2003). The Chinese data were segmented using the Stanford segmenter. Arabic data were preprocessed with an HMM segmenter (Lee et al., 2003). We trained a 4-gram LM on the English side of each corpus with additional data randomly selected from the non-NYT and non-LAT portions of the Gigaword v4 corpus using modified Kneser-Ney smoothing (Chen and Goodman, 1996). We tuned the parameters of the system to optimize BLEU (Papineni et al., 2002b) on the NIST MT06 tuning corpus.

We use a standard maximum-derivation MIRA optimizer as our baseline, but note that it is now computing the cost-augmented and diminished hypotheses over the entire hypergraph, not from a k-best approximation. More specifically, since we made approximations to the BLEU score when decomposing it over the hypergraph, instead of relying on the one-best hope and fear, we extract a k-best hope list from a hypergraph rescored

with a BLEU feature on each edge, and k-best fear list from the same hypergraph rescored with -BLEU, and select our hope and fear from those lists, respectively.

We compare the baseline against several alternatives that explicitly account for the latent structure. For tuning, all include marginalizing over the paths in the forest to obtain translations, but they differ in how we obtain the hope and fear candidates from the cost-augmented forest. In the first model, LMM$_d$, we extract hope and fear as above, using a standard maximum derivation decoding. In the second, LMM$_t$, we sample from the forest to obtain the maximum probability translations for the hope and fear. In both cases, the candidates we obtain will be composed with the full forest as described above to obtain the maximum translation feature expectations. We also include a maximum translation softmax-margin optimizer, LSMM.

The regularization strength C was set to 0.01 in all cases. The oracle weight for cost-augmented decoding is set to 1, and as previously stated, is only used for selecting the hope and fear hypotheses and then excluded from all optimization.

For decoding, we can decode the test set with (1) maximum derivation or (2) maximum translation decoding by sampling, even if we did not train with latent variables or perform maximum translation sampling while tuning. All results are averaged over 2 runs. We use MultEval to estimate statistical significance.

6.4.2 Results

Results for Chinese-English translation with baseline features using the latent max-margin, LMM$_d$ and LMM$_t$, latent softmax-margin, LSMM, and MIRA learners are pre-

sented in Table 6.2. We first note that the standard condition is derivation training with derivation decoding, presented in the first row, and first column of each test set. Simply using the derivation trained MIRA and decoding for maximum translation, we already observe a 0.5 BLEU and 0.9 TER improvement. This indicates that performing maximum translation decoding is useful in of itself, even without pairing it with an analogous training regime.

Looking at the LMM_d results, we first note that the BLEU score even for derivation decoding is up to 0.3 BLEU points better than MIRA derivation decoding, suggesting the complementary direction: learning toward translations, even when not decoding toward them, imbues some additional useful information to the system. Moving to translation decoding, we observe an additional small gain of 0.2 BLEU and over 1 TER improvement over maximum translation MIRA.

We see that LSMM performance paired with derivation decoding is quite bad, and although it improves substantially in translation decoding, it is still lagging far behind that of MIRA and LMM_d. Finally, LMM_t further improves upon LMM_d, gaining in derivation decoding, and up to 0.9 BLEU and 1.5 TER for translation decoding. This results in overall gains of up to 0.9 BLEU and over 2 TER over the best MIRA results.

While the LMM_t results in the low-dimensional feature setting are already encouraging, we believe that a larger feature set with sparse features should benefit more from feature expectations than the dense feature used thus far. Thus, we conducted sparse feature experiments comparing the models. Moving to sparse features in Table 6.1, we again see significant gains going from MIRA to LMM_d to LMM_t. As compared to the baseline feature set, we see improvements from sparse features for MIRA and LMM_d, but small

gains for LMM$_t$.

Training	Decoding							
	MT03				MT05			
	derivation		translation		derivation		translation	
	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER
MIRA (derivation)	36.18	59.41	36.72	58.89	32.57	62.85	32.96	62.00
LSMM (translation)	32.81	62.42	33.55	61.82	29.06	66.10	29.53	65.61
LMM$_d$ (derivation)	36.48	57.97	36.88	57.51	32.66	60.89	33.01	60.65
LMM$_t$ (translation)	**36.99**	**57.05**	**37.13**	**56.38**	**33.58**	**59.67**	**33.87**	**59.17**

Table 6.1: Results on Zh-En translation using maximum derivation and maximum translation with base features. Improvement of LMM (translation) over MIRA is significant at the p <0.001 level for both decoding regimes in TER and BLEU.

Training	Decoding							
	MT03				MT05			
	derivation		translation		derivation		translation	
	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER
MIRA (derivation)	36.29	58.94	36.57	58.49	32.94	62.03	33.12	61.76
LMM$_d$ (derivation)	**36.69**	**56.73**	36.84	**56.30**	**33.28**	**60.32**	**33.47**	59.98
LMM$_t$ (translation)	**37.02**	**57.11**	**37.24**	**56.66**	**33.48**	**59.97**	**33.76**	**59.41**

Table 6.2: Results on Zh-En translation using maximum derivation and maximum translation with sparse features. Improvement of LMM (translation) over MIRA is significant at the p <0.001 level for both decoding regimes in TER and BLEU.

Similar results for Arabic-English translation are presented in Table 6.3. When moving MIRA to translation decoding, we see a 0.7 BLEU and 0.4 TER improvement. Moving to the LMM$_d$ results, we note that the BLEU score for derivation and translation decoding is much better than MIRA in one case, and worse in another, while TER is better for both. Once again, we see that LSMM performance with derivation decoding is bad,

Training	Decoding							
	MT05				MT08			
	derivation		translation		derivation		translation	
	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER
MIRA (derivation)	52.25	40.86	52.99	40.38	**41.32**	50.73	**41.71**	50.34
LSMM (translation)	48.68	44.55	49.25	44.13	38.57	53.96	39.07	53.50
LMM_d (derivation)	**53.45**	**39.32**	53.92	39.04	40.44	49.69	40.86	**49.34**
LMM_t (translation)	**53.37**	**39.66**	**54.30**	**38.86**	**41.03**$^-$	**49.67**	**41.48**	**49.21**

Table 6.3: Results on Arabic translation using maximum derivation and maximum translation with base features. Improvement of LMM (translation) over MIRA is significant at the p <0.001 level in TER and BLEU unless otherwise indicated: ($^-$) indicates significance at $p < 0.05$ level.

and although it improves substantially from derivation to translation, it is still far behind that of MIRA and LMM_d. LMM_t improves upon the performance of LMM_d, gaining in all but one case (derivation decoding on MT05). Focusing on that condition, it seems that LMM_d overfit, and thus produced great results there, and seriously underperformed on MT08. Meanwhile, LMM_t improved upon MT05, and has comparable BLEU performance with MIRA, with significantly better TER.

In the sparse condition, we again see LMM_t outperforming all other models by significant margins in both BLEU and TER.

6.5 Discussion

Table 6.5 presents the active feature counts for MIRA, LMM_d, and LMM_t on both language pairs. In both settings, LMM_d and LMM_t activate considerably more features than MIRA, showing that marginalizing over alternative derivations for the hope and fear

Training	Decoding							
	MT05				MT08			
	derivation		translation		derivation		translation	
	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER	↑BLEU	↓TER
MIRA (derivation)	53.06	40.08	53.45	39.75	41.62	49.97	41.78	49.68
LMM$_d$ (derivation)	52.85	40.13	53.59	39.55	41.73	50.01	41.83	49.89
LMM$_t$ (translation)	**54.00**	**39.19**	**54.44**	**39.01**	**41.97**	**49.59**	**42.26**	**49.09**

Table 6.4: Results on Arabic translation using maximum derivation and maximum translation with sparse features. Improvement of LMM (translation) over MIRA is significant at the p <0.001 level for both decoding regimes in TER and BLEU.

hypotheses introduces more features of the pair into the optimization.

Optimizer	Chinese	Arabic
MIRA	102k	113k
LMM$_d$	181k	308k
LMM$_t$	286k	454k

Table 6.5: Active sparse features for MIRA, LMM$_d$, and LMM$_t$.

Figures 6.2 and 6.3 compare the tuning performance across the different learners. We first see that all learners are stable, never deviating much from their best performing iteration. On Chinese-English translation, MIRA, LMM$_t$, and LMM$_d$ are practically indistinguishable on the tuning set, yet LMM$_t$ significantly outperforms the others on the test sets. On Arabic-English, MIRA and LMM$_t$ are very close, with LMM$_d$ slightly lower, and on test LMM$_t$ once again outperforms the others. In both cases, LSMM has a lower performance across all iterations, being approximately 3 BLEU points lower on tuning than the others, but improving slightly with time. This transfers to significantly lower performance on the test sets. Interestingly, LMM$_t$ quickly excels to its optimal

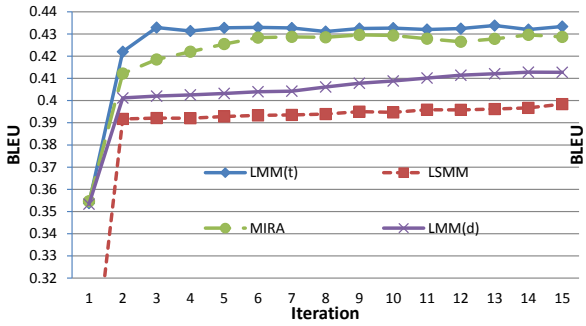


Figure 6.2: Comparison of performance on the tuning set for ar-en between LMM_d, LMM_t, LSMM, and MIRA.

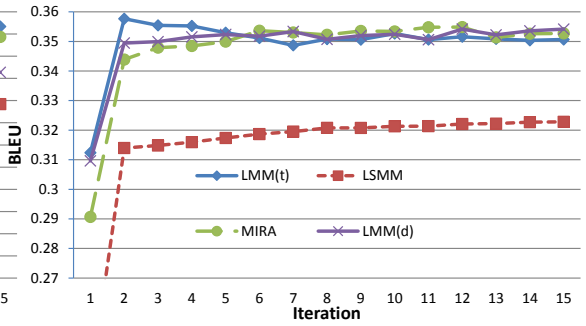


Figure 6.3: Comparison of performance on the tuning set for zh-en between LMM_d, LMM_t, LSMM, and MIRA.

Method	Chinese-English			Arabic-English		
	Ave (min/iter)	Std	Sent/Min	Ave (min/iter)	Std	Sent/Min
MIRA (derivation)	4.77	1.02	349	7.67	0.54	234
LMM (derivation)	4.3	0.74	386	8.43	1.38	213
LMM (translation)	5.4	0.55	308	12.38	3.38	145

Table 6.6: Wallclock time per iteration for MIRA, LMM_d and LMM_t.

performance within the first few iterations, and stays within close proximity to it throughout tuning. The other optimizers also reach their respective optimal performances fairly quickly, but keep improving with later iterations. Thus, we can see that not only does LMM_t produce a better model, but it does so more efficiently. Table 6.6 presents a comparison of time taken by each learner per iteration. While explicitly accounting for latent derivations and sampling for the maximum probability translation does incur computation cost, it still remains efficient, especially when considering that fewer iterations are necessary to achieve optimal performance.

6.6 Summary

In this chapter we presented a novel optimization procedure for maximum probability translation learning and inference, thereby introducing a new way of combining a probabilistic framework for dealing with latent variables with a geometric model that performs margin maximization. We presented a unified representation of latent variable objectives as a family of latent structured losses, and developing a novel loss function for large-margin learning in the latent variable setting which explicitly accounts for derivational ambiguity. Empirical evaluation of this loss showed significant gains in two language settings over standard MIRA training with Viterbi decoding.

7 Conclusion

> *The strongest arguments prove nothing so long as the conclusions are not verified by experience. Experimental science is the queen of sciences and the goal of all speculation.*
>
> — Roger Bacon

> *We are in the ordinary position of scientists of having to be content with piecemeal improvements: we can make several things clearer, but we cannot make anything clear.*
>
> — Frank Plumpton Ramsay

Designing learning algorithms that are capable of efficiently learning models in high-dimensions over large training sets is a core challenge in machine learning. Throughout this dissertation, we introduced novel algorithms and methods focused on improving scalability and generalization for feature-rich learning in SMT. We approached the problem of learning in high-dimensional feature spaces for SMT from several perspectives: what to optimize, how to optimize, and what features to use. The contributions in these areas as well as future work are discussed below.

In Chapter 3, we presented a generalized loss for cost-augmented learning with latent variables, and performed an extensive empirical evaluation of different strategies employed in constructing a state-of-the-art SMT system. We examined learning charac-

teristics across objectives, solutions for the parameter update, parallelization, and introduction of sparse features, and showed the cost-augmented hypothesis selection is important for learning stability and generalization ability. We then introduced the capability for large-scale training by developing a scalable learner on a highly distributed architecture.

In Chapter 4 we further improved our large-margin learner by introducing a loss for structured relative margin maximization that incorporates spread information into the optimization procedure, thereby focusing on creating the margin in the correct direction. We introduced a novel online margin-based algorithm designed for optimizing high-dimensional feature spaces, which introduces constraints into a large-margin optimizer that bound the spread of the projection of the data while maximizing the margin. We showed that bounding the spread significantly improves translation performance.

In Chapter 5 we utilized our learning capabilities and extended translation modeling for dynamic domain adaptation. We abstracted the usual notion of a domain to finer-grained topic distributions induced in an unsupervised fashion. Utilizing our learning algorithms from Chapters 2 and 4, we showed that incorporating lexical weighting features conditioned on soft domain membership directly into our model is an effective strategy for dynamically biasing SMT towards relevant translations.

Finally, in Chapter 6 we defined a framework for latent variable models which explicitly accounted for the derivational ambiguity problem posed by our translation model in both learning and decoding. We presented a novel loss function from a family of latent structured losses for large-margin learning in the latent variable setting, and developed a suitable optimization algorithm for training an SMT system with this objective.

7.1 Future Work

7.1.1 Large-Scale Discriminative Training

This dissertation has taken steps toward large-scale training, by developing fast online learning algorithms and implementing them on a scalable distributed architecture, showing that such training is feasible. Despite several recent advances in large-scale training which utilize large numbers of features (Simianer et al., 2012; Flanigan et al., 2013; Yu et al., 2013), while we were able to show improvement in translation performance with sparse features, we were unable to achieveresults indicating that large bitext tuning produces gains in translation quality on domain specific evaluation sets over the sparse feature tuning on smaller tuning sets. The causes of this were discussed in Chapter 3, including domain mismatch problems, and lack of requisite features for taking advantage of the larger tuning set.

One avenue of exploration would be to analyze how to properly select sentences from the bitext in such a way as to maximize similarity with existing in-domain tuning sets. This could involve methods from information retrieval and domain adaptation. Another possibility it to examine the grammar rules themselves, and either remove, i.e. hard constrain, or weight, i.e. soft constrain, rules based on some measure of general applicability and generalization.

7.1.2 Structured Prediction (beyond SMT)

As SMT is an instance of the more general structured prediction problem, the algorithms we introduced in this dissertation have the capability to be applicable to a number of other problems in and outside of NLP. Online learning is also becoming popular for its ability to scale with the data and handle sparse features. For instance, parsing has benefited from large-margin learning, and is thus a natural fit for the improvements we have introduced (McDonald and Pereira, 2006). Although SMT introduces problems with regard to the reference translation that are somewhat unique, the field of structured multi-label classification is emerging as important in other communities, and thus the exploration and techniques we have described could well be of use there as well (Lampert, 2011). Furthermore, computational biology and vision both have settings that require latent variables and predicting a complex structure (Nowozin and Lampert, 2011).

7.1.3 Kernel Learning

The joint feature map $\mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ we use maps the input and output pair into a feature space: $\mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{H}$, which can be defined explicitly, as is in the formulation we have been using for our linear model. However, it can also be *implicitly* defined through a kernel function $k : \mathcal{X} \times \mathcal{Y} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ (Bishop, 2006). Kernel functions are a principal advantage to large-margin learning with SVM and MIRA, and are widely used in binary, multiclass, and structured prediction settings outside SMT, since they allow non-linear learning by mapping the input into a higher-dimensional space. However, to the best of our knowledge, the use of kernels for machine translation has not yet been explored. This

would involve defining the right kernel to use, and how to factor it in while performing inference over packed representations such as hypergraphs. In general, non-linear learning is an area which has received little attention for SMT, although there has been a recent resurgence in using neural networks for language and translation modeling (Son et al., 2012; Servan and Schwenk, 2011).

7.1.4 Improved Topic Modeling

Although we showed in Chapter 5 that by inducing unsupervised domains using topic models we could bias our translation model, the topic model we used was a standard Vanilla LDA implementation. We only modeled the source side and ignored the relationship between the source and target languages. However, if a word is ambiguous in the source language, we may not be able to place it in the correct topic, thereby reducing our ability to model the document and translate it correctly. By modeling both the source and target language, either with polylingual LDA (Mimno et al., 2009) or tree-based topic models (Hu and Boyd-Graber, 2012), we can introduce information about the target language and learn topics that better fit the aligned corpora. In this case, if the target topic distribution is unambiguous, or has a better topic distribution, it can help adjust the source documents topic distribution toward the desired context.

7.1.5 Semantic Features

Coming from a background in philosophy, we have been enchanted by the proposition of semantic features since beginning this research. Alas, thus far they have had

limited success in SMT. Semantic Role Labeling (SRL) has been gaining some traction recently. SRL features have been used to help predict reordering (Li et al., 2013; Xiong et al., 2012), as well as serving as a soft-constraint on production rules, in the same spirit as soft-syntactic constraints. Work moving even further in this direction and outside of our hierarchical model of translation is aimed at constructing semantic meaning representations of sentences, and using them to generate sentences for translations (Jones et al., 2012). Regardless of the translation paradigm, our learning algorithms allow the introduction of expressive semantic features, allowing the progression of research in developing such features that can aid translation.

Bibliography

Arun, A., Dyer, C., Haddow, B., Blunsom, P., Lopez, A., and Koehn, P. (2009). Monte carlo inference and maximization for phrase-based translation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 102–110, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arun, A., Haddow, B., and Koehn, P. (2010). A unified approach to minimum risk training and decoding. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10.

Arun, A. and Koehn, P. (2007). Online learning methods for discriminative training of phrase based statistical machine translation. In *MT Summit XI*.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Bahl, L. R., Jelinek, F., and Mercer, R. (1983). A maximum likelihood approach to continuous speech recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-5(2):179–190.

Bar-Hillel, Y., Perles, M. A., and Shamir, E. (1961). On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, (14):143–172.

Bartlett, P. L. and Mendelson, S. (2003). Rademacher and gaussian complexities: risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Blei, D. M., Ng, A. Y., Jordan, M. I., and Lafferty, J. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:2003.

Blunsom, P., Cohn, T., and Osborne, M. (2008a). Bayesian synchronous grammar induction. In *NIPS*.

Blunsom, P., Cohn, T., and Osborne, M. (2008b). A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, Columbus, Ohio.

Blunsom, P. and Osborne, M. (2008). Probabilistic inference for machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, Hawaii.

Bottou, L. and Bousquet, O. (2011). The tradeoffs of large scale learning. In Sra, S., Nowozin, S., and Wright, S. J., editors, *Optimization for Machine Learning*, pages 351–368. MIT Press.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Boyd-Graber, J. and Resnik, P. (2010). Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85.

Brown, P. F., Pietra, V. J., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Carpuat, M., Daume III, H., Henry, K., Irvine, A., Jagarlamudi, J., and Rudinger, R. (2013). Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria.

Cer, D., Jurafsky, D., and Manning, C. D. (2008). Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 26–34.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. (2005). A second-order perceptron algorithm. *SIAM J. Comput.*, 34(3):640–668.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318.

Chen, W., Liu, T.-Y., Lan, Y., Ma, Z., and Li, H. (2009). Ranking measures and loss functions in learning to rank. In *NIPS*.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of NAACL*.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *In ACL*, pages 263–270.

Chiang, D. (2007). Hierarchical phrase-based translation. In *Computational Linguistics*, volume 33(2), pages 201–228.

Chiang, D. (2012). Hope and fear for discriminative training of statistical translation models. *J. Machine Learning Research*.

Chiang, D., DeNeefe, S., and Pust, M. (2011). Two easy improvements to lexical weighting. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, Stroudsburg, PA, USA.

Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 218–226.

Chiang, D., Marton, Y., and Resnik, P. (2008a). Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, Hawaii.

Chiang, D., Marton, Y., and Resnik, P. (2008b). Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Waikiki, Honolulu, Hawaii.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 176–181, Stroudsburg, PA, USA.

Collins, M. (2002). Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 1–8.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585.

Crammer, K., Dredze, M., and Pereira, F. (2012). Confidence-weighted linear classification for text categorization. *J. Mach. Learn. Res.*, 98888:1891–1926.

Crammer, K., Kulesza, A., and Dredze, M. (2009a). Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 22*, pages 414–422.

Crammer, K., Mohri, M., and Pereira, F. (2009b). Gaussian margin machines. *Journal of Machine Learning Research - Proceedings Track*, 5:105–112.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

Daumé, III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic.

Daumé, III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. *J. Artif. Int. Res.*, 26(1):101–126.

Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. In *OSDI*.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113.

Dempster, A. P., Laird, M. N., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 39:1–22.

Dredze, M. and Crammer, K. (2008). Confidence-weighted linear classification. In *In ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM.

Dreyer, M., Hall, K., and Khudanpur, S. (2007). Comparing reordering constraints for smt using efficient bleu oracle computation. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, SSST '07, pages 103–110.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Duh, K., Sudoh, K., Tsukada, H., Isozaki, H., and Nagata, M. (2010). N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 375–383.

Dyer, C. (2010a). *A Formal Model of Ambiguity and its Applications in Machine Translation*. PhD thesis, University of Maryland.

Dyer, C. (2010b). Two monolingual parses are better than one (synchronous parse). In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, California.

Dyer, C., Lopez, A., Ganitkevitch, J., Weese, J., Ture, F., Blunsom, P., Setiawan, H., Eidelman, V., and Resnik, P. (2010). cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL System Demonstrations*.

Dyer, C. and Resnik, P. (2010). Context-free reordering, finite-state translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 858–866.

Eidelman, V. (2012). Optimization strategies for online large-margin learning in machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*.

Eidelman, V., Boyd-Graber, J., and Resnik, P. (2012). Topic models for dynamic translation model adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 115–119.

Eidelman, V., Marton, Y., and Resnik, P. (2013a). Online relative margin maximization for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1116–1126.

Eidelman, V., Wu, K., Ture, F., Resnik, P., and Lin, J. (2013b). Mr. mira: Open-source large-margin structured learning on mapreduce. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Sofia, Bulgaria. Association for Computational Linguistics.

Eidelman, V., Wu, K., Ture, F., Resnik, P., and Lin, J. (2013c). Towards efficient large-scale feature-rich statistical machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, Sofia, Bulgaria. Association for Computational Linguistics.

Finch, A. and Sumita, E. (2008). Dynamic model interpolation for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 208–215.

Flanigan, J., Dyer, C., and Carbonell, J. (2013). Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Foster, G., Goutte, C., and Kuhn, R. (2010). Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA. Association for Computational Linguistics.

Foster, G. and Kuhn, R. (2007). Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 128–135.

Foster, G. and Kuhn, R. (2009). Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 242–249, Athens, Greece. Association for Computational Linguistics.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 961–968.

Gallo, G., Longo, G., Pallottino, S., and Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete Appl. Math.*, 42(2-3):177–201.

Gimpel, K. and Smith, N. A. (2010). Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*.

Gimpel, K. and Smith, N. A. (2012). Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics*.

Goodman, J. T. (1998). *Parsing inside-out*. PhD thesis, Harvard University, Cambridge, MA, USA.

Green, S., Wang, S., Cer, D., and Manning, C. (2013). Fast and adaptive online training of feature-rich translation models. In *ACL*.

Gruber, A., Weiss, Y., and Rosen-Zvi, M. (2007). Hidden topic markov models. *Journal of Machine Learning Research - Proceedings Track*, 2:163–170.

Haddow, B., Arun, A., and Koehn, P. (2011). Samplerank training for phrase-based machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland. Association for Computational Linguistics.

Hasler, E., Haddow, B., and Koehn, P. (2012). Sparse lexicalised features and topic adaptation for SMT. In *Proceedings of IWSLT*.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer New York Inc., New York, NY, USA.

Heafield, K. (2011). Kenlm: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Hu, Y. and Boyd-Graber, J. (2012). Efficient tree-based topic modeling.

Huang, L. and Chiang, D. (2005). Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64.

Huang, L. and Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151.

Huang, L., Knight, K., and Joshi, A. (2006). A syntax-directed translator with extended domain of locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, CHSLP '06, pages 1–8.

Hutchins, W. J. and Somers, H. L. (1992). *An Introduction to Machine Translation*. Academic Press.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Nédellec, C. and Rouveirol, C., editors, *European Conference on Machine Learning*, pages 137–142, Berlin. Springer.

Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59.

Jones, B., Andreas, J., Bauer, D., Hermann, K. M., and Knight, K. (2012). Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of COLING 2012*, Mumbai, India.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proceedings of IEEE Internation Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86. AAMT, AAMT.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, Stroudsburg, PA, USA.

Koehn, P. and Schroeder, J. (2007). Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07.

Kumar, S. and Byrne, W. (2004). Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*.

Kumar, S., Macherey, W., Dyer, C., and Och, F. (2009). Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289.

Lampert, C. H. (2011). Maximum margin multi-label structured prediction. In *Advances in Neural Information Processing Systems 24*, pages 289–297.

Lee, Y.-S., Papineni, K., Roukos, S., Emam, O., and Hassan, H. (2003). Language model based Arabic word segmentation. In *ACL '03: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 399–406.

Li, J., Resnik, P., and Daumé III, H. (2013). Modeling syntactic and semantic structures in hierarchical phrase-based translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia.

Li, Z. and Eisner, J. (2009). First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09.

Li, Z., Eisner, J., and Khudanpur, S. (2009). Variational decoding for statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09.

Li, Z. and Khudanpur, S. (2009). Efficient extraction of oracle-best translations from hypergraphs. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09.

Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. (2006). An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 761–768.

Lin, C.-Y. and Och, F. J. (2004). Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*.

Lin, J. and Dyer, C. (2010). *Data-Intensive Text Processing with MapReduce*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Liu, D. C. and Nocedal, J. (1989). On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45:503–528.

Liu, Y., Lü, Y., and Liu, Q. (2009). Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 558–566.

Lopez, A. (2007). Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP*, pages 976–985.

Lopez, A. (2008). Statistical machine translation. *ACM Computing Survey*, 40(3).

Macháček, M. and Bojar, O. (2013). Results of the WMT13 metrics shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria. Association for Computational Linguistics.

Macherey, W., Och, F. J., Thayer, I., and Uszkoreit, J. (2008). Lattice-based minimum error rate training for statistical machine translation. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 725–734, Morristown, NJ, USA. Association for Computational Linguistics.

Madnani, N. (2010). *The Circle of Meaning: From Translation to Paraphrasing and Back*. PhD thesis, Department of Computer Science, University of Maryland College Park.

Martins, A. F. T., Gimpel, K., Smith, N. A., Xing, E. P., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2010). Learning structured classifiers with dual coordinate descent. Technical Report CMU-ML-10-109, Carnegie Mellon University.

Matsoukas, S., Rosti, A.-V. I., and Zhang, B. (2009). Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore. Association for Computational Linguistics.

McAllester, D., Hazan, T., and Keshet, J. (2010). Direct loss minimization for structured prediction. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 1594–1602.

McAllester, D. and Keshet, J. (2011). Generalization bounds and consistency for latent structural probit and ramp loss. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 2205–2212.

Mccallum, A. K. (2002). MALLET: A Machine Learning for Language Toolkit.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05. Association for Computational Linguistics.

McDonald, R., Hall, K., and Mann, G. (2010). Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464, Los Angeles, California.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006))*, volume 6, pages 81–88.

Mimno, D., Wallach, H. M., Naradowsky, J., Smith, D. A., and Mccallum, A. (2009). Polylingual topic models. In *In EMNLP*.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Nowozin, S. and Lampert, C. H. (2011). Structured learning and prediction in computer vision. *Found. Trends. Comput. Graph. Vis.*, 6(3–4):185–365.

Och, F. and Ney, H. (2003). A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 29(21), pages 19–51.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302.

Och, F. J. and Ney, H. (2004). The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002a). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002b). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING.

Pletscher, P., Ong, C. S., and Buhmann, J. M. (2010). Entropy and margin maximization for structured output learning. In Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M., editors, *Proceedings of the 20th European Conference on Machine Learning (ECML)*, volume 6321 of *Lecture Notes in Computer Science*, pages 83–98.

Quattoni, A., Wang, S., p Morency, L., Collins, M., Darrell, T., and Csail, M. (2007). Hidden-state conditional random fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ratliff, N., Bagnell, J. A., and Zinkevich, M. (2006). Subgradient methods for maximum margin structured learning. In *Proceedings of the ICML Workshop on Learning in Structured Output Spaces*.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rosti, A.-V., Zhang, B., Matsoukas, S., and Schwartz, R. (2010). Bbn system description for wmt10 system combination task. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, Uppsala, Sweden. Association for Computational Linguistics.

Rosti, A.-V., Zhang, B., Matsoukas, S., and Schwartz, R. (2011). Expected bleu training for graphs: Bbn system description for wmt11 system combination task. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 159–165, Edinburgh, Scotland. Association for Computational Linguistics.

Rush, A., Chang, Y.-W., and Collins, M. (2013). Optimal beam search for machine translation. In *EMNLP*.

Schaul, T. and LeCun, Y. (2013). Adaptive learning rates and parallelization for stochastic, sparse, non-smooth gradients. In *International Conference on Learning Representations*, Scottsdale, AZ.

Schaul, T., Zhang, S., and LeCun, Y. (2013). No More Pesky Learning Rates. In *International Conference on Machine Learning (ICML)*.

Servan, C. and Schwenk, H. (2011). Optimising multiple metrics with mert. *Prague Bull. Math. Linguistics*, 96:109–118.

Shen, L. (2007). Guided learning for bidirectional sequence classification. In *In ACL*.

Shen, L., Xu, J., and Weischedel, R. (2010). String-to-dependency statistical machine translation. *Comput. Linguist.*, 36(4):649–671.

Shivaswamy, P. and Jebara, T. (2009a). Structured prediction with relative margin. In *In International Conference on Machine Learning and Applications*.

Shivaswamy, P. K. and Jebara, T. (2009b). Relative margin machines. In *In Advances in Neural Information Processing Systems 21*. MIT Press.

Sima'an, K. (1996). Computational complexity of probabilistic disambiguation by means of tree-grammars.

Simianer, P., Riezler, S., and Dyer, C. (2012). Joint feature selection in distributed stochastic learning for large-scale discriminative training in smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea.

Smith, D. A. and Eisner, J. (2006). Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia. Association for Computational Linguistics.

Smith, N. A. (2011). *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Son, L. H., Allauzen, A., and Yvon, F. (2012). Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12.

Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. In *Intl. Conf. on Spoken Language Processing*.

Su, J., Wu, H., Wang, H., Chen, Y., Shi, X., Dong, H., and Liu, Q. (2012). Translation model adaptation for statistical machine translation with monolingual topic information. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea.

Tam, Y.-C., Lane, I., and Schultz, T. (2007). Bilingual lsa-based adaptation for statistical machine translation. *Machine Translation*, 21(4):187–207.

Tan, M., Xia, T., Wang, S., and Zhou, B. (2013). A corpus level MIRA tuning strategy for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA.

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. (2005). Learning structured prediction models: a large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 896–903.

Taskar, B., Guestrin, C., and Koller, D. (2004). Max-margin markov networks. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Taskar, B., Lacoste-Julien, S., and Jordan, M. I. (2006). Structured prediction, dual extragradient and bregman projections. *J. Mach. Learn. Res.*, 7:1627–1653.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Tillmann, C. and Zhang, T. (2006). A discriminative global training algorithm for statistical mt. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 721–728.

Tseng, H., Chang, P.-C., Andrew, G., Jurafsky, D., and Manning, C. (2005). A conditional random field word segmenter. In *Fourth SIGHAN Workshop on Chinese Language Processing*.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Wainwright, M. J. and Jordan, M. I. (2008). *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA.

Watanabe, T. (2012). Optimized online rank learning for machine translation. In *Proceedings of NAACL*.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic. Association for Computational Linguistics.

Weaver, W. (1949/1955). Translation. In Locke, W. N. and Boothe, A. D., editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA. Reprinted from a memorandum written by Weaver in 1949.

White, J., O'Connell, T., and L. (1993). Evaluation of machine translation. In *Human Language Technology:Proceedings of the Workshop ARPA*.

Wu, D. (1996). A polynomial-time algorithm for statistical machine translation. In *In 34th Annual Meeting of the Association for Computational Linguistics*, pages 152–158.

Xiao, X., Xiong, D., Zhang, M., Liu, Q., and Lin, S. (2012). A topic similarity model for hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 750–758, Jeju Island, Korea. Association for Computational Linguistics.

Xiong, D., Zhang, M., and Li, H. (2012). Modeling the translation of predicate-argument structure for smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yu, C.-N. J. and Joachims, T. (2009). Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1169–1176, New York, NY, USA. ACM.

Yu, H., Huang, L., Mi, H., and Zhao, K. (2013). Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA.

Yuille, A. L. and Rangarajan, A. (2003). The concave-convex procedure. *Neural Comput.*, 15(4):915–936.

Zaidan, O. F. and Callison-Burch, C. (2010). Predicting human-targeted translation edit rate via untrained human annotators. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 369–372, Los Angeles, California. Association for Computational Linguistics.

Zhang, T. (2003). Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–134.

Zhao, B. and Xing, E. P. (2006). Bitam: Bilingual topic admixture models for word alignment. In *In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 138–141.