Optimizing Machine Translation by Learning to Search

by

Daniel Galron

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2012

I. Dan Melamed

Acknowledgements

I could not have done this work without the help and encouragement of many other people.

First, I would like to thank my advisor, I. Dan Melamed for his advice, support, patience, and kindness. He has shaped how I approach problem solving and research. He has taught me how to be a scientist, which is something for which I will be eternally grateful.

I would also like to thank Srinivas Bangalore, Ralph Grishman, Mehryar Mohri, and Satoshi Sekine for serving on my dissertation committee, suggestions for strengthening this dissertation, and for the pointers to the literature and other fruitful discussions on this work.

Additionally, thank you to Anca Brates-Galron and Sarah Witman for their help in manually evaluating the translation outputs of the systems presented in this work.

I would also like to thank Sergio Penkale, Andy Way, and the Centre for Next Generation Localisation for the fruitful research we did during my stay in Ireland, Srinivas Bangalore for the work we did at AT&T Research, and Pascal Fleury for the great internship at Google.

On a personal note, I'm so grateful for the kindness, love, support, and encouragement of my family and friends. My deepest thanks and love to my parents for their words of encouragement and advice. To Clair, I couldn't have done this without your love, support, patience, and rational reassurances.

Finally, I would like to thank the coffee shops around NYU for the copious amounts of espresso needed to complete this work.

Abstract

We present a novel approach to training discriminative tree-structured machine translation systems by learning to search. We describe three primary innovations in this work: a new parsing coordinator architecture and algorithms to generate the required training examples for the learning algorithm; a new semiring that provides an unbiased way to compare translations; and a new training objective that measures whether a translation inference improves the quality of a translation. We also apply the reinforcement learning concept of exploration to SMT. Finally, we empirically evaluate our innovations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last several decades, statistical approaches have come to the forefront in the natural language processing research community. Statistical approaches have been especially beneficial to automatic translation of natural language. In statistical machine translation (SMT) an automatic translation system uses an automatically learned model to choose between candidate translations.

In many NLP tasks, the size of the output space, that is, the number of possible outputs for an input sentence is polynomial or exponential in the size of the input. For finite-state SMT the size of the output space is exponential, since the translation of each source phrase can be placed anywhere in the target string. When the placement of translations is not constrained, the set of possible outputs contains all permutations of phrase translations. For tree-structured SMT, the size of the output space is a high-order polynomial of the size of the input, as discussed in [Melamed and Wang2004].

Because the size of the output space is prohibitively large, all approaches to SMT must infer outputs incrementally by constructing the structured labels (i.e. possible translations of an input sentence) from their constituent substructures (i.e. possible translations of parts of the input sentence). The process of constructing outputs can be viewed as a search in a weighted hypergraph [Klein and Manning2001]. Each hyperedge in the hypergraph represents an *inference* taken by the search algorithm to infer a larger structure from constituent substructures. Translations are incrementally inferred by the search algorithm as it traverses the hyperedges in the hypergraph.

Most current state-of-the-art approaches to SMT use machine learning to learn how to traverse the hypergraph. A commonly held view is that to apply machine learning techniques to solve a problem well, one must train a model according to an objective function that is well-correlated with the quality of the output. If the quality can be quantified, ideally the objective measure will be exactly the quantified measure of output quality.

In current state-of-the-art approaches to SMT, the translator's search procedure is guided by a model with a very large number of parameters. However, very few of them are selected (i.e. optimized) for the search task on the hypergraph. For example, in generative models (e.g. [Brown et al.1993, Och et al.1999, Alshawi et al.2000, Yamada and Knight2001]) the parameters are all estimated according to maximum likelihood estimates (MLE) of the features they weight. In mildly discriminative models, most weights are also estimated according to MLE, but these weights are multiplied by a handful of other model weights[1] that are estimated according to minimum error rate training [Och2003] or maximum entropy [Och and Ney2004]. Discriminative approaches such as MIRA [Watanabe et al.2007, Chiang et al.2008] estimate more weights discriminatively than mildly discriminative approaches, but still have a very large number of weights trained by MLE. For other discriminative approaches, for instance [Wellington et al.2006a], the weights are chosen by optimizing proxy objectives that receive no feedback from, nor are otherwise informed by the search procedure. Such models are unlikely to be optimal for the search process.

Although [Liang et al.2006, Tillmann and Zhang2006] have presented fully discriminative finite-state approaches to SMT, recent research, such as [Chiang2005, Galley et al.2006, Chiang et al.2008, Mi et al.2008] has shown that tree-structured systems can better represent translational equivalence patterns and generate better translations than finite-state approaches. Our goal is to develop fully discriminative methods for tree-structured SMT utilizing millions of fine-grained features for prediction to maximize the predictive capability of the model selected by the training procedure. Furthermore, we would like all the features to be parameterized using a training objective that is well-correlated with translation accuracy.

Most current approaches to SMT training attempt to model the quality of whole translations, thereby teaching the translator a preference between complete translations. A popular method for discriminative training in the SMT literature is by reranking. In reranking approaches, the translator generates an n-best list of translations for an input sentence according to the preferences encoded by the current model. Depending on the algorithm, the weights of the model are adjusted so that either: 1) the translations in the n-best list are ordered according to decreasing accuracy, where the lowest-cost translation has the highest accuracy; or 2) that the highest-accuracy translation in the n-best list is selected as a positive example, and all others as negative examples, and the model is trained by maximizing the margin between the positive and negative examples. One of the advantages of reranking is that the model can be iteratively improved by having it learn to discriminate between low-cost predictions made under the current model and a low-cost prediction that has the highest accuracy, an idea related to maximizing the smallest margin between the candidate translations and the decision

[1] Usually 15-30 parameters

boundary. We suspect, however, that training in this manner may be sub-optimal for two primary reasons:

1. The process by which the translations are constructed is not being learned. The translator's search procedure is being guided by the model's preferences among whole translations, which might not correctly choose between partial translations proposed by the translator.

2. To train millions of features, we need a sufficient number of training examples. To achieve good prediction accuracy, one should have many more training examples than parameters. Since training examples contain features on the input and the structured output, enumerating enough of the output space to generate enough training examples is computationally infeasible.

In this work, we've decided to take a different approach. The approach we utilize learns the inference procedure the translator should use at test time by learning to predict whether the translator should fire a given inference, or, equivalently, whether it should traverse a given hyperedge in the search hypergraph. Doing so directly addresses the two issues above:

1. By learning to predict inferences, the search procedure is guided by a measure of the utility of each inference in outputting a good translation, thereby training it for the task actually undertaken during test-time.

2. By learning to predict inferences, we can utilize more training examples because we do not need to infer whole translation candidates to use for training the model. We can utilize inferences that are not expanded to infer full translations, thereby saving us some computational cost in constructing a large-enough training set.

By learning the search procedure, we're not only solving a more general problem, but we're actually learning to directly model the method the translator uses to construct translations. A potential argument against our approach is summarized by Vladimir Vapnik's comment: "When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one" [Vapnik1995]. Unfortunately, the only way we know how to solve the translation problem is through hypergraph search. By utilizing our proposed training method, we are directly solving the problem of how to infer good translations by learning which inferences the translator should make to infer them.

To achieve our research objectives, we had to solve many problems – some theoretical, some empirical, and some architectural. In this dissertation, we present solutions for some of these problems. Here we give a brief overview of some of our contributions to research on SMT.

Synthesizing training data The ideal training set for learning how to search the hypergraph would consist of inferences with indicators as to which inference is the best to make from each possible node. We know of no such dataset. The best approximation we know of are parallel treebanks, but they tend to be too small for SMT and they exist only for a handful of language pairs (e.g. [Ahrenberg2007, Megyesi et al.2008]). We therefore had to rely on automatically sentence-aligned bitexts, which do not specify the optimal sequence of inferences. Thus, in addition to learning to search for a translation for a given input sentence, the training algorithms we apply must guess the optimal search path through the hypergraph, even given a reference translation.

To facilitate controlled experimentation, we present a novel *parsing coordinator* architecture that allows us to control both the search procedure undertaken by the translator and the space in which it searches. The coordinator contains an "expert" *supervisor* that limits the space of allowed derivations for each sentence pair. The supervisor is constrained to infer the reference translation. The coordinator also contains a *predictor* translator that is not constrained to infer the reference. The coordinator consults with the supervisor to determine whether the inferences fired by the predictor are on a path towards the reference translation. The supervisor's determination is given to an *example generator* that extracts a feature vector from the inference and assigns a label to the example indicating its utility in finding the reference. The coordinator can also guide and restrict the search procedure of the predictor by telling it which inference it should take next, or by telling the predictor that it should not make a particular inference. Using the generalized parsing paradigm [Melamed and Wang2004] and the GenPar toolkit [Burbank et al.2005] facilitated building the coordinator and allowed us to ensure that the predictor and supervisor are searching the same hypergraph under the same constraints, except for those that we wish to vary. An additional contribution is the algorithm utilized by the parsing coordinator. It is efficient in the sense that it doesn't do any more work than is necessary for learning at any given time – it lazily generates only the training examples needed to represent the inferences we wish to learn from. The parsing coordinator architecture can be applied to other NLP problems as well, including parsing and finite-state SMT.

Training objectives To learn at the inference level, we need a suitable objective function. Current objective functions that are utilized in the research literature are not well suited for this task. The maximum likelihood objective function of generative approaches is not well correlated with the search procedure. The minimum error-rate objective [Och2003], max-margin objective [Watanabe et al.2007,Chiang et al.2008], and cost-sensitive margin objective [Tillmann and Zhang2006] do not decompose to be applicable to the inference level. The discriminative approach of [Wellington et al.2006a] does decompose to the inference level, but it tries to

learn a single multitree over each sentence pair in the bitext. Doing so restricts the set of legal multitrees their translator attempts to learn and may make it harder to learn consistent tree structures if similar sentence pairs have dissimilar tree structures. So we describe a novel class of objective functions, the whole inference quality (WIQ) criterion, which measures the quality of an item's yield relative to the quality of the yields of its antecedent items. We also experiment with *exploration*, in the sense used in reinforcement learning. We are not aware of any other SMT methods that control exploration in the same deliberate manner as we do. Our experiments in Chapter 5 demonstrate the usefulness both of the WIQ objective and of controlling the amount of exploration.

Comparing inference sets Having the translator make each inference in a locally-optimal manner does not guarantee that the entire structure that the translator infers will be globally optimal. One of the challenges in finding the optimal inference set is that the traditional method of comparing sets of inferences by summing over their costs is biased against larger sets. Furthermore, in summing over the costs, we lose information about which inferences were good and which were bad, information that can help us compare the quality of inference sets. We expect that having this information is beneficial since it would allow us to determine which inferences are responsible for mistakes in the translation, which the sum of costs does not reflect. To address this problem we invented an unbiased way to compare inference sequences. Our primary innovation to address this problem is a new semiring, which we call the MinMerge semiring. The domain of the MinMerge semiring is sorted sequences of real numbers, representing each partially-ordered set (poset) of inferences. The semiring is not limited to tree-structured SMT. It can be applied to any search problem. The lesioning experiment presented in Chapter 5 show that the MinMerge semiring gives better translation results than the Viterbi semiring for our model and training method.

The primary disadvantage of our approach is that it currently does not scale up very well. However, this work was a necessary step in learning how to build a system that learns a model according to an objective for the search procedure the translator undertakes. We describe such a system in this work and show that learning to search is beneficial for SMT. Since we think that our system can be scaled up to learn from corpora of millions of sentence pairs, as we will discuss in Chapter 6, we believe that our work is a valuable contribution to the field.

While the focus of this dissertation is on statistical machine translation, we conjecture that many of the techniques we have innovated for SMT are applicable to a wide range of other structured prediction problems. Many structured prediction problems, both in natural language processing and in other subfields of artificial

intelligence, utilize an incremental search procedure for constructing outputs. We believe that the example generation framework, the concepts behind our objective functions, and the MinMerge semiring may be applicable to NLP problems such as parsing and named entity recognition, as well as non-NLP problems in computer vision and robotics.

The dissertation is structured as follows. We first discuss the relative advantages and disadvantages of other approaches to SMT in Chapter 2. In Chapter 3 we present a high-level overview of our training algorithm and our example generation framework in terms of a generic translation algorithm. In Chapter 4 we present the details of the translation algorithms employed by the training algorithm and example generator. Finally, we experiment with our innovations and present and analyze the empirical results in Chapter 5. Finally, we discuss future work and other applications of our approach in Chapter 6.

Chapter 2

Literature Review

2.1 Correspondence Structures

Recall that most approaches to MT search for a hidden correspondence structure between an input sentence and a translation candidate that represents translational equivalence between the input and translation. There are three prominent formulations of the hidden correspondence structure in the research literature, which are distinguished by the types of translational equivalence patterns they can represent. While the distinction is somewhat orthogonal to the focus of this dissertation, it is important to note the differences, as the hidden structure types can have a great effect on how the model can be parameterized.

Finite-state word models Finite-state word models use words as the basic unit of translation. They are frequently referred to as "word-based models" in the research literature (e.g. in [Koehn and Hoang2007]). Finite-state word translation systems tend to view the translation problem as bag-of-words translation, where: 1) each word in an input sentence is translated into a target-language word (possibly an $\varnothing$ token indicating a word deletion); 2) words are possibly inserted on the target side; and 3) the target-language words are reordered to adhere to target-language word order rules. Examples of these approaches include [Brown et al.1988, Brown et al.1993, Vogel et al.1996, Tillmann et al.1997, Och and Weber1998]. Typically, finite-state word models are comprised of several sub-models, including a lexical translation model for the words in the input sentence, a language model and a word distortion model for determining the target-language word order, and some form of word insertion model. Most finite-state word translation algorithms make strong independence assumptions in their models, for example, by assuming that all source words are independently translated of one another (as in IBM model 1 of [Brown et al.1993]), or by assuming that the position of each target word depends only on the position of the target translation of the previous source word

(as in [Tillmann et al.1997]). The hidden correspondence structure for finite-state word translation approaches tends to be a word alignment, which is a partial bipartite matching between the words in the source sentence and the words in the target sentence.

Finite-state phrase models Rather than assuming each word is independently translated from every other word, finite-state phrase translation systems model the translation of word sequences, which they call "phrases", thereby capturing some inter-dependencies in how adjacent words are translated. Finite-state phrase approaches to SMT often work as follows: they first segment the input sentence into phrases of varying lengths, translate each source phrase, and re-order the phrase translations to conform with the target-language word order model. Examples of these include [Och et al.1999, Koehn et al.2003, Och and Ney2004, Koehn and Hoang2007]. The translational equivalence patterns expressed by the models tend to be phrase pairs. The hidden correspondence structure still tends to be a word alignment (as in the finite-state word approach), but where various heuristic or statistical methods are used to estimate a mapping over phrase pairs. Finite-state phrase translation approaches tend to yield higher-accuracy translations than finite-state word translation approaches because rather than translating and ordering words independently, by utilizing (possibly) multi-word phrases as the atomic unit of translation, the translator essentially memorizes how adjacent source words jointly translate into locally adjacent target words. Because the resulting phrases are atomic, there is a hard-coded dependency between the words within each source phrase and each target phrase. This structural dependency relaxes the independence assumptions made in finite-state word models and allows the model to memorize local target word reordering patterns.

The length of the phrase pairs determine how much context is captured by the phrases. Short phrases only capture the immediate context for the words in the phrases. To model long-distance dependencies and re-orderings, one would need to extract long phrases from the training bitext. However, there are fewer occurrences of the longer phrases than the shorter ones, so the model estimates over the longer phrase-pairs may be unreliable for rare phrases. Finite-state phrase SMT approaches tend to be very good at capturing local translational equivalence and reordering patterns, but not very good at modeling long-distance dependencies. For example, if one is translating from a Subject-Verb-Object (SVO) word-order language such as English to a SOV language such as Hindi, it tends to be difficult for finite-state models to invert the object noun phrase and verb phrase when translating.

Tree-structured models Tree-structured models use synchronous parse trees or multitrees [Wu1997, Melamed2004] to represent the hierarchical structure of a

sentence pair and to capture long-distance re-orderings and dependencies. Hierarchical models can be learned from an (automatically or manually) annotated treebank, for instance, [Galley et al.2006, Liu et al.2006, Mi et al.2008], or can have no relation to any linguistic notion of constituency, but merely represent a hierarchical nesting of phrases, for instance the approach of [Chiang2005]. The hidden correspondence structure in tree-structured models tend to be constrained by synchronous grammars that model the translational equivalences between the source and target languages.

2.2 Generative models

The first approaches to SMT parameterization were generative. Generative models present a "story" for how a set of observed data was generated by a stochastic process parameterized by a set of probability distributions. Most generative systems followed the so-called *noisy-channel model* or *source-channel model*, that attempted to model the probability of a target-language translation $\mathbf{t}$ for a given source-language sentence $\mathbf{s}$ by applying Bayes' rule:

$$\Pr(\mathbf{t}|\mathbf{s}) = \frac{\Pr(\mathbf{s}|\mathbf{t})\Pr(\mathbf{t})}{\Pr(\mathbf{s})}$$

These systems typically apply a decoder to search for the translation that maximizes the following:

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \Pr(\mathbf{t}|\mathbf{s}) = \arg\max_{\mathbf{t}} \Pr(\mathbf{s}|\mathbf{t})\Pr(\mathbf{t})$$

Since the denominator $\Pr(\mathbf{s})$ is constant for a given input sentence $\mathbf{s}$, it has no effect on the maximization. The $\Pr(\mathbf{t})$ term represents the *language model*, which gives the likelihood that a translation $\mathbf{t}$ is a valid sentence in the target language, and the $\Pr(\mathbf{s}|\mathbf{t})$ term represents the *translation model* that models the likelihood that sentence $\mathbf{s}$ has been generated from $\mathbf{t}$. Modeling $\Pr(\mathbf{s}|\mathbf{t})$ directly is intractable. Most systems that follow this approach decompose the translation model into smaller tractable sub-models which can then be estimated, usually by making very strong independence assumptions about the translation process (e.g. [Brown et al.1993, Och et al.1999, Yamada and Knight2001]). The estimation is often done with the aid of hidden correspondence variables modeling translational equivalence patterns.

2.2.1 Finite-state generative word models

The finite-state word models of [Brown et al.1993] are commonly viewed as the first statistical approaches to SMT. They present five translation models of increasing complexity and decreasing number of independence assumptions. The first

model, **IBM Model 1**, assumes that every target-language word in a target sentence of length m is generated independently as a mixture of the source-language words in a sentence of length l given a set of hidden correspondence structures consisting of possible alignments a_k: $\Pr(\mathbf{s}|\mathbf{t}) \sim \sum_{a_1=0}^{l} \ldots \sum_{a_m=0}^{l} \prod_{j=1}^{m} \Pr(\mathbf{s}_j|\mathbf{t}_{a_j})$. This model assumes that all possible alignments $a_1 \ldots a_m$ are equally likely. **IBM Model 2** removes this assumption by introducing a probability distribution over alignments: $\Pr(\mathbf{s}|\mathbf{t}) \sim \sum_{a_1=0}^{l} \ldots \sum_{a_m=0}^{l} \prod_{j=1}^{m} \Pr(\mathbf{s}_j|\mathbf{t}_{a_j}) \Pr(a_j|j,m,l)$. **IBM Model 3** decomposes $\Pr(\mathbf{s}|\mathbf{t})$ into three models: a *fertility model* that models the probability that source word $\mathbf{s}_i$ generates a given number of target words[1]; a lexical translation model; and a distortion model which models the probability of a permutation of target-side word order into source-side word order. **IBM Model 4** further decomposes the distortion model of Model 3 into two sub-models, to better capture the fact that word sequences tend to move together (i.e. as phrases). Models 3 and 4, however, are deficient in that they assign probability mass to structurally impossible events (e.g. two source words being moved into the same target position). **IBM Model 5** is a non-deficient version of Model 4. All the distributions in the IBM models are estimated via the Expectation-Maximization algorithm [Dempster et al.1977] using the maximum approximation, where rather than summing over all possible alignments for a sentence they take the most likely one.

[Tillmann et al.1997] follow [Vogel et al.1996] in formulating the translation model as a Hidden Markov Model (HMM), and assume that all word-to-word alignments are one-to-one. They decompose the translation model into two sub-models: the lexicon model modeling the lexical translation probability $\Pr(\mathbf{s}_j|\mathbf{t}_{a_j})$ and the alignment model modeling the conditional probability of a word-to-word alignment a_j given the preceding alignment a_{j-1}:

$$\Pr(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \prod_{j=1}^{J} \Pr(a_j|a_{j-1}) \Pr(\mathbf{s}_j|\mathbf{t}_{a_j})$$

This formulation corresponds to an HMM where the transition probabilities are given by the $\Pr(a_j|a_{j-1})$ term and the emission probabilities are given by the $\Pr(\mathbf{s}_j|\mathbf{t}_{a_j})$ terms. To make their model tractable, they make the assumption that the alignments are mostly monotonic, which allows them to make a Markov assumption over alignments.

The translation approach described in [Och and Weber1998] is a combination of the statistical approach of [Brown et al.1993] with an example-based approach to machine translation. Their approach consists of a statistical translation lexicon model (a variant of IBM Model 2, where instead of an absolute position model,

[1]Thus, the distribution of source-language words is now a mixture conditioned on a subset of words in $\mathbf{t}$.

they use a relative one, as in [Vogel et al.1996]); a word-class model that assigns each source word to a class; and a set of translation rules. A translation rule is a triple $(\mathbf{S}, \mathbf{T}, \mathbf{Z})$ where $\mathbf{S}$ is the sequence of source-language word classes for a sentence, $\mathbf{T}$ is the sequence of target-language word classes, and $\mathbf{Z}$ is an alignment matrix between the two (where the alignments are induced using their variation of IBM model 2). Their model is given by:

$$\Pr(\mathbf{t}|\mathbf{s}) = \sum_{\mathbf{S},\mathbf{Z}} \Pr(\mathbf{T}, \mathbf{Z}|\mathbf{s}) \cdot \Pr(\mathbf{t}|\mathbf{T}, \mathbf{Z}, \mathbf{s})$$

In practice, rather than taking the sum they take the max (i.e. the maximum approximation) and approximate $\Pr(\mathbf{t}|\mathbf{T}, \mathbf{Z}, \mathbf{s})$ as the product of a language model probability, a translation model probability, and a word class probability:

$$\Pr(\mathbf{t}|\mathbf{T}, \mathbf{Z}, \mathbf{s}) \approx p(t_j|t_1^{j-1}) \cdot p(t_j|\mathbf{s}, \mathbf{Z}) \cdot p(t_j|T_j)$$

These models are estimated with the relative frequency maximum-likelihood estimator.

2.2.2 Finite-state maximum likelihood phrase models

[Och et al.1999] observe that finite-state word models fail to capture dependencies between groups of words when translating, meaning that, for example, translating compound nouns is very difficult for this model. They provide a new alignment model which takes into account both dependencies between single words, and dependencies between phrase structures. Their approach follows that of [Och and Weber1998] in that they utilize alignment templates (defined in much the same way as Och and Weber's translation rules). An alignment template is a tuple $z = (\tilde{S}, \tilde{T}, \tilde{A})$ where $\tilde{S}$ is a source word class sequence, $\tilde{T}$ is a target word class sequence, and $\tilde{A}$ is the alignment matrix between the two word sequences. In order for an alignment template to match a subsequence of the source sentence, the classes of all the words in the subsequence of the input must match $\tilde{S}$. Their "generative story" then is: 1) A segmentation model segments the source sentence into phrases; 2) A phrase alignment model generates the positions at which to fill in the target phrases relative to the source phrases; 3) the lexical alignment model determines where to generate each target word in each phrase relative to the source words; 4) the lexical translation model determines which translation to generate for each word in each phrase. Using this method, their model can capture local dependencies between words, making this model better suited for translating between syntactically-similar languages. However, their method still suffers from not being able to handle non-monotonicity very well. Their underlying approach assumes that the alignment is monotonic with respect to the word order for most word alignments.

[Marcu and Wong2002] present a generative joint probability model that models word and phrase equivalency. They assume that each sentence in their training bitext is produced by the following stochastic process:

1. Generate a bag of *concepts* C
2. Initialize T and S to be the empty sentences ϵ
3. Randomly select a concept $c_i \in C$ and generate a phrase pair (t_i, s_i) according to the distribution $p_t(t_i, s_i)$. Remove c_i from C
4. Append phrase t_i at the end of T. Let k be the start position of t_i in T.
5. Insert phrase s_i at position l in S if there is no phrase occupying any positions between l and $|s_i|$. Then, the probability of the alignment between s_i and t_i is given by: $\prod_{j=k}^{k+|s_i|} p_d(j, (l + |t_i|)/2)$ where $p_d(\cdot, \cdot)$ is the distortion model.
6. Repeat steps 3 to 5 until $C = \varnothing$

The joint probability of the sentence pair $(\mathbf{s}, \mathbf{t})$ is given by:

$$p(T, S) = \sum_{C \in \mathcal{C}: L(T,S,C)} \prod_{c_i \in C} \left[p_t(t_i, s_i) \prod_{k=1}^{|s_i|} p_d(pos(s_i^k), pos(|t_i|)/2) \right]$$

where $pos(s_i^k)$ is the position of the kth word of phrase s_i, and $pos(|t_i|/2)$ is the position of the center word of phrase t_i. The two models $p_t(\cdot, \cdot)$ and $p_d(\cdot, \cdot)$ are estimated via EM. First, they determine the high-frequency n-grams in the bitext. They estimate an initial model p_t from the n-grams that meet a pre-specified frequency threshold and from unigrams. They then apply EM on the Viterbi alignment for each sentence pair to estimate $p_t(\cdot, \cdot)$ and $p_d(\cdot, \cdot)$. Finally, to apply this model, they marginalize over t_i to compute the conditional distributions $p_t(\cdot|\cdot)$ and $p_d(\cdot|\cdot)$.

[Koehn et al.2003] present a noisy-channel finite-state phrase model composed of four sub-models:

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} p(\mathbf{t}|\mathbf{s}) = \arg\max_{\mathbf{t}} p(\mathbf{s}|\mathbf{t}) \times p_{LM}(\mathbf{t}) \times \omega^{|\mathbf{t}|}$$

$$p(\mathbf{s}|\mathbf{t}) = \prod_{i=1}^{I} \phi(s_i|t_i) d(a_i - b_{i-1})$$

where ω is a length penalty that penalizes shorter translations, p_{LM} is the language model, ϕ is the phrase translation probability model, and d is the distortion model. They use a variety of heuristics to extract phrases from an automatically word-aligned bitext, and estimate the models using relative frequencies.

[Tillmann2003] presents a generative block model for machine translation. A block is a pair consisting of contiguous source- and target-side spans. They present an algorithm for generating the blocks from the intersection of bidirectional HMM alignments, and expanding them using the union of the alignments. The model computes the probability of a block sequence covering the source sentence: $\Pr(b_1^n) \approx \prod_{i=1}^{n} \Pr(b_i|b_{i-1}) = \prod_{i=1}^{n} \left[p^{\alpha}(b_i) \cdot p^{(1-\alpha)}(b_i|b_{i-1})\right]$ where α is a hyperparameter that controls the weights in the mixture. Like [Marcu and Wong2002], [Tillmann2003]'s model is a joint model. The probabilities are estimated via relative frequencies.

2.2.3 Tree-structured maximum likelihood models

Maximum likelihood models have been applied to tree-structured SMT approaches as well. [Wu and Wong1998] present a grammatical channel model for SMT using stochastic inversion transduction grammar (SITG) [Wu1997]. They seek the translation $\hat{\mathbf{t}}$ that maximizes $\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \Pr(\mathbf{s}, \mathbf{t}, \mathbf{q}) \Pr(\mathbf{t})$ where $\mathbf{q}$ is a synchronous parse tree over the sentence pair $\mathbf{s}, \mathbf{t}$. They produce an ITG from a monolingual target side treebank by *mirroring*: for every target-language CFG production, they create two ITG productions – one whose RHS is monotonic with respect to the original order, and one whose RHS is inverted. They map part-of-speech tags between the source and target languages using a translation lexicon. In their experiment, they constructed their lexicon by aligning a bitext via EM, and extracting the highest-likelihood alignments as lexicon entries. The probabilities of the ITG productions are then estimated via EM.

[Alshawi et al.1998, Alshawi et al.2000] present a finite state translation model using collections of *head transducers*, which can be thought of as a synchronous dependency grammar. The role of each transducer is to translate a specific source word w into a particular target word, and to recursively translate the dependents of the head word w. Unlike the finite-state word or finite-state phrase methods, their approach generates translations outwards from the head, rather than left-to-right. A head transducer for translating sentence $\mathbf{s}$ to $\mathbf{t}$ in language $\mathbf{T}$ consists of a set of states $Q(w : v)$ for each $w \in \mathbf{s}$ and $v \in \mathbf{T}$, and transitions between states are given by $(q_i(w : v), q_j(w : v), w_d, v_d, \alpha, \beta)$, where upon transitioning from q_i to q_j, reading a source dependent word w_d at relative position α to the head w, it generates translation v_d for w_d at the relative position β to v. The transitions are parameterized by probability $\Pr(q_j(w : v), w_d, v_d, \alpha, \beta | q_i(w : v))$, which is estimated via maximum likelihood estimation by counting the head transduction derivations on a bitext. This method has the advantage of capturing the nested structure of natural language while still being a finite-state model.

[Yamada and Knight2001] apply the noisy channel model to tree transduction. They assume that a source-language parse tree is fed into a noisy-channel model

and is translated into a target-language sentence. Their method has several steps. First, the nodes of the source parse tree are stochastically re-ordered. Then, an extra target-language word is stochastically inserted at each node. Finally, each source-language leaf is translated into a target-language word. They define and parameterize a stochastic model for each operation. The probability of getting a target-language sentence $\mathbf{t}$ given source-language tree $\mathcal{S}$ is:

$$\Pr(\mathbf{t}|\mathcal{S}) = \sum_{\theta:\text{yield}(\theta)=\mathbf{t}} \Pr(\theta|\mathcal{S})$$

$$\Pr(\theta|\mathcal{S}) = \prod_{i=1}^{n} \Pr(\theta_i|\theta_1, \theta_2, \ldots, \theta_{i-1}, \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n) \approx \prod_{i=1}^{n} \Pr(\theta_i|\mathcal{S}_i)$$

$$\Pr(\theta_i|\mathcal{S}_i) = p(\nu_i|\mathcal{N}(\mathcal{S}_i))p(\rho_i|\mathcal{R}(\mathcal{S}_i))p(\tau_i|\mathcal{T}(\mathcal{S}_i))$$

where each $\mathcal{S}_i$ is a node in tree $\mathcal{S}$, ν_i, ρ_i, τ_i represent an insertion, reorder, or translation on node $\mathcal{S}_i$ respectively, and $\mathcal{N}, \mathcal{R}, \mathcal{T}$ are the respective feature functions over the node. The parameters are estimated via EM.

2.3 Mildly discriminative models

The source-channel model approach to SMT has a few disadvantages. First, the objective the generative models optimize is not necessarily well-correlated with the measure by which the translations are evaluated. Furthermore, it is not clear that all the sub-models should have equal weights. For instance, in the source-channel model the distortion model and the language model contribute the same amount to the probability of translation. It may be that the language model gives more reliable estimates as to target-language word positions than the distortion model, due to data sparsity. Finally, the source-channel model allows us to use only probabilistic sub-models. If there is some information that we wish to incorporate in the model that does not have a probabilistic interpretation (e.g. a bias against shorter translations), we cannot do so in the probabilistic source-channel model.

To overcome some of the problems with the source-channel model, researchers have shifted towards *mildly discriminative* models. Like the source-channel model, mildly discriminative models model the probability of the target sentence given the source: $\Pr(\mathbf{t}|\mathbf{s})$. However, unlike the source-channel model, mildly discriminative models directly model the posterior probability $\Pr(\mathbf{t}|\mathbf{s})$ by defining a set of feature functions $h_m(\mathbf{s}, \mathbf{t})$, and parameterizing each feature function by parameter λ_m. The posterior probability is then given by:

$$\Pr(\mathbf{t}|\mathbf{s}) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{s}, \mathbf{t})]}{\sum_{\mathbf{t}'} \exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{s}, \mathbf{t}')]}$$

The translator searches for the translation that maximizes:

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(\mathbf{s}, \mathbf{t}) \right\}$$

Typically, the feature functions range over a hidden correspondence structure $\mathbf{a}$ as well:

$$\Pr(\mathbf{t}, \mathbf{a}|\mathbf{s}) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{s}, \mathbf{t}, \mathbf{a})]}{\sum_{\mathbf{t}', \mathbf{a}'} \exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{s}, \mathbf{t}', \mathbf{a}')]}$$

Mildly discriminative approaches differ in their choice of feature functions h_m and in their methods of selecting parameters λ_m. Two promising methods for tuning the weights λ_m are Minimum Error Rate Training (MERT), which uses a measure of translation accuracy as the objective [Och2003]; and maximizing the mutual information criterion, derived from the maximum entropy principle [Och and Ney2004].

The idea of MERT [Och2003] is to find the values of λ_m that lead to the lowest translation error rate on some tuning set. For a given tuning set of sentences S, let $\mathbf{r}_1^S$ denote a set of reference translations $\mathbf{r}_s$ for each sentence $s \in S$, and let $\mathbf{T}_s = \{\mathbf{t}_{s,1}, \ldots, \mathbf{t}_{s,K}\}$ be the set of candidate translations output by the translation system for sentence s. Finally, let $E(\mathbf{r}_1^S, \mathbf{t}_1^S) = \sum_{s=1}^{S} E(\mathbf{r}_s, \mathbf{t}_s)$ be the translation accuracy metric comparing the candidate translation to the reference. MERT minimizes the following objective:

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^{S} \sum_{k=1}^{K} E(\mathbf{r}_s, \mathbf{t}_{s,k}) \delta(\hat{\mathbf{t}}(\mathbf{s}_s; \lambda_1^M), \mathbf{t}_{s,k}) \right\}$$

with

$$\hat{\mathbf{t}}(\mathbf{s}_s; \lambda_1^M) = \arg\max_{\mathbf{t} \in \mathbf{T}_s} \left\{ \sum_{m=1}^{M} \lambda_m h_m(\mathbf{t}|\mathbf{s}_s) \right\}$$

This objective function finds the parameters λ_1^M that re-rank a top K list of translation candidates so that the lowest-cost candidates have minimum error. [Och2003] presents a grid search algorithm based on Powell's method for finding a set of optimal λ_m. Given that the objective is non-convex, it is not guaranteed that the algorithm will find the global optimum. Furthermore, it has been found that MERT can reliably scale to no more than 15-30 parameters [Chiang et al.2008].

[Och and Ney2004] utilize a maximum-entropy approach [Berger et al.1996] to find the parameters λ_1^M. They apply the finite-state alignment template approach of [Och et al.1999] to a mildly discriminative model. Their system generates alignments using the HMM method of [Vogel et al.1996] and the IBM Model 4 of [Brown

et al.1993]. They symmetrize the alignments by taking the intersection of the directional alignments in both directions and heuristically augmenting them with alignments from the union. Phrases that are consistent with the alignments are then extracted from the training corpus, and alignment templates (e.g. [Och et al.1999]) are extracted from the phrases. They then use the following feature functions:

- The alignment template probability given a source phrase $s_{j_{\pi_k-1}+1}^{j_{\pi_k}}$: $\log \prod_{k=1}^{K} p(z_k | s_{j_{\pi_k-1}+1}^{j_{\pi_k}})$
- The lexical translation probability: $\log \prod_{i=1}^{I} p(t_i | \{f_j : (i,j) \in A\}, S_i)$
- A measure of the number of crossing alignment templates[2]: $\sum_{k=1}^{K+1} |j_{\pi_k - 1} - j_{\pi_{k-1}}|$
- A trigram language model and 5-gram class language model: $\log \prod_{i=1}^{I+1} p(t_i | t_{i-2}, t_{i-1})$ and $\log \prod_{i=1}^{I+1} p(C(t_i) | C(t_{i-4}), \ldots, C(t_{i-1}))$
- A penalty that penalizes translations that are too short
- The number of occurrences of bilingual lexicon entries in the given sentence pair.

They then estimate the λ_m weights by maximizing the class-posterior probability using Generalized Iterative Scaling:

$$\hat{\lambda}_1^M = \arg\max_{\lambda_1^M} \left\{ \sum_{i=1}^{I} \log p_{\lambda_1^M}(\mathbf{t}_i | \mathbf{s}_i) \right\}$$

Unlike MERT, this objective has the advantage of being convex, so a global optimum will be found by the training procedure.

[Blunsom et al.2008] present a method similar to that of [Och and Ney2004]: a global mildly discriminative model that defines a conditional probability distribution over the translation given the source sentence. They introduce a latent variable modeling the derivations of the translation, and define the conditional probability as:

$$p_\Theta(\mathbf{d}, \mathbf{t} | \mathbf{s}) = \frac{\exp \sum_k \Theta_k \mathbf{h}_k(\mathbf{d}, \mathbf{t}, \mathbf{s})}{Z_\Theta(\mathbf{t})}$$

[2]Because they focus on similar European languages, they expect the alignments to be almost completely monotonic.

where $\mathbf{h}_k(\mathbf{d}, \mathbf{t}, \mathbf{s})$ is the k'th feature, and $Z_\Theta(\mathbf{t})$ is a normalizer. They then marginalize over derivations $\mathbf{d}$. To learn the parameters, they minimize the log-likelihood objective using a maximum a posteriori (MAP) estimator:

$$\sum_{(\mathbf{s},\mathbf{t})\in\mathcal{D}} \log p_\Theta(\mathbf{t}|\mathbf{s}) + \sum_k \log p_0(\Theta_k)$$

where $p_0(\Theta_k)$ is a Gaussian prior. Their approach effectively places an ℓ_2 regularizer on the objective of [Och and Ney2004].

[Xiao et al.2011] also apply a discriminative latent variable model to SMT using the grammar of [Chiang2007]. They optimize the same MAP objective as [Blunsom et al.2008], and introduce a novel linear-time algorithm for generating the translation forest necessary for training. Given a synchronous parse tree covering a source sentence and its reference translation (i.e. a reference derivation), their algorithm greedily generates a forest by iterating over each node in the derivation. For each incoming hyperedge of each node, they append the list of competing derivations for the hyperedge to a list from a table of derivation rules. They then apply a transformation to the node to infer a new reference derivation consistent with the source and reference sentences. Using this algorithm, they derive a set of reference derivations and non-reference derivations[3] that are used in estimating the model. While their method may be linear in the number of nodes of a reference derivation, they still require the initial step of bi-parsing to infer a reference derivation. Bi-parsing is $O(n^6)$ in the worst case. To apply the MAP objective function, one must have a complete reference derivation. Due to the constraints of the grammar, however, a reference derivation may not be inferable for many sentences in the bitext. [Blunsom et al.2008] use only sentences for which a reference is inferable using an ITG alignment algorithm [Wu1997] and filter out the rest. [Xiao et al.2011] augment the set of rules allowed by [Chiang2007] with added rules using the method of [Zhang et al.2008].

2.4 Fully discriminative models

While mildly discriminative approaches have performed well, they tend to struggle in scaling up to parameterize many features. It has been found that MERT is only feasible when parameterizing approximately 15-30 terms in the mildly discriminative model [Chiang et al.2008]. Furthermore, the feature functions in the mildly discriminative models tend to be generative sub-models. Thus, while the objective on which the λ_ms are tuned may be well-correlated with the objective of maximizing translation accuracy, the sub-models themselves may not be.

[3]Derivations that do not yield the reference sentence in a bitext.

Fully discriminative approaches to SMT view the task as a structured classification problem, where the objective is to learn a mapping from input sentences $\mathbf{s}$ to target language translation $\mathbf{t}$. Like the generative and mildly discriminative approaches, fully discriminative approaches usually assume the existence of a hidden correspondence structure between $\mathbf{s}$ and $\mathbf{t}$, and for each tuple $(\mathbf{s}, \mathbf{t}, \mathbf{h})$ they assume a feature vector $\Phi(\mathbf{s}, \mathbf{t}, \mathbf{h})$ over the tuple. For linear models, the goal is to parameterize a weight vector $\mathbf{w}$ for the classification rule:

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \mathbf{w} \cdot \Phi(\mathbf{s}, \mathbf{t}, \mathbf{h})$$

A difficulty in SMT is that unlike in many other structured classification problems there are many possible valid (i.e. "correct") outputs for a given input, rather than a single one. The training set, however, usually only has one gold-standard reference translation for a given input sentence. Furthermore, there may be many valid correspondence structures $\mathbf{h}$, making the problem of deciding what to learn rather non-trivial.

[Liang et al.2006] present a finite-state model optimized by an online perceptron training algorithm. Their algorithm iterates multiple times over the training corpus. A beam-decoder generates a translation hypothesis $\mathbf{t}$ with hidden correspondence $\mathbf{h}$ for each sentence $\mathbf{s_i}$ in the corpus using parameter vector $\mathbf{w}$. They then use the perceptron update rule to update $\mathbf{w}$ after each translation is generated:

$$\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{s_i}, \mathbf{t}', \mathbf{h}') - \Phi(\mathbf{s_i}, \mathbf{t}, \mathbf{h})$$

where $(\mathbf{t}', \mathbf{h}')$ is the target the algorithm updates towards. Following [Collins2002] they then average over the parameter vectors estimated on each iteration. They experiment with three parameter update methods. The first, which they call *bold updates*, updates towards the Viterbi derivation under the current model of the gold-standard reference translation. The second more conservative *local update* strategy generates an n-best list of translations for each sentence in the bitext and updates towards the one that has the highest evaluation measure with respect to the gold-standard reference. The third is a *hybrid update* which does a bold update if the reference is inferable under the model, and a local update otherwise. They find that the bold and hybrid updates are too aggressive. Often, it's really hard for the decoder to derive the reference – the Viterbi derivation cost of the reference may be very high under the model making it an undesirable target. They achieve highest translation accuracies on the development and test set using local updates.

[Watanabe et al.2007] apply the Margin Infused Relaxed Algorithm (MIRA), an online variant of the large-margin training algorithm of [Taskar et al.2004b]. The main idea behind the MIRA update rule is to keep the updates to the weight vector minimal, but at least as large as the loss of the incorrect classification. Their

update rule is:

$$\hat{\mathbf{w}}^{i+1} = \arg\min_{\mathbf{w}^{i+1}} ||\mathbf{w}^{i+1} - \mathbf{w}^{i}|| + C \sum_{\hat{\mathbf{t}},\mathbf{t}} \xi(\hat{\mathbf{t}}, \mathbf{t})$$

subject to

$$\mathbf{w}^{i+1} \cdot \mathbf{h}(\mathbf{s}, \hat{\mathbf{t}}) - \mathbf{w}^{i+1} \cdot \mathbf{h}(\mathbf{s}, \mathbf{t}) + \xi(\hat{\mathbf{t}}, \mathbf{t}) \geq L(\hat{\mathbf{t}}, \mathbf{t}; \mathbf{t}')$$

where $\xi(\hat{\mathbf{t}}, \mathbf{t})$ is a slack variable greater than 0, C is a constant controlling the strength of the update, and $L(\hat{\mathbf{t}}, \mathbf{t}; \mathbf{t}')$ is the loss function measuring the difference between the BLEU score [Papineni et al.2002] of oracle $\hat{\mathbf{t}}$ and candidate $\mathbf{t}$ with respect to gold-standard reference $\mathbf{t}'$. Since BLEU is a document-level evaluation measure, they accumulate translations for the entire training set to compute the BLEU score. Like [Liang et al.2006], [Watanabe et al.2007] use local updating, except that when they select the oracle, they select it from the union of the n-best list output by the decoder and the oracle of the previous iteration.

[Chiang et al.2008] also employ MIRA to train a discriminative SMT system. Their learning approach differs from that of [Watanabe et al.2007] in three ways. First, they use a different update rule than that of [Watanabe et al.2007]. Second, instead of accumulating translations to compute the BLEU score, they keep an exponentially-weighted moving average of previous translations. Finally, to select an oracle, they propose a hybrid local-update method. As an oracle, they select a translation that maximizes a combination of BLEU score and model score.

[Tillmann and Zhang2005] present a finite-state discriminative phrase lexicalized block re-ordering model that parameterizes the generation of a block b_i with orientation o_i relative to its predecessor block b_{i-1}. The decoder searches for the block sequence that maximizes $\prod_{i=1}^{n} p(b_i, o_i | b_{i-1}, o_{i-1})$. They discriminatively estimate a local model that estimates $p(b_i, o_i | b_{i-1}, o_{i-1})$. For each block orientation pair $(b, o; b', o')$ they assume a feature vector $f(b, o; b', o') \in \mathbb{R}^d$, and try to estimate a parameter vector w for each block orientation pair for sentence s:

$$p(b, o \in \{L, R\} | b', o'; w, s) = \frac{\exp(w \cdot f(b, o; b', o'))}{Z(b', o'; s)}$$

The w's are estimated by an online training algorithm to maximize the conditional log-likelihood objective.

[Tillmann and Zhang2006] apply a discriminative global training algorithm to the finite-state phrase block sequence model of [Tillmann and Zhang2005]. Rather than training local models for each block orientation pair, they optimize a global parameter vector w. The score of a block sequence is given by $s_w(b_1^n, o_1^n) = \sum_{i=1}^{N} w \cdot f(b_i, o_i, b_{i-1})$. To estimate w, they present the Approximate Relevant Set method. For each block sequence $\mathbf{z} \in V(S_i)$ corresponding to a whole translation candidate for sentence S_i with BLEU score $\textsc{Bl}(\mathbf{z})$, they let $V_K(S_i)$ be

the set of candidates with highest BLEU score. They train to maximize the "cost margin" so that the cost assigned to the candidates in $V_k(S_i)$ is higher than the costs of the incorrect alternatives in $V(S_i)$. Because $V(S_i)$ is too large to be enumerated, they approximate it by iteratively selecting a "relevant set" of incorrect alternative translations that have low cost under the model and performing stochastic gradient descent to maximize the cost margin. Their objective function is:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left[\frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{w}, V_K(\mathbf{s}_i), V(\mathbf{s}_i)) + \lambda \mathbf{w}^2 \right]$$

$$\text{where } \Phi(\mathbf{w}, V_K, V) = \frac{1}{K} \sum_{\mathbf{z} \in V_k} \max_{\mathbf{z}' \in V - V_K} \psi(\mathbf{w}, \mathbf{z}, \mathbf{z}')$$

$$\psi(\mathbf{w}, \mathbf{z}, \mathbf{z}') = \phi(s_w(\mathbf{z}), \mathrm{Bl}(\mathbf{z}); s_w(\mathbf{z}'), \mathrm{Bl}(\mathbf{z}'))$$

$$\phi(s, b; s', b') = (b - b') \max\left(0, (1 - (s - s'))^2\right)$$

[Arun and Koehn2007] compare the averaged perceptron approach to MIRA on a finite-state phrase model. While they find no appreciable difference in the accuracies of the two models, they validate the finding of [Liang et al.2006] that local updates outperform both bold updates and the updates of [Tillmann and Zhang2006].

Like [Tillmann and Zhang2005], [Wellington et al.2006a] present a tree-to-string discriminative local model for SMT. They approach MT as a structured prediction problem where they classify *inferences* in a synchronous parse tree. Rather than discriminating between possible trees, they train a set of confidence-rated binary classifiers that determine whether to make individual inferences. Their classifiers are ensembles of ℓ_1-regularized confidence-rated gradient boosted decision trees. For each inference proposed by their translator, a classifier for that inference type computes a confidence that the inference is correct. The multitree output by the translator is the one where the sum of inference confidences is highest (alternatively, whose *cost* is lowest). By decomposing their model into a set of confidence-rated classifiers, they are able to model the distributions of different inference types, allowing them to parameterize different sets of features for different inference types. Furthermore, because different inference types have different underlying distributions, constructing a classifier for each inference type allows them to model that distribution rather than to assume one distribution over whole translations. To train their classifiers they must generate labeled training examples. Given a synchronous treebank, each node in the treebank corresponds to an inference. They decompose each synchronous tree in their corpus into a partially ordered set (poset) of inferences. For each poset, they randomly select a totally ordered sequence. They generate a positive training example for each inference

in the sequence. To generate negative training examples, given each prefix of the sequence, for each inference that can possibly follow that prefix that differs from the true inference following that prefix, they add a negative example.

Chapter 3

System Overview

3.1 Translation prediction

In this work, a *translator* is an algorithm that searches the set of possible translations $\mathbf{T}$ of an input sentence $\mathbf{s}$ to find the best translation $\hat{\mathbf{t}} \in \mathbf{T}$ according to a utility function $E(\mathbf{s}, \mathbf{t})$. This function represents translation quality as a cost, so that

$$\hat{\mathbf{t}} = \arg\min_{\mathbf{t} \in \mathbf{T}} E(\mathbf{s}, \mathbf{t}) \tag{3.1}$$

To facilitate the search, we follow most other SMT works (e.g. [Liang et al.2006, Blunsom et al.2008]) and posit a hidden correspondence structure $\mathbf{C}$ between $\mathbf{s}$ and $\mathbf{t}$. We define $\mathcal{C}(\mathbf{s}, \mathbf{t})$ as the set of possible correspondence structures between $\mathbf{s}$ and $\mathbf{t}$, and define $E(\mathbf{s}, \mathbf{t})$ as:

$$E(\mathbf{s}, \mathbf{t}) = \min_{\mathbf{C} \in \mathcal{C}(\mathbf{s}, \mathbf{t})} E(\mathbf{C}) \tag{3.2}$$

Each correspondence structure in the set $\mathcal{C}(\mathbf{s}, \mathbf{t})$ represents the translationally equivalent components of $\mathbf{s}$ and $\mathbf{t}$. For a set of possible translations $\mathbf{T}$ for sentence $\mathbf{s}$, the translator searches over the set $\bigcup_{\mathbf{t} \in \mathbf{T}} \mathcal{C}(\mathbf{s}, \mathbf{t})$ and selects the structure with least cost:

$$\hat{\mathbf{t}} = \arg\min_{\mathbf{t} \in \mathbf{T}} \min_{\mathbf{C} \in \mathcal{C}(\mathbf{s}, \mathbf{t})} E(\mathbf{C}) \tag{3.3}$$

After the translator outputs its best guess about $\mathbf{C}$, a post-process deterministically extracts $\mathbf{t}$. The algorithm used to search over the set $\bigcup_{\mathbf{t} \in \mathbf{T}} \mathcal{C}(\mathbf{s}, \mathbf{t})$ can be constrained by a reference translation $\mathbf{t}$, thereby forcing the translator to search for the least-cost correspondence structure between $\mathbf{s}$ and $\mathbf{t}$. As we will see in Section 3.3, this property will be very useful in selecting an optimal scoring function $E(\mathbf{C})$.

In this work, $\mathbf{C}$ is a *multitree* or *synchronous parse tree* (e.g. [Wu1997], [Melamed2004]). Since the translator searches over the set of possible multitrees

consistent with a sentence $\mathbf{s}$, the translator can be conceptualized as a generalized parser [Melamed and Wang2004]. When the translator is also constrained by a reference translation $\mathbf{t}$, it can be conceptualized as a multiparser. An example of a multitree is given in Figure 3.1. One side of the multitree covers the input sentence and the other side covers a translation. Each aligned node pair in the multitree represents translational equivalence between source and target word sequences dominated by the node pair. During parsing, every multitree is represented by an *item*. The construction of a multitree can be decomposed into a sequence of decisions called *inferences*. In our bottom-up approach, each inference produces a *consequent* item representing a multitree from a pair of *antecedent* items representing its sub-multitrees. Each inference fired by the translator has a cost given by the translator's *model*. The strategy we use to select the order in which inferences are fired is such that inferences are fired in ascending order of cost. The cost of a multitree is a function of the costs of the inferences fired to construct it. Without any pruning of items, the first item inferred that represents a multitree that fully covers the source sentence will be $\hat{\mathbf{c}} \in \mathcal{C}(\mathbf{s})$, where $\mathcal{C}(\mathbf{s}) = \bigcup_{\mathbf{t}\in\mathbf{T}} \mathcal{C}(\mathbf{s},\mathbf{t})$ and $\hat{\mathbf{c}}$ is the least-cost multitree covering the source sentence.

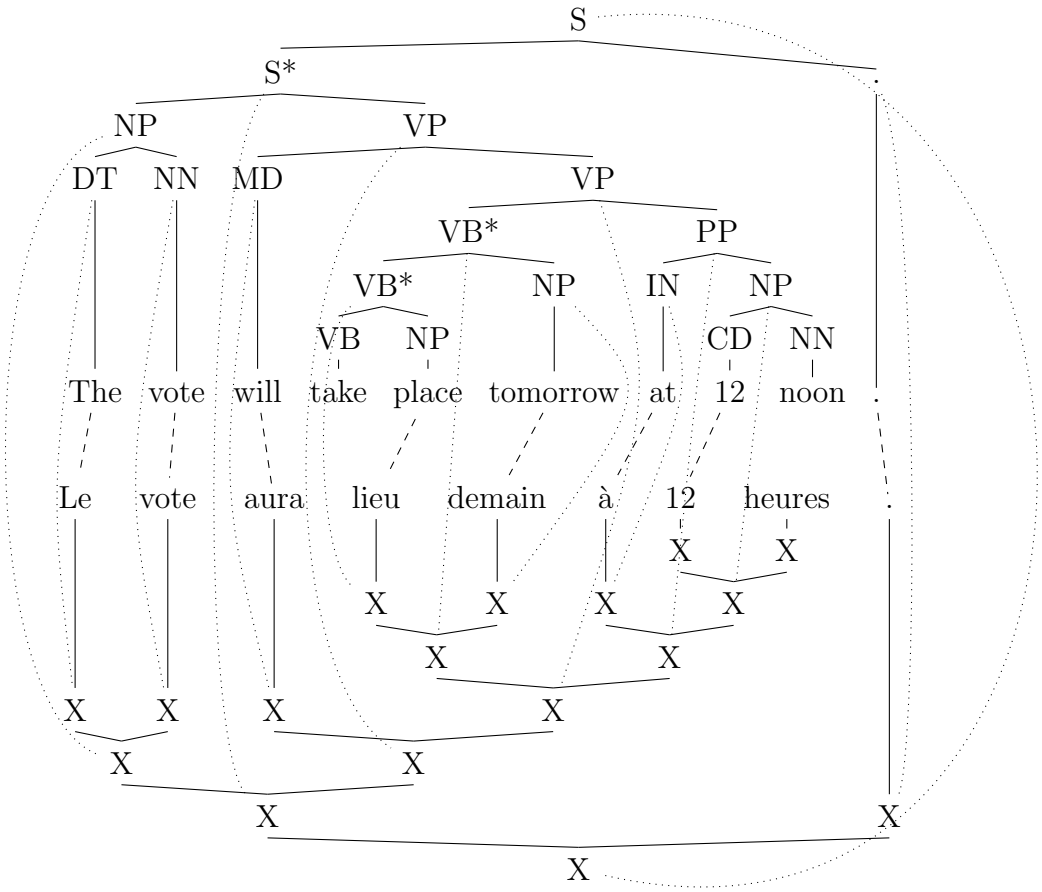
3.2 Translation Model Training Algorithm

In SMT, the cost function $E(\mathbf{C})$ must be learned from training data. As discussed in Chapter 1 we wish for the cost function to model the inference process undertaken by the translator in inferring structure $\mathbf{C}$ for $\mathbf{s}$,$\mathbf{t}$. The task corresponds to learning which hyperedges to traverse in the hypergraph for $\bigcup_{\mathbf{t}\in\mathbf{T}} \mathcal{C}(\mathbf{s},\mathbf{t})$. Therefore the function $E(C)$ for $C \in \bigcup_{\mathbf{t}\in\mathbf{T}} \mathcal{C}(\mathbf{s},\mathbf{t})$ must weight the hyperedges in the hypergraph, or, correspondingly, assign cost to inferences made by the translator. We therefore need to train the model on a set of labeled inferences, where the labels indicate the utility of the inferences in inferring an optimal structure $\mathbf{C}$[1]. Unfortunately, such a training set is not readily available. In SMT, models are usually trained on parallel corpora, or *bitexts*. The structures we actually wish to learn are hidden. We must therefore synthesize a labeled training set from the inferences proposed by a translator that is the same as the one used during test time, and label the inferences according to a criterion of how useful they are in inferring a correspondence structure yielding a good translation. We refer to the criterion used for selecting and labeling training examples as the *global objective function*, which should be, but need not be, well-correlated with the measure by which the output is evaluated. We defer discussion of how we synthesize the training set to Section 3.3.

The training procedure for the translator involves two tasks – determining

[1]Or the utility of the inferences in inferring an optimal *set* of structures $\mathcal{C}$.

Figure 3.1: An example binary multitree for an English-French sentence pair. The dashed lines indicate word alignments. The dotted lines indicate node alignments. The English word "noon" and the French word "heures" are both unaligned.



the examples to train on and updating the weights of the model based on those training examples. We therefore train the model by an iterative two-step process. The first step generates training examples from the inferences proposed by the translator(s) using the latest model. These examples are added to a global training set. The second step rebuilds the model. This process iterates for a fixed number of iterations. The pseudocode for our iterative training algorithm is presented in Algorithm 1.

We can initialize the model in any way we like in line 4. For example, we can include an inductive bias for the model on subsequent iterations. In line 7

Algorithm 1 The iterative model training procedure

Input: training bitext $\mathcal{B}$, model h, maximum iteration M

```
1: procedure TRAIN
2:     i ← 0
3:     I ← {}
4:     h ← INITMODEL()
5:     while i < M do
6:         for each (s, t) ∈ B do
7:             I ← I ∪ GENERATETRAININGEXAMPLES(B, h)
8:         h ← TRAIN(I)
```

of the algorithm we take the union of the training set generated on the previous iteration and the training set generated in the current iteration. We retrain the model in line 8 using all the examples generated on all iterations thus far. Doing so allows the translator to retain what it has already learned while taking into account new information about the relative merits of previously unexplored states (i.e., hypergraph nodes). If we were to discard previous training examples, the translator would likely fire inferences that it had previously learned not to fire.

3.3 Parsing Coordinator

To synthesize the training set of inferences that we use to train the model, we present a novel architecture and algorithm for example generation. Our architecture revolves around a parsing *coordinator* that coordinates the inference processes of a *supervisor* and a *predictor*. The supervisor and predictor are inference engines each of which keeps an agenda of items. The supervisor is like an "expert" because it is given the reference translation and is constrained to infer it. It can also be constrained to infer only certain kinds of correspondence structures $\mathbf{c}$. The predictor is the same translator as the one used at test time, which doesn't know the reference translation. Apart from that, the two parsers are exactly the same. The parsing coordinator filters and labels the inferences proposed by the predictor to use as a training set according to an objective function. All the objectives with which we experimented utilized the supervisor to determine the utility of any given inference fired by the predictor in inferring a multitree yielding a good translation. Because the supervisor and predictor have identical configurations[2], the supervisor has the property that the set of items it can infer is a subset of the set of items that the predictor can infer. The coordinator has a function that can determine

[2]apart from the fact that the supervisor is constrained by the reference translation whereas the predictor is not

whether an item proposed by the predictor is inferable by the supervisor. If the function determines that an item is inferable by the supervisor, we say that the inference that inferred the item is *approved* by the supervisor.

The coordinator can use the supervisor's approval of an item in two ways. First, its objective function can use the supervisor's approval or lack thereof of an item to label the inference inferring it, thereby labeling the training example. Second, the coordinator can use the supervisor's approval of items to control how far the predictor strays from inferring a multitree yielding the reference translation. To do so, we utilize a distance measure counting the maximum number of inferences between an item inferred by the predictor and an approved item. In terms of the hypergraph, it measures the number of hyperedges between a node (corresponding to an item) and the approved predecessor node with the longest distance from the node being evaluated. The coordinator can place a limit on the maximum distance a predictor is allowed to have from an approved multitree, thereby limiting the number of incorrect states the predictor is allowed to explore.

The constraints we apply to the supervisor have an impact on the predictor items it can approve. In turn, this has an effect on whether the inference inferring the item becomes part of the training set and how it is labeled by the objective function. For example, we can constrain the supervisor to infer a single pre-specified multitree for each sentence pair in the bitext. The only inferences proposed by the predictor approved by the supervisor will be the ones that are involved in inferring the pre-specified multitree. If the coordinator employs a strict labeling strategy, that is, if it labels all inferences inferring approved items as positive and all others as negative, the translator will be trained to infer the single pre-specified structure for each sentence pair. Additionally, when constrained to infer a single tree, the predictor will be allowed to explore only a certain distance from that one tree. Alternatively, if we allow the supervisor to infer any multitree consistent with the sentence pair, the translator will learn to infer any multitree yielding the reference translation. Furthermore, removing the single tree constraint will allow the predictor to explore more correct and incorrect states, since the predictor will be allowed to explore a given distance from *any* reference-yielding multitree.

The parsing coordinator is presented in Algorithm 2. It operates as follows. First, the supervisor $\mathcal{S}$ and predictor $\mathcal{P}$ are initialized in lines 2 and 3 and the training set I is initialized to an empty set. The agendas of both the supervisor and the predictor are populated by the initialization functions. The main loop in lines 5–20 iterates until a given termination condition is satisfied by the predictor. Lines 6–9 of the algorithm check whether the predictor can *expand* any more items. Expansion refers to when an item is dequeued from an agenda and is used to infer new consequent items. If the predictor cannot expand any more items but the supervisor can, the next item of the supervisor is expanded and returned to the predictor. The predictor can run out of items to expand for two reasons. Either the

Algorithm 2 The abstract parsing coordinator

Input: Supervisor $\mathcal{S}$,
Predictor $\mathcal{P}$,
Sentence pair $(\mathbf{s}, \mathbf{t})$,
Supervisor model $h^{\mathcal{S}}$,
Predictor model $h^{\mathcal{P}}$,
Exploration threshold δ
Output: Training set I

procedure GENERATETRAININGEXAMPLES($\mathcal{S}, \mathcal{P}, (\mathbf{s}, \mathbf{t}), h^{\mathcal{S}}, h^{\mathcal{P}}$)
  $\mathcal{S}$.INITIALIZE($h^{\mathcal{S}}, (\mathbf{s}, \mathbf{t})$)
  $\mathcal{P}$.INITIALIZE($h^{\mathcal{P}}, \mathbf{s}$)
  $I \leftarrow \{\}$
  repeat
    if $\mathcal{P}$'s agenda is empty and $\mathcal{S}$'s is not **then**
      $\mathbf{i} \leftarrow \mathcal{S}$.DEQUEUE()
      if $\mathcal{S}$ does not prune **i then**
        $\mathcal{S}$.ENQUEUE($\mathcal{S}$.EXPAND($\mathbf{i}, h^{\mathcal{S}}$))
    else if $\mathcal{P}$'s agenda is empty and $\mathcal{S}$'s agenda is empty **then** ▷ Abort
      break
    else
      $\mathbf{i} \leftarrow \mathcal{P}$.DEQUEUE()
    if $\mathcal{S}$.APPROVES($\mathbf{i}$) **then**
      $\mathcal{S}$.ENQUEUE($\mathcal{S}$.EXPAND($\mathbf{i}, h^{\mathcal{S}}$))
    $I \leftarrow I \cup$ MAKEEXAMPLE($\mathbf{i}, \mathcal{S}$.APPROVES($\mathbf{i}$))
    if $d(\mathbf{i}) \leq \delta$ **then**
      if $\mathcal{P}$ does not prune **i then**
        $\mathcal{P}$.ENQUEUE($\mathcal{P}$.EXPAND($\mathbf{i}, h^{\mathcal{P}}$))
  until termination condition is satisfied
  return I

predictor has pruned an item necessary for satisfying the termination condition[3], or the item needed to infer a multitree satisfying the termination condition is not inferable by the predictor, e.g. because of its logic. The EXPAND function in lines 9 and 19 takes an item **i**, finds other items that can participate in an inference with **i**, and proposes a new set of inferences weighted by the model. It then computes a cost for the consequent item of each proposed inference and returns the weighted items. MAKEEXAMPLE in line 16 makes a set of training examples given **i** and a determination of whether they are approved by the supervisor. It first computes a feature vector for the inference whose consequent item is **i**. It then determines whether the example should be positive or negative. Line 17 determines whether $\mathcal{P}$ should try to expand the item. Finally, in line 19, $\mathcal{P}$ expands **i** and puts the consequent items onto its agenda.

To facilitate limiting the amount of exploration, the coordinator must be able to determine the distance between an item proposed by the predictor and its most distant ancestor item approved by the supervisor whose parent item[4] is not approved by the supervisor. $d(\mathbf{i})$ computes this distance for item **i**. This distance measure is illustrated in Figure 3.2. If **i** is approved by $\mathcal{S}$, then $d(\mathbf{i}) = 0$. We call $d(\mathbf{i})$ the *exploration distance*. δ, the *exploration limit*, is a parameter we set that controls how far we allow the predictor to stray from the search paths allowed by the supervisor. We also refer to this as the amount of *exploration*[5] the predictor is allowed. By setting δ to be a high value, we can allow the translator to build long sequences of incorrect inferences before correcting the predictor and leading it back to a correct inference. Limiting the amount of exploration in this manner is related to the early update approach of [Daumé and Marcu2005], except that as soon as their algorithm makes an error, they restart training from the correct solution, whereas we allow the translator to explore the space of incorrect solutions further, to give the model a bigger picture of incorrect states. The approach of seeding the predictor's agenda with supervisor items is also related to the early update strategy of [Collins and Roark2004], where they restart training as soon as the correct parse falls outside the beam. Our approach guides the predictor back onto a path followed by the supervisor after the predictor has been given the opportunity to make incorrect inferences (i.e. make mistakes), which the training algorithms can use as negative examples.

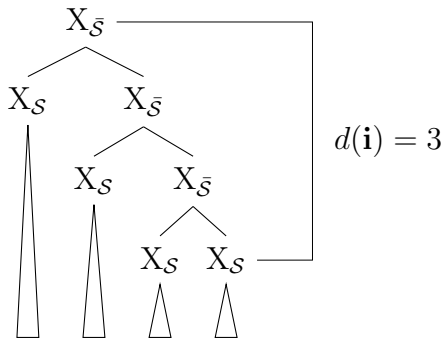
The parsing coordinator architecture affords us many advantages in terms of flexibility and efficiency. The coordinator treats the underlying predictor and su-

[3]if, for instance, the termination condition is that the predictor infer a multitree consistent with the reference translation

[4]i.e. the consequent item of the inference for which the supervisor-approved item is an antecedent

[5]We borrow the term from reinforcement learning. It refers to non-optimal actions taken to observe previously unexplored states.

Figure 3.2: An illustration of the distance measure between predictor items and supervisor-approved items using only one side of a multitree. The root node corresponds to a predictor-proposed item $\mathbf{i}$. Nodes labeled $X_{\mathcal{S}}$ correspond to items approved by the supervisor, and nodes labeled $X_{\bar{\mathcal{S}}}$ correspond to items not approved by the supervisor. The distance here is 3, since the greatest height from root to any $x_{\mathcal{S}}$ is 3.



pervisor as black boxes, allowing us in principle to use the architecture not only for tree-structured translation algorithms as we do here, but, more generally for many structured prediction tasks. The architecture allows us to explicitly control the search procedure of the supervisor independently of the predictor to select the objective on which to update the model. It allows us to explicitly control the amount of exploration done by the predictor. It has the advantage that it constructs training examples *while* the predictor follows its search procedure, rather than post hoc. This property allows us to learn from incomplete multitrees as well as complete ones, allowing us to exploit more of the search space for learning. The coordinator can determine how to label the training examples without needing the predictor to infer a complete output. If the predictor runs out of items to expand, or if we terminate the search prematurely, the coordinator will still generate useful training examples.

A distinguishing characteristic of our parsing coordinator algorithm is that it does not generate any inferences that are not required for learning or for guiding the predictor. The supervisor acts only when the predictor needs guidance (i.e. when the predictor's agenda is empty and it has not generated a reference translation). Furthermore, the predictor does not need to recreate any inferences fired by the supervisor. The supervisor needs only to pass an item to the predictor, and the predictor can use it to continue parsing. This process is an advantage over having each inference engine build up a structure separately, and then comparing the structures *post-hoc*.

An alternative architecture we considered was one that one that would have been sufficient for the perceptron update approach [Liang et al.2006] and MIRA

[Watanabe et al.2007], where we first had the supervisor and predictor infer a whole multitree or set of multitrees. We would have then compared the inferences generated by the supervisor to those generated by the predictor. This architecture did not allow us to use permissive example generation to train the model (discussed in Chapter 4). Furthermore, when the supervisor or predictor were unable to achieve a complete parse, we could not learn from the inferences generated.

To illustrate the flexibility of our architecture we now demonstrate how it can be used to implement the training scheme of [Wellington et al.2006a]. Recall that [Wellington et al.2006a] attempt to learn a single multitree for each sentence pair in their bitext. Moreover, they randomly selected a single totally ordered sequence of inferences used to infer each multitree. The inferences in the total ordering became positive examples for the classifiers. For each prefix of each inference sequence, they considered the alternative subsequent inferences that could have been made, and added them as negative examples. This training routine can be implemented merely by specifying some of the components of Algorithm 2. Specifically, we set $h^{\mathcal{S}}$ to return zero cost for any inference inferring part of the multitree and infinite cost to all others; we set δ to be 0 (as to allow only the alternative inferences to be added as negative examples and not allowing for expansion of incorrect multitrees); and we implement MakeExample so that it makes all inferences proposed by the predictor that have zero cost under $h^{\mathcal{S}}$ positive examples and all others negative.

3.4 Comparison to other work on learning to search

We pause here to situate our approach in the literature, specifically discussing the similarities and differences of our approach to other works on learning to search.

[Daumé and Marcu2005] frame the problem of structured learning as a problem where the objective is to learn a model to rank the items on the agenda of states, so that "good" hypotheses are expanded before "bad" hypotheses. They assume that each hypothesis is scored by $g + h$, where g is a linear function of features of a hypothesis x, and where h is a heuristic component (thereby allowing for different search strategies, such as A*). For a given node s in their search graph, and goal state y, they assume that they can tell whether s is on a path from the start state to y. Their objective is that the first state that's dequeued from the agenda should be on the path to y, and the queue should always contain a state on the path to y. They train by performing a search. Whenever they dequeue a state that is not on the path to y or whenever the agenda does not contain a state leading to y, they update the parameter vector. When one of these situations occurs, they clear the agenda and insert all the correct states. They experiment with two parameter update rules – perceptron updates [Rosenblatt1958] and approximate

large margin updates (ALMA [Gentile2002]), and apply this learning method to syntactic chunking and tagging. Unlike [Daumé and Marcu2005], we do allow for some exploration of the search graph before bringing the predictor back to the correct path. In this work we do not incorporate a heuristic component to our cost function, but we can, and might in future work.

Like the experiments of [Wellington et al.2006a], the approach of [Daumé and Marcu2005] attempts to learn an *optimal policy* for navigating the search graph. This target, however, is suboptimal in that it does not allow an algorithm exploring the search graph to recover from mistakes – if a parser or sequence labeler finds itself considering an incorrect state, any decision it makes thereafter may be unreliable since it was never trained to deal with this scenario. To deal with these problems, [Daumé III et al.2009] present SEARN, a meta-algorithm that decomposes structured prediction problems into binary classification problems, much as we do. SEARN can apply to any loss function, feature set, or search space. The main loop of the algorithm uses a policy (initialized to be the optimal policy) to generate a set of labeled training examples by running the policy over the search space for each input in the training data, generating a path through the search space for each input, thereby creating a single training example for each state on each path. A new set of binary classifiers is trained from this example set, and is stochastically interpolated[6] with a bias with the previous iteration's policy. That is, given the policies estimated on iteration $i-1$ and i, an action is randomly selected with probability β from the policy of iteration i and with probability $1-\beta$ from the policy of iteration $i-1$. Of all the related published work, SEARN is closest to our method. While SEARN comes with some theoretical guarantees on its performance, we cannot apply it to our problem, because we cannot select an optimal policy with which to initialize our system. There are many possible policies we can follow to infer translation y from input sentence x, but we have no good way to *a priori* decide which one is optimal.

Recall that translation can be viewed as a generalization of parsing [Melamed2004]. [Neu and Szepesvari2009] note the similarities between the problem of training a parser and the *inverse reinforcement learning* (IRL) problem. In IRL, an agent uses a utility function (called a *reward function*) to determine which actions to take in an environment. Given an agent following an unknown reward function, the goal of IRL is to estimate the agent's reward function from the sets of its observed *trajectories*. The environment and the agent's search strategy are modeled as an episodic Markovian decision process (MDP). Formally, an MDP is a 5-tuple $M = (\mathcal{X}, \mathcal{A}, T, \mathcal{X}_T, r)$ where $\mathcal{X}$ is a countable (but possibly infinite) set of states; $\mathcal{A}$ is a finite set of actions; T is a transition function, where $T(x_{t+1}|x_t, a)$

[6]Meaning they figuratively flip a coin with a bias, and if it comes up heads they use the most recently trained classifier to score the state, otherwise they use one of the previous ones (stochastically interpolating between the models).

is the probability of transitioning from state x_t to x_{t+1} upon taking action a; $\mathcal{X}_T$ is a set of terminal states; and r is a real-valued reward function determining the reward upon executing action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$. A policy is a function $\pi : \mathcal{A} \times \mathcal{X} \rightarrow [0, 1]$, such that $\sum_{a \in \mathcal{A}} \pi(a|x) = 1$. The value of a policy is the probability of taking action a in state x. The value of a state x under policy π and reward function r is given by:

$$V_r^\pi(x) = \mathbb{E}_\pi \left[\sum_{t=0}^{H-1} r(x_t, a_t) | x_0 = x \right] \tag{3.4}$$

where the sum is over the set of transitions taken under policy π from state x to a terminal state and H is the first time when the process enters a terminal state.

In IRL, a training algorithm is given an MDP without a reward function specified, and a list of *trajectories* $\mathcal{D}$ through the MDP generated by an expert following a policy in the MDP. The goal of training is to find a reward function r such that the trajectories that one obtains by following an MDP with that reward function are close to the trajectories generated by the expert. This is done by defining and minimizing a dissimilarity function: $\hat{r} = \arg\min_r J(r; \mathcal{D})$. $J(r; \mathcal{D})$ measures the dissimilarity between the observed behavior $\mathcal{D}$ of an expert and a reward function hypothesized to be the one followed by the expert. [Neu and Szepesvari2009] demonstrate how the problem of finding the best parse under a PCFG with the Viterbi semiring is equivalent to following an optimal policy in a deterministic MDP (where taking an action can only lead to one possible successor state). They then show how several standard IRL algorithms can be used to derive well-known parser training algorithms in the literature when the parsing problem is properly defined as an MDP, including max-margin parsing [Taskar et al.2004b], a variant of perceptron parsing [Collins and Roark2004], and max-margin reranking [Charniak and Johnson2005].

The problem of learning an optimal translation model can also be viewed as a problem of learning an optimal reward function for an MDP. Our problem consists of a translator agent trying to learn a reward function that allows it to generate translations (i.e. trajectories) close enough to those output by the expert. If the expert is constrained to infer only one hidden correspondence structure for each sentence pair, there will be a single trajectory for each sentence pair in the bitext. If the expert is configured to infer any multitree consistent with the sentence pair, then all of the resulting multitrees become trajectories that the algorithm can learn from[7]. Under the Viterbi semiring, the cost of each state is given by the V_r^π function in Equation 3.4. However, under the semiring we will use (described in Section 4.3), the cost of a state x under a policy π is the lowest reward for taking

[7]Although not all of them will be used, given that the space of possible multitrees is so large. Instead, a subset will be used that receive the lowest cost under the current iteration's model.

action a in any subsequent state to x. Therefore, when learning, we will not be trying to find the reward functions that maximize the cumulative reward earned by a policy; instead, we will try to find reward functions that maximize the lowest reward earned by a policy.

Chapter 4

System Details

4.1 Generalized parsing

Recall from Chapter 3 that the translator we employ at test time, and by extension the predictor and supervisor we use during training, predict multitrees to represent translational equivalence between a source language sentence and a translation. The algorithms behind the supervisor, predictor, and translator at test time can be described as parsing algorithms with different constraints and configurations. One way of specifying the algorithms behind each of these parsers is by framing them as generalized parsers [Melamed2004, Melamed and Wang2004]. In the generalized parsing framework a *parser* is any algorithm that infers a structure for input text(s). Many NLP tasks can be viewed as generalized parsing, including ordinary monolingual parsing, synchronous parsing, hierarchical alignment, some methods of translation evaluation, the inside/outside algorithm [Baker1979], and most methods of statistical machine translation (this is discussed in depth in [Melamed and Wang2004]). In MT, the parser's task is to infer a multitree – a synchronous parse tree covering each component of the input tuple (e.g. [Wu1997,Melamed2004]) and specifying the hierarchical correspondence between them.

A parser can be fully specified using five components: *logic*, *grammar*, *semiring*, *search strategy*, and *termination condition*.

Logic The logic defines a parser's possible states and transitions between states. It is specified in three parts:

- the set of *term type* signatures. The logics we use in this work define two term types: items and grammatical constraint terms. Under the logics we present, a single item always represents a single node in a multitree, although this does not hold for parsing logics such as Earley-style logics [Earley1970].

- the set of legal *inference types*. Inferences are expressed as follows: $\frac{x_1,\ldots,x_k}{y}$ means that antecedent terms $x_1, \ldots, x_k$ (which can be either items or grammatical constraint terms) infer consequent item y. We say the parser "fires" *inferences* to create new items.

- the set of *axiom* terms. Axioms are term instances with which the parser is initialized. The axioms typically come from the grammar and from the input sentence(s).

To illustrate these definitions, Table 4.1 presents a logic for a monolingual bottom-up parser using context-free grammar productions, from [Melamed and Wang2004]. Two item types are defined in this logic – one for terminal items representing a single word in the sentence and its position, and one for nonterminal items representing a nonterminal node in a parse tree. The grammar constraints come from CFG production rules in Chomsky Normal Form – unary terminating production rules and binary nonterminating production rules. The axioms initializing the parser are terminal items for each word in the input sentence and the production rule instances of the grammar. The logic allows for two types of inference – one that generates a nonterminal item from a terminal item and a unary production rule, and one that composes two nonterminal items, using a binary production rule to generate another non-terminal item. A parser can fire a scan inference for the i'th word w_i if that word appears on the right hand side of a terminating production rule. It can fire a compose inference when: 1) the labels of the antecedents match the right hand side of a nonterminating production in the grammar and 2) the spans of the items are adjacent and their relative order is the same as that of their labels in the right hand side of the nonterminating production.

For a generalized parser to act as a translator between two languages, its logic must have the following properties:

- It must take a single sentence as input

- Its item signatures and inference types must be bilingual

- Its logic must have a way to generate lexical items of the target language. In other words, the logic's axiom set must include items for each word in the vocabulary of possible translations of the sentence.

The predictor utilized by the parsing coordinator described in Chapter 3 satisfies these three properties. The supervisor also satisfies these properties, except that the target-language lexical items the logic proposes are constrained by a reference translation for the source-language sentence. In terms of the taxonomy of parsing algorithms described in [Melamed and Wang2004], the predictor is a translator and the supervisor is a multiparser.

Table 4.1: Logic MonoC. w_i are input words; X, Y, and Z are non-terminal labels; t is a terminal; i and j are word boundary positions; and n is the length of the input

Term Types	
terminal items	$\langle t, i \rangle$
nonterminal items	$[X; (i, j)]$
terminating productions	$X \Rightarrow t$
nonterminating productions	$X \Rightarrow YZ$
Axioms	
input words	$\langle w_i, i \rangle$ for $1 \leq i \leq n$
grammar terms	as given by grammar
Inference Rule Types	
Scan	$\frac{\langle t, i \rangle, X \Rightarrow t}{[X; (i-1, i)]}$
Compose	$\frac{[Y; (i, j)], [Z; (j, k)], X \Rightarrow YZ}{[X; (i, k)]}$

Grammar A grammar is a function that assigns a value to a grammatical constraint term instance in an inference. *In this dissertation, we define the value of the inference as the value of the grammatical constraint term used as an antecedent in the inference.* This value can range over booleans (in the case of determining whether an inference is valid); over real numbers bounded by $[0, 1]$ (in the case of determining the probability of an inference); or over arbitrary real numbers (for instance, confidence estimates [Turian et al.2006] or feature weights [Chiang2005]). For instance, a PCFG is a function mapping the set of productions in the grammar $\mathcal{R} \rightarrow [0, 1]$, so that a nonterminating production $X \Rightarrow YZ$ would have probability $P(X \Rightarrow YZ)$ and a terminating production $X \Rightarrow t$ would have probability $P(X \Rightarrow t)$. In this work, we restrict ourselves to grammars that assign a real-valued *cost* to each inference. The grammar determines the weight of each hyperedge in the search hypergraph that the predictor, supervisor, and test-time translator traverse. To learn to search well, we must evaluate at least some of the weights of the hyperedges based on the corresponding inference's utility in inferring a good translation. The training set output by the parsing coordinator is used to

learn some parts of the grammar. We elaborate on grammar learning in Section 4.5.5

Semiring The grammar assigns a cost to an inference. To assign a cost to an item, and consequently the multitree it represents, we use a *semiring*. A semiring R is an algebraic structure consisting of a set (the domain of the semiring), two binary operators (the additive operator $\oplus_R$ and the multiplicative operator $\otimes_R$), and two identity elements (the additive identity $\mathbf{0}_R$ and the multiplicative identity $\mathbf{1}_R$). When the identity of the semiring is clear, we omit the subscript. In this work, a semiring is used in the *semiring equation* to map items to values in the semiring's domain:

$$V(y) = \bigoplus_{\substack{x_1,\ldots,x_k \\ \text{such that} \\ \frac{x_1,\ldots,x_k}{y}}} \bigotimes_{i=1}^{k} V(x_i) \tag{4.1}$$

Like the grammar, the values (i.e. domain) of the semiring can be boolean values, probabilities, or other real-valued cost domains. However, the domain can also consist of complex structures. For instance, the Viterbi-derivation semiring presented in [Goodman1999] computes the most likely parse tree for a sentence, in addition to its likelihood. The Viterbi-n-best-derivation semiring computes the n most likely parse trees for a sentence and their likelihood. The derivation-forest semiring computes a packed parse forest of trees for a sentence and their costs. Parsers use the cost assigned to items by the semiring equation to encode a preference between alternative items/multitrees.

Search strategy The search strategy determines the order in which inferences should fire. Common search strategies include the strict bottom-up search strategy (for instance, CKY search in which smaller-span items must be inferred before larger-span items [Kasami1965, Younger1967]) and the best-first search strategy (in which the inference operating on the highest-scoring/least-cost consequent is selected first, for instance [Klein and Manning2003, Turian et al.2006]).

Termination condition The *termination condition* is a predicate that defines the conditions under which the parsing algorithm terminates. This predicate can be defined in terms of a single goal item (i.e. an item that must be inferred for parsing to stop); a set of goal items (i.e. any item within a pre-defined set must be inferred for parsing to stop); and/or another kind of limit on the parsing algorithm, for instance, the number of items generated, the amount of time spent parsing, or the amount of memory consumed by a parsing algorithm.

The generalized parsing algorithm in Algorithm 3 is a notational variant of that in [Melamed and Wang2004]. The algorithm first instantiates the axioms and evaluates them under the semiring. It then iteratively asks the search strategy **S** for a set of antecedent items. It looks for a grammatical constraint term that unifies with the antecedent items. It then fires an inference for those items and grammar term. Finally, it uses the semiring equation to compute a value for the consequent item y of the inference and stores it in **S**. This loop proceeds until termination condition **C** is satisfied.

Algorithm 3 The Generalized Parsing Algorithm

Input: Logic **L**
Grammar **G**
Semiring **R**
Search Strategy **S**, which also keeps track of items participating in a parse
Termination Condition **C**
Text tuple **T**

1: **for** each axiom $p' \in \mathbf{L}$ corresponding to a grammatical constraint $p \in \mathbf{G}$ **do**
2: $V(p') \leftarrow \mathbf{G}(p)$ ▷ $\mathbf{G}(p)$ is the value that the grammar assigns to p
3: **for** each non-grammar axiom $w' \in \mathbf{L}$ **do**
4: $V(w') \leftarrow \mathbf{1}_R$
5: **while** **C** is not satisfied **do**
6: Let $X = \{x_1, \ldots, x_k\}$ be a set of antecedents given by **S**
7: **for** each grammar term $\gamma \in \mathbf{L}$ such that γ unifies with X **do**
8: inference $I \leftarrow \dfrac{X;\gamma}{y}$
9: **for** all possible terms y that unify with consequent of I **do**
10: $Ants(y) \leftarrow \{X' = \{x'_1, \ldots, x'_l\} : \exists \dfrac{x'_1, \ldots, x'_l;\gamma}{y} \in L\}$
11: $V(y) \leftarrow \bigoplus_{X \in Ants(y)} \bigotimes_{i=1}^{|X|} V(x_i)$
12: add y to **S**

In the rest of this chapter, we specify the components of the supervisor and predictor used by the parsing coordinator in our experiments.

4.2 Logics

The logics (along with the grammars) define the search space explored by the supervisor and predictor. By firing inferences, they should be able to infer hidden correspondence structures representing patterns of translational equivalence between sentences. These patterns include:

- single word translation
- source word deletion (i.e. the translator should be able to decide not to translate a source word)
- target word insertion (i.e. the translator should be able to insert a target language word into the translation candidate even if it has no direct translational equivalent in the source)
- multiword translation (i.e. be able to jointly translate a sequence of source words into a sequence of target words)
- reordering of syntactic structures

Although individual words tend to be the atomic unit of meaning in natural languages that are written with spaces between words, most languages have multiword compounds that should be analyzed as a single unit of meaning. Examples in English include "would like" and "give shelter". These compounds might (and likely do) have translations in other languages that are not literal or one-to-one. The French word for "would like" is "voudrais", the word for "give shelter" is "héberger", and the phrase for "please" is "s' il vous plaît". We would like a logic that can support selecting the source- and target-language word sequences that should be translated as a unit. To have the translator perform insertion, deletion, and multiword translation, we utilize monolingual composition inferences that we call *fuse* inferences. A source fuse inference creates a segment of the source sentence that should be translated as a unit. A target fuse inference creates a target word sequence that can be a translation of a source-side word or segment. We can allow word deletions by adding any source word we wish to delete into a source fuse. We can allow word insertions by adding any target word we wish to insert into a target fuse.

4.2.1 Multiparsing Logic

Our multiparsing logic, *Logic MP* (LMP), is a specialization of Logic C presented in [Melamed and Wang2004]. It is utilized by the supervisor in the parsing coordinator. The logic takes a sentence tuple containing both a source sentence and reference translation as input. Logic MP can be found in Table 4.2. Like Logic MonoC, Logic MP defines two item types: terminal items and non-terminal items. A **terminal item** i is a triple $\langle t; d, i\rangle$ denoting a leaf node: word t is located at position i on tuple component d. A **non-terminal item** i is a pair $\langle X; \sigma\rangle$ containing a bilingual node **label** X and a bilingual **span** σ indicating the word positions dominated by the node in both components of the sentence tuple.

A label $X = [X_1; X_2]$ is a length-2 vector of nonterminals. When an item covers a single word on a single component of the sentence tuple, the NT will be the word's part-of-speech (POS) tag. We refer to such non-terminal items as monolingual items. The label of the other component will be $\varnothing$. A monolingual item label active on the source component is denoted as $X^1_\varnothing = [X_1; \varnothing_2]$, and a monolingual item label active on the target component is denoted as $X^\varnothing_2 = [\varnothing_1; X_2]$.

A *span* is a length-2 vector of word index pairs covering a possibly empty subsequence of each component of the sentence tuple. We denote a span as $\sigma = \begin{bmatrix} i_1, j_1 \\ i_2, j_2 \end{bmatrix}$. Monolingual spans are represented as $\sigma^1_\varnothing = \begin{bmatrix} i_1, j_1 \\ \varnothing \end{bmatrix}$ and $\sigma^\varnothing_2 = \begin{bmatrix} \varnothing \\ i_2, j_2 \end{bmatrix}$. We refer to the non-empty component of a monolingual span as the *active component*. In this work, we restrict our spans to those that are contiguous. [Wellington et al.2006b, Sogaard and Kuhn2009] and references therein showed that this restriction will render the parser incapable of generating complete multitrees over some sentence pairs. We hope to incorporate discontinuous constituents in future work.

Inference types Logic MP uses four kinds of inferences. These are:

- **Scans** – Scan inferences take a terminal item and a terminal grammar constraint as antecedents, and generate a monolingual non-terminal item as a consequent.

- **Monolingual compositions** – Monolingual compositions take two monolingual items as antecedents, where the two have the same active component. The resulting consequent is also a monolingual item. When the active component of an item is the source, we call the inferences *source fuses*. When the active component of an item is the target, we call the inferences *target fuses*.

- **Bilingual compositions** – Bilingual compositions take two monolingual items as antecedents, where the two have different active components. The resulting consequent is a bilingual item. A bilingual composition inference posits translational equivalence between a source-sentence substring and a target-sentence substring. It is a generalization of the word transduction inferences of [Wellington et al.2006a].

- **2x2 compositions** – 2x2 compositions take two bilingual items as antecedents, and return a bilingual item as a consequent. The inference can be *monotonic* or *inverted*. In a monotonic composition, the target components of the antecedents are in the same order as the source components. In an inverted composition, the target components of the antecedents are in inverted order with respect to the source components. [Melamed2004] presented more

general inference types by utilizing *role templates* (also called *precedence array vectors* (PAVs) in [Melamed and Wang2004]) that can represent any sort of synchronous reordering patterns. However, since our logic is restricted to contiguous items, there are only two possible relative re-orderings so we explicitly list them to decrease the symbolic complexity of their description.

The manner in which LMP defines the inference types imposes a partial ordering on the inferences: monolingual inferences must be fired before bilingual inferences, and bilingual inferences must be fired before 2x2 inferences. A monolingual item cannot compose with a bilingual item. This decision was taken so that the translator first selects the terminal sequences of the source sentence and candidate translation that should be translated as a unit, then determines the translationally equivalent units, and then orders the target-side translations. Inferences allowing monolingual items to compose with bilingual items were a part of extension logics, discussed in Section 4.2.3. We found that it was difficult for the model to learn these kinds of inferences.

Axioms Finally, the axioms of the logic are defined as follows. For each word w_i in the input on component d, we create a terminal item $\langle w; d, i\rangle$. For each production allowed by the grammar, we add a grammar term to the logic.

4.2.2 Translation Logic

Our translation logic, Logic T (LT) can be found in Table 4.4. It can represent the same patterns of translational equivalence as Logic MP. However, whereas LMP is constrained by both the source and target components of the bitext, LT is constrained by just the source.

Items Monolingual items that are active on the source component are defined in the same manner as in LMP. However, monolingual items that are active on the target component no longer have spans, since the absolute positions of target terminals are unknown until the translation is complete. The items are represented as follows:

- **Source-side terminal items:** $\langle t, i\rangle$ covers word t at source position i.
- **Target-side terminal items:** $\langle t\rangle$ represents word t on the target component.
- **Non-terminal items:** $\langle X; \sigma\rangle$, with label X and source span σ.

Table 4.2: The specification of Logic MP – Part 1

Term Types	
terminal item	$\langle t; d, i \rangle$
non-terminal item	$\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \rangle$
grammar terms	$\gamma(\cdot, \cdot, \cdot)$
Axioms	
input words	$\langle w; d, i \rangle$ for each word w at position i on component d
grammar terms	as given by the grammar
Inference Rules	
Source Scan	$\dfrac{\langle t; 1, i \rangle, \gamma(t, \varnothing, X)}{\left\langle \begin{bmatrix} X \\ \varnothing \end{bmatrix} ; \begin{bmatrix} i-1, i \\ \varnothing \end{bmatrix} \right\rangle}$
Target Scan	Analogous to Source Scan
Source Fuse	$\dfrac{\left\langle \begin{bmatrix} Y \\ \varnothing \end{bmatrix} ; \begin{bmatrix} i, j \\ \varnothing \end{bmatrix} \right\rangle, \left\langle \begin{bmatrix} Z \\ \varnothing \end{bmatrix} ; \begin{bmatrix} j, k \\ \varnothing \end{bmatrix} \right\rangle, \gamma(\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}, \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix})}{\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \begin{bmatrix} i, k \\ \varnothing \end{bmatrix} \rangle}$
Target Fuse	Analogous to Source Fuse

Inference types The inference types of LT are the same as those of LMP, but with target-side position information removed. LMP renames the target scan inference as a *load inference*.

Axioms This logic replaces the target-side axioms of LMP with a target terminal item for each word in the target vocabulary.

The inferences fired by LMP and LT are those that can be evaluated by grammars in the class of Inversion Transduction Grammars (ITG) [Wu1997]. Alignments allowed by grammars in the class of ITG allow only one-to-one phrase matchings, as do our bilingual composition inferences. Furthermore, ITG allows only for inferences yielding items with contiguous spans, as the inferences in our logics do.

Table 4.3: The specification of Logic MP – Part 2

Inference Rules	
Bilingual Composition	$\dfrac{\left\langle \begin{bmatrix} Y_1 \\ \varnothing \end{bmatrix} ; \begin{bmatrix} i,j \\ \varnothing \end{bmatrix} \right\rangle, \left\langle \begin{bmatrix} \varnothing \\ Z_2 \end{bmatrix} ; \begin{bmatrix} \varnothing \\ k,l \end{bmatrix} \right\rangle, \gamma\left(\begin{bmatrix} Y_1 \\ \varnothing \end{bmatrix}, \begin{bmatrix} \varnothing \\ Z_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \right)}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} i,j \\ k,l \end{bmatrix} \right\rangle}$
Monotonic 2x2 Compositions	$\dfrac{\left\langle \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} ; \begin{bmatrix} i_1,j_1 \\ i_2,j_2 \end{bmatrix} \right\rangle, \left\langle \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} ; \begin{bmatrix} j_1,k_1 \\ j_2,k_2 \end{bmatrix} \right\rangle, \gamma\left(\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}, \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \right)}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} i_1,k_1 \\ i_2,k_2 \end{bmatrix} \right\rangle}$
Inverted 2x2 Compositions	$\dfrac{\left\langle \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} ; \begin{bmatrix} i_1,j_1 \\ j_2,k_2 \end{bmatrix} \right\rangle, \left\langle \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} ; \begin{bmatrix} j_1,k_1 \\ i_2,j_2 \end{bmatrix} \right\rangle, \gamma\left(\begin{bmatrix} Y_1 \\ Z_2 \end{bmatrix}, \begin{bmatrix} Z_1 \\ Y_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \right)}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} i_1,k_1 \\ i_2,k_2 \end{bmatrix} \right\rangle}$

4.2.3 Extension Logics

Our initial attempt to model translational equivalence utilized an improved version of the logic used by [Wellington et al.2006a]. In addition to bilingual inferences and 2x2 composition inferences our *extension logic* used *extension* and *finishing* inferences, and added a flag to each item indicating its *stage*. An item can have one of two stages, *finished* or *unfinished*. A finished item was inferred by a finishing inference. An unfinished item is one that was inferred by a bilingual composition or extension inference. A target extension inference takes an unfinished bilingual item and a monolingual target item and infers an unfinished bilingual item. Thus, target extension inferences take a translation of a source word and insert a word to the left or right to augment the translation. Rather than representing word deletions as translations of a source word to a null token as [Wellington et al.2006a] do, we define source extension inferences to model word deletions. When proposing that a source word should be deleted, the parser/translator fires an inference to make the word to be deleted an extension of an adjacent source word. When 0 or

Table 4.4: The specification of Logic T – Part 1

Term types	
source terminal item	$\langle t; i \rangle$
target terminal item	$\langle t \rangle$
non-terminal item	$\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \rangle$
grammar terms	$\gamma(\cdot, \cdot, \cdot)$
Axioms	
input words	$\langle w; i \rangle$ for each word w at position i on the source component
target language vocabulary	$\langle w \rangle$ for each word w in the target-language vocabulary
grammar terms	given by the grammar
Inference Rules	
Source Scan	$\dfrac{\langle t; i \rangle, \gamma(t, \varnothing, \begin{bmatrix} X \\ \varnothing \end{bmatrix})}{\left\langle \begin{bmatrix} X \\ \varnothing \end{bmatrix} ; \begin{bmatrix} i-1, i \\ \varnothing \end{bmatrix} \right\rangle}$
Target Load	$\dfrac{\langle t \rangle, \gamma(\varnothing, t, \begin{bmatrix} \varnothing \\ X \end{bmatrix})}{\left\langle \begin{bmatrix} \varnothing \\ X \end{bmatrix} ; \begin{bmatrix} \varnothing \\ \varnothing \end{bmatrix} \right\rangle}$
Source Fuse	$\dfrac{\left\langle \begin{bmatrix} Y \\ \varnothing \end{bmatrix} ; \begin{bmatrix} i_1, j_1 \\ \varnothing \end{bmatrix} ; \varnothing \right\rangle, \left\langle \begin{bmatrix} Z \\ \varnothing \end{bmatrix} ; \begin{bmatrix} j_1, k_1 \\ \varnothing \end{bmatrix} ; \varnothing \right\rangle, \gamma(\begin{bmatrix} Y \\ \varnothing \end{bmatrix}, \begin{bmatrix} Z \\ \varnothing \end{bmatrix}, \begin{bmatrix} X \\ \varnothing \end{bmatrix})}{\langle \begin{bmatrix} X \\ \varnothing \end{bmatrix}, \begin{bmatrix} i_1, k_1 \\ \varnothing \end{bmatrix} \rangle}$

Table 4.5: The specification of Logic T – Part 2

Inference Rules	
Target Fuse	$$\frac{\left\langle \begin{bmatrix} \varnothing \\ Y \end{bmatrix} ; \begin{bmatrix} \varnothing \\ \varnothing \end{bmatrix} \right\rangle, \left\langle \begin{bmatrix} \varnothing \\ Z \end{bmatrix} ; \begin{bmatrix} \varnothing \\ \varnothing \end{bmatrix} \right\rangle, \gamma(\begin{bmatrix} \varnothing \\ Y \end{bmatrix}, \begin{bmatrix} \varnothing \\ Z \end{bmatrix}, \begin{bmatrix} \varnothing \\ X \end{bmatrix})}{\left\langle \begin{bmatrix} \varnothing \\ X \end{bmatrix} ; \begin{bmatrix} \varnothing \\ \varnothing \end{bmatrix} \right\rangle}$$
Bilingual Composition	$$\frac{\left\langle \begin{bmatrix} Y_1 \\ \varnothing \end{bmatrix} ; \begin{bmatrix} \tau \\ \varnothing \end{bmatrix} \right\rangle, \left\langle \begin{bmatrix} \varnothing \\ Z_2 \end{bmatrix} ; \begin{bmatrix} \varnothing \\ \varnothing \end{bmatrix} \right\rangle, \gamma\left(\begin{bmatrix} Y_1 \\ \varnothing \end{bmatrix}, \begin{bmatrix} \varnothing \\ Z_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}\right)}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} \tau \\ \varnothing \end{bmatrix} \right\rangle}$$
Monotonic 2x2 Compositions	$$\frac{\left\langle \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} ; \begin{bmatrix} i_1, j_1 \\ \varnothing \end{bmatrix} ; \mathcal{Y} \right\rangle, \left\langle \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} ; \begin{bmatrix} j_1, k_1 \\ \varnothing \end{bmatrix} ; \mathcal{Z} \right\rangle, \gamma\left(\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}, \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}\right)}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} i_1, k_1 \\ \varnothing \end{bmatrix} \right\rangle}$$
Inverted 2x2 Compositions	$$\frac{\left\langle \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} ; \begin{bmatrix} i_1, j_1 \\ \varnothing \end{bmatrix} \right\rangle, \left\langle \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} ; \begin{bmatrix} j_1, k_1 \\ \varnothing \end{bmatrix} \right\rangle, \gamma\left(\begin{bmatrix} Y_1 \\ Z_2 \end{bmatrix}, \begin{bmatrix} Z_1 \\ Y_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}\right)}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} ; \begin{bmatrix} i_1, k_1 \\ \varnothing \end{bmatrix} \right\rangle}$$

more extensions have been generated, a finish inference can be fired to generate a finished item. A left target extension inference looks like:

$$\frac{\left\langle \begin{bmatrix} \varnothing \\ Y \end{bmatrix}; [\ |\tau_2]; \text{unfin} \right\rangle, \left\langle \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}; [\sigma_1|\sigma_2]; \text{unfin} \right\rangle, \gamma(\begin{bmatrix} \varnothing \\ Y \end{bmatrix}, \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix})}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}; [\sigma_1|\tau_2 + \sigma_2]; \text{unfin} \right\rangle}$$

Similarly, a left source extension inference looks like:

$$\frac{\left\langle \begin{bmatrix} Y \\ \varnothing \end{bmatrix}; [\tau_1|\ ]; \text{unfin} \right\rangle, \left\langle \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}; [\sigma_1|\sigma_2]; \text{unfin} \right\rangle, \gamma(\begin{bmatrix} Y \\ \varnothing \end{bmatrix}, \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}, \begin{bmatrix} X_1 \\ X_2 \end{bmatrix})}{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}; [\tau_1 + \sigma_1|\sigma_2]; \text{unfin} \right\rangle}$$

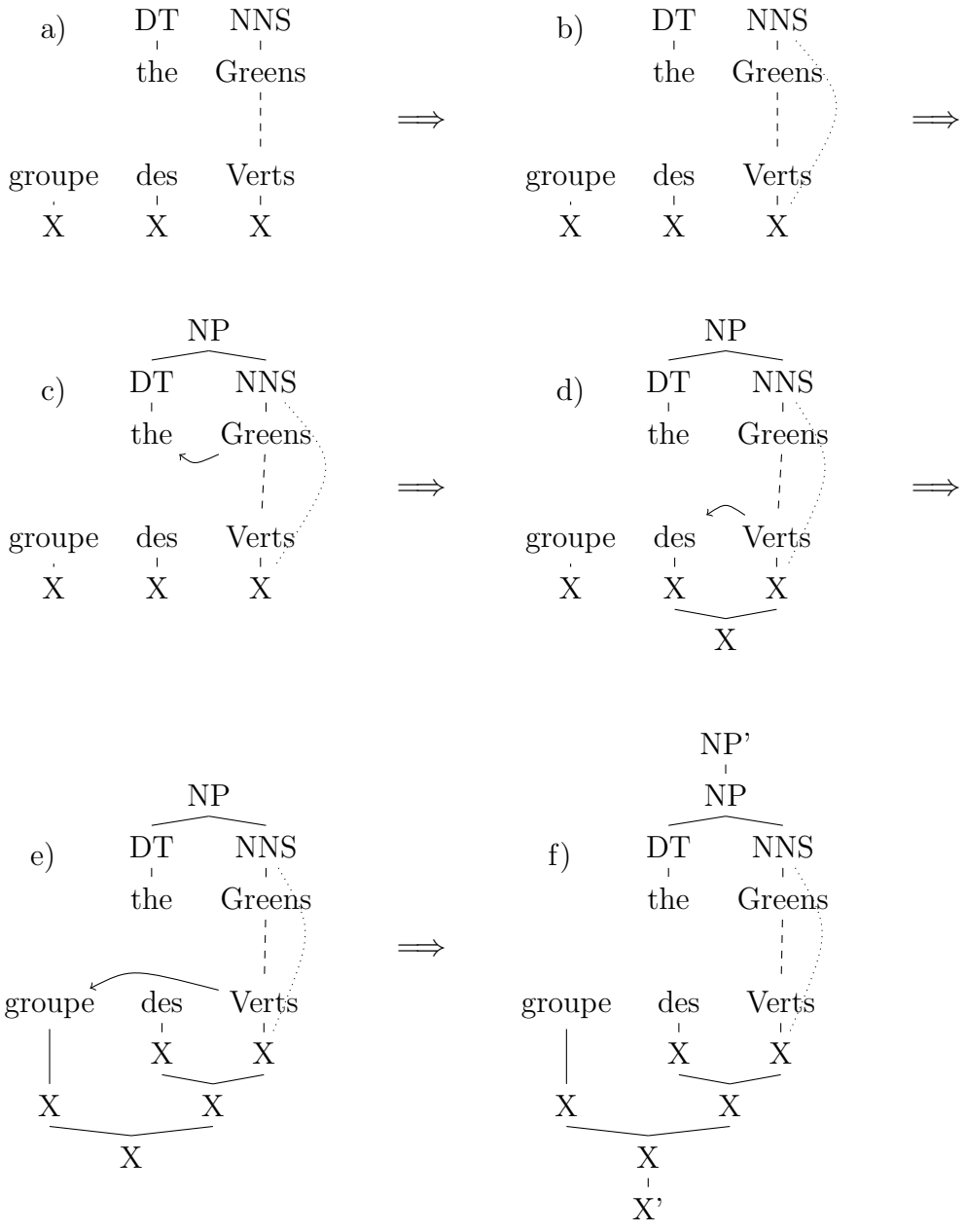
Right target and source extensions are analogous. Finally, a finish inference has the form

$$\frac{\left\langle \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}; [\sigma_1|\sigma_2]; \text{unfin} \right\rangle, \gamma(\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \begin{bmatrix} X_1' \\ X_2' \end{bmatrix})}{\left\langle \begin{bmatrix} X_1' \\ X_2' \end{bmatrix}; [\sigma_1|\sigma_2]; \text{fin} \right\rangle}$$

Bilingual inferences and source/target extensions apply only to *unfinished* antecedent items. 2x2 composition inferences apply only to *finished* antecedent items. The logic cannot extend a finished item. An illustrative example is in Figure 4.1. Diagram a) contains two source terminal items and three target terminal items. In diagram b), the parser fires a bilingual composition inference creating a bilingual item out of "Greens/Verts". In diagram c), the parser proposes a source extension to the left, composing the determiner with the noun. In diagram d), "des" is proposed as a left extension of "Verts", and in e) "groupe" is proposed as a left extension of "Verts". Finally, in diagram f) a finish inference is proposed, finishing the multitree translating "the Greens" to "groupe des Verts."

We expected this logic to work well because of the degree of freedom given by extension inferences. We could have source-extended or target-extended any source/target word pair, and let the model learn which are correct and which are incorrect. In general, our design philosophy was to reduce the number of hard constraints we imposed on the system, and let the model learn the soft constraints necessary to disambiguate between candidates. In practice, however, it was difficult to train classifiers to correctly predict the extensions. It turned out that the most useful features for predicting the cost of an extension inference were the lexical head on the source component of the consequent, its translation, and the lexical yields of all other extensions in the poset of inferences. These lexical features, however, tend to be very sparse and so do not make very strong signals. They

Figure 4.1: A sequence of inferences capped by a finish inference for the pair "the Greens"/"groupe des Verts".



therefore led to many incorrect extension inferences being made, which turned out to be a major source of translation error. The logics we ended up using allowed us to impose harder constraints on the translational equivalence of word sequences, which reduced the entropy of the training set for the relevant inferences.

4.2.4 Item Mapper

Recall from the description of our example generation framework in Section 3.3 that we wish for the coordinator to determine whether an item inferred by the predictor is one that is inferable by the supervisor. Conversely, the coordinator needs to give items proposed by the supervisor to the predictor when the predictor's agenda is empty. To do so, we will need to map items inferable by one logic to items inferable by the other. Therefore we define a mapping function $\phi : \mathbb{I}_A \rightarrow \mathbb{I}_B$ mapping items from Logic A to the items inferable under Logic B. The algorithms underlying the mapping functions are logic-specific. In our case, any item inferable under LMP is inferable under LT by ignoring the target-side spans. The inverse, however is more complicated. Only a subset of items inferred by LT will be in the set of items inferable by LMP. The two mapping functions $\phi_{\mathrm{MP}\rightarrow\mathrm{T}}$ and $\phi_{\mathrm{T}\rightarrow\mathrm{MP}}$ are given in Algorithms 4 and 5.

Conceptually, both algorithms operate recursively. Given an item i inferred as a consequent of inference f under a logic, the mapping function first maps the antecedents of f into items inferable by the target logic. In the case of mapping items from LT to LMP, each item might map to several items in LMP, for instance, if the same word appears in multiple positions in the target sentence. In that case, the mapper finds a set of items that map to i, as in Lines 4-8 of Algorithm 5. If all antecedent items are inferable by the target logic, the mapper finds an inference of the target logic that takes the mapped antecedents and an appropriate grammar production g, and generates the target-logic consequent. For the terminal items that are the base case for the recursions, we map them to the appropriate axioms in the target logic if they exist. Otherwise, we map them to $\varnothing$. In practice we employ the bookkeeping that is necessary to avoid recomputing the mappings of antecedent items.

4.3 Semiring

The Viterbi-derivation semiring [Goodman1999] is commonly utilized in many natural language processing tasks to find the least-cost (or most probable) structured output for an input sentence along with its cost. For probabilistic parsing, the structured output is the most likely parse tree along with its probability (e.g. [Charniak1997]). For probabilistic sequence labeling, such as part-of-speech

Algorithm 4 The mapping function $\phi_{\text{MP}\to\text{T}}$

Input: Item I_{MP} generated by Logic MP,
Inference f such that I_{MP} is the consequent of f
Output: the corresponding item I_{T} inferable under Logic T

procedure $\phi_{\text{MP}\to\text{T}}(I_{\text{MP}}, f)$
  if I_{MP} has the form $\langle t; 1, i\rangle$ **then**
    return $\langle t; i\rangle$
  else if I_{MP} has the form $\langle t; 2, i\rangle$ **then**
    return $\langle t\rangle$
  else
    $Ants_{\text{MP}} \leftarrow \text{antecedents}(f)$
    $Ants_{\text{T}} \leftarrow \{\}$
    for $a \in Ants_{\text{MP}}$ **do**
      $f_a \leftarrow$ the inference such that $a = \text{consequent}(f_a)$
      $Ants_{\text{T}} \leftarrow Ants_{\text{T}} \cup \phi_{\text{MP}\to\text{T}}(a, f_a)$
    $\gamma \leftarrow$ grammar constraint that unifies with set $Ants_{\text{T}}$
    $g \leftarrow \frac{Ants_{\text{T}}, \gamma}{j}$
    return $j = consequent(g)$

tagging, the structured output is the most likely tag sequence (e.g. [Church1988]) and its probability.

In contrast, the elements in the domain of the Viterbi semiring are the costs of the possible structures, without the structure itself. The semiring equation for the Viterbi-derivation semiring returns the least-cost structure and its cost. In a probabilistic task, the domain of the semiring ranges over the values $[0, 1]$. The additive operator is the max operator. The multiplicative operator is the arithmetic multiplication operator. The probability of a partially-ordered set (poset) of inferences is the product of the probabilities of the inferences in that poset. The probability of an item is the maximum of the probabilities of all the inference posets that inferred that item. In the negative log-space, that is, where the domain consists of the negative logs of probabilities, the domain of the Viterbi semiring is $[0, \infty]$. The additive operator is the min operator. The multiplicative operator is the arithmetic addition operator.

The Viterbi-derivation semiring accumulates the cost for a poset of inferences into a single value, leading to a loss of information about which inferences in the sequence had high cost versus those that had low cost. This information might be important for selecting between posets of inferences both during test-time translation and during training set synthesis. For example, consider the two trees in Figure 4.2. Both trees have total cost α according to the semiring, and both have n inferences. All the inferences in tree **a** have cost $\frac{\alpha}{n}$, whereas all but

Algorithm 5 The mapping function $\phi_{\mathrm{T}\to\mathrm{MP}}$

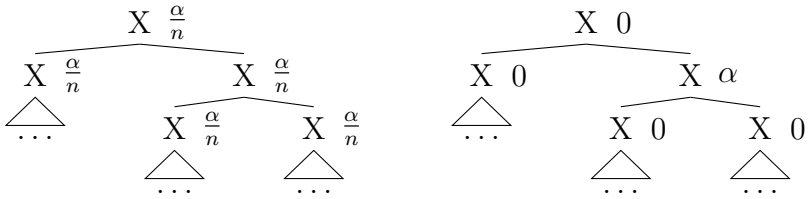
Input: Item I_{T} generated by Logic T,
Inference f such that I_{T} is the consequent of f
Output: the set of corresponding items I_{MP} inferable under Logic MP

procedure $\phi_{\mathrm{T}\to\mathrm{MP}}(I_{\mathrm{T}}, f)$
  if I_{T} has the form $\langle t; i\rangle$ **then**
    return $\langle t; 1, i\rangle$
  else if I_{T} has the form $\langle t\rangle$ **then** ▷ It might map to multiple target positions
    $r \leftarrow \{\}$
    for $i : \langle t; 2, i\rangle \in$ Axioms(Logic MP) **do**
      $r \leftarrow r \cup \{\langle t; 2, i\rangle\}$
    return r
  else
    $Ants_{\mathrm{T}} \leftarrow$ antecedents(f)
    $Ants_{\mathrm{MP}} \leftarrow \{\}$
    for $a \in Ants_{\mathrm{T}}$ **do**
      $f_a \leftarrow$ the inference such that $a =$ consequent(f_a)
      $Ants_{\mathrm{MP}} \leftarrow Ants_{\mathrm{MP}} \cup \phi_{\mathrm{T}\to\mathrm{MP}}(a, f_a)$
    if $Ants_{\mathrm{T}} = \{\}$ **then**
      return $\varnothing$
    else
      $\gamma \leftarrow$ grammar constraint that unifies with set $Ants_{\mathrm{T}}$
      if $\gamma = \varnothing$ **then**
        return $\varnothing$
      else
        $g \leftarrow \frac{Ants_{\mathrm{MP}},\gamma}{j}$
        return $\{j = consequent(g)\}$

one of the inferences in tree **b** have cost 0, and one has cost α. If the costs of the inferences are a good measure of their quality, then tree **b** has one inference of very poor quality, whereas all the inferences in tree **a** have the same quality. For some objectives, the accumulated cost assignment may be very useful, for instance, if the task is to minimize the total cost of a poset or sequence of inferences. For SMT, however, this function might not be ideal. We conjecture that when people evaluate translation quality, a single major error is typically seen as worse than several fairly minor errors. For example, if the system translates a word that is important to capture the meaning of the sentence incorrectly, the single mistranslation might be seen as worse than a few mistakes of, say, gender agreement.

Another problem with using the Viterbi-derivation semiring for SMT is that

Figure 4.2: Two parse trees, parse **a** and parse **b**. Both have total cost α under the Viterbi-derivation semiring.



(a) Parse **a** (b) Parse **b**

posets of inferences for a given source span can have different sizes: an inference poset that yields a translation with fewer words will typically have fewer inferences than a poset that yields a translation with more words. Under this semiring, the translator is biased towards shorter translations. Several other works add a heuristic counter-bias in their cost functions to penalize short translations (e.g. [Koehn et al.2003, Och and Ney2004, Chiang2005]) as a workaround.

To overcome these two problems, we define a semiring that, to our knowledge, is novel. The *MinMerge semiring* (and the corresponding MinMerge-derivation semiring) assigns a sequence of costs to each item y. The sequence contains one element for the cost of each inference in the poset of inferences that derived y, sorted in descending order. Items are compared on their highest differing cost. Given two competing items, the additive operator of the semiring selects the one whose highest cost is the least. In other words, if the values are seen as a measure of the quality of the inference, where higher costs represent lower quality, under the MinMerge semiring, the additive operator will select the item whose inference poset has the least bad worst inference.

Formally, the semiring is defined as follows:

Definition 1. *The MinMerge semiring is a tuple* $\langle \mathbf{D}, \textsc{Min}, \textsc{Merge}, \langle\infty\rangle, \langle\rangle\rangle$ *consisting of:*

- ***Domain*** **D** – *The domain is the set of sequences of real values in descending order.*
- ***Additive identity*** $\langle\infty\rangle$ – *The additive identity is a sequence of length 1 whose single element is* ∞*.*
- ***Multiplicative identity*** $\langle\rangle$ – *The multiplicative identity is the empty sequence.*
- ***Additive operator*** Min – *The additive operator does an element-wise comparison of two sequences* d_1 *and* d_2*. It returns the sequence whose highest*

differing element is the least. If given two sequences such that one is a prefix of the other (i.e. if d_2 contains only the n highest elements of d_1, or vice-versa), the shorter one is returned by the operator. Refer to Algorithm 6 for the details of this operator.

- ***Multiplicative operator*** MERGE – *The multiplicative operator takes two sequences d_1 and d_2 and merges them so that the elements remain in sorted order. Refer to Algorithm 7 for details.*

Theorem 1. *The tuple* $\langle \mathbf{D}, \textsc{Min}, \textsc{Merge}, \langle\infty\rangle, \langle\rangle\rangle$ *is a semiring.*

Proof sketch To prove that the tuple is a semiring, we must prove the semiring axioms:

(a) $(\mathbf{D}, \textsc{Min})$ is a commutative monoid with identity element $\langle\infty\rangle$.

The MIN operator in Algorithm 6 performs an element-wise comparison between the entries in the two sequences. The commutativity of the operator follows from the commutativity of the min function (lines 5-8 of Algorithm 6). To show that for $a \in \mathbf{D}$ $\textsc{Min}(a, \langle\infty\rangle) = \textsc{Min}(\langle\infty\rangle, a) = a$, we note that the single entry in the sequence $\langle\infty\rangle$ is positive infinity, so if there exists a finite value in a, Algorithm 6 will return a. Otherwise, $a = \langle\infty\rangle$, since no sequence in the domain can contain an ∞ other than $\langle\infty\rangle$.

(b) $(\mathbf{D}, \textsc{Merge})$ is a monoid with identity $\langle\rangle$.

There are two properties that must hold. The first is that for $a \in \mathbf{D}$, $\textsc{Merge}(a, \langle\rangle) = a$. $\langle\rangle$ is an empty sequence, and because of lines 12-17 of Algorithm 7, a will be copied to the output. The second is that for $a, b, c \in \mathbf{D}$, $\textsc{Merge}(a, \textsc{Merge}(b, c)) = \textsc{Merge}(\textsc{Merge}(a, b), c)$. This property follows from the implementation of the MERGE operator in Algorithm 7.

(c) The multiplicative operator distributes over the additive operator.
(I.e. for $a, b, c \in \mathbf{D}$, $\textsc{Merge}(a, \textsc{Min}(b, c)) = \textsc{Min}(\textsc{Merge}(a, b), \textsc{Merge}(a, c))$.)

To show that the MinMerge semiring upholds this axiom, let $a = \langle a_1, a_2, \ldots, a_L\rangle$, let $b = \langle b_1, b_2, \ldots b_M\rangle$, and let $c = \langle c_1, c_2, \ldots, c_N\rangle$. Without loss of generality, suppose that $\textsc{Min}(b, c) = b$, and that the LHS of the equality, $\textsc{Merge}(a, b) = \langle a_1, \ldots, b_1, \ldots, a_x, \ldots, b_y\rangle$. We must show that the RHS, $\textsc{Min}(\textsc{Merge}(a, b), \textsc{Merge}(a, c)) = \textsc{Merge}(a, b)$. First, assume that for some $k < M, N$ $b_1, \ldots, b_k = c_1, \ldots, c_k$. The first distinct elements between b and c will be b_{k+1} and c_{k+1}. Since we assumed that $\textsc{Min}(b, c) = b$, we know that $b_{k+1} < c_{k+1}$. Therefore, there exists a prefix $a_1, \ldots, b_k$ of $\textsc{Merge}(a, b)$ and a prefix $a_1, \ldots, c_k$ of $\textsc{Merge}(a, c)$ such that $a_1, \ldots, b_k = a_1, \ldots, c_k$. If, in the sequence returned by $\textsc{Merge}(a, b)$, b_{k+1} follows b_k, then in the sequence returned by $\textsc{Merge}(a, c)$, c_{k+1} follows c_k, meaning that the MIN operation on the right hand side will return $\textsc{Merge}(a, b)$. Otherwise, if some sequence of $a_i, \ldots, a_j$ separates b_k from b_{k+1}, then $a_i \geq c_{k+1} > b_{k+1}$, meaning that the sequence returned by $\textsc{Merge}(a, c)$ will have

the highest distinct element c_{k+1}. The other reason that $\textsc{Min}(b, c)$ could be b is if b is a prefix of the sequence of costs c. That implies that $\textsc{Merge}(a, b)$ is a prefix of $\textsc{Merge}(a, c)$, meaning that $\textsc{Min}(\textsc{Merge}(a, b), \textsc{Merge}(b, c)) = \textsc{Merge}(a, b)$.

(d) Applying the multiplicative operator to the additive identity annihilates $\mathbf{D}$ (i.e. $\textsc{Merge}(a, \langle\infty\rangle) = \langle\infty\rangle$).

This axiom follows from the special case in Line 1 of Algorithm 7 where if one of the arguments of the MERGE operator is $\langle\infty\rangle$, we return $\langle\infty\rangle$. □

The MinMerge-derivation semiring is similarly defined, except that the domain consists of pairs of cost sequences and posets of inferences. The structural component of the MinMerge-derivation semiring is the same as the structural component of the Viterbi-derivation semiring presented in [Goodman1999]:

Definition 2. *The MinMerge-derivation semiring is a tuple*

$$\left\langle \mathbf{D} \times 2^{\mathbb{E}}, \textsc{MinDeriv}, \textsc{MergeDeriv}, \langle\langle\infty\rangle, \varnothing\rangle, \langle\langle\rangle, \{\langle\rangle\}\rangle \right\rangle$$

consisting of:

1. ***Domain*** $\mathbf{D} \times 2^{\mathbb{E}}$ – *The cross product of the set* $\mathbf{D}$ *from Definition 1 and the power set of all inference posets* $\mathbb{E}$

2. ***Additive operator*** MINDERIV – *The additive operator*[1]

$$\textsc{MinDeriv}(\langle d_1, E_1\rangle, \langle d_2, E_2\rangle) = \begin{cases} \langle d_1, E_1\rangle & \text{if } \textsc{Min}(d_1, d_2) = d_1 \\ \langle d_2, E_2\rangle & \text{if } \textsc{Min}(d_1, d_2) = d_2 \\ \langle d_1, E_1 \cup E_2\rangle & \text{if } d_1 = d_2 \end{cases}$$

3. ***Multiplicative operator*** MERGEDERIV – *The multiplicative operator*

$$\textsc{MergeDiv}(\langle d_1, E_1\rangle, \langle d_2, E_2\rangle) = \langle \textsc{Merge}(d_1, d_2), E_1 \cup E_2\rangle$$

following [Goodman1999].

The choice of semiring is closely tied into the training objective, that is, the criteria by which the parsing coordinator selects and labels inferences to train the model. Recall from Section 4.1 that the semiring assigns cost to an item, and thereby to the multitree rooted at the corresponding node. Furthermore, recall from Section 3.1 that the objective towards which the algorithm updates the model is a set of multitrees. Because we utilize a uniform cost search in exploring the search graph (Section 4.4), under the Viterbi-derivation semiring the

[1] In practice, when $d_1 = d_2$ we select between E_1 and E_2 randomly. However, we follow the method of [Goodman1999] and use a union of the sets to ensure associativity of the additive operator.

Algorithm 6 The additive operator MIN of the MinMerge Semiring

Input: sequences d_1, d_2
Output: sequence R

```
1: i ← 1
2: while i ≤ |d1| ∧ i ≤ |d2| ∧ d1[i] = d2[i] do
3:     i ← i + 1
4: if i > |d1| then
5:     return d1
6: else if i > |d2| then
7:     return d2
8: else if d1[i] > d2[i] then
9:     return d2
10: else
11:     return d1
```

predictor, supervisor and test-time translator will explore states in order of the accumulated cost of the multitree rooted at the item representing the consequent state. This means that under the Viterbi-derivation semiring, the items inferred by the predictor (and therefore the inferences inferring them) will be those that have lowest accumulated cost. The inferences that will become part of the training set, therefore, will likely have relatively low cost. The low cost would imply that the predictor's model considers the inferences as inferring multitrees yielding relatively good translations, even if they do not yield the correct translation. The model will be updated to correct the predictor's preferences for these relatively minor errors by training it to assign higher cost to these inferences. On the other hand, under the MinMerge semiring, the predictor will expand items in ascending order of the highest-cost inference in the multitree rooted at the added item. This means that inferences with higher cost will be proposed by the predictor and will become part of the training set. The model will be trained to assign an even higher cost to these incorrect inferences, thereby teaching the translator a further bias against them. In this way, the translator will learn to avoid making these kinds of supposedly egregious errors, rather than expending computation time learning to avoid making relatively minor errors at the expense of larger ones. Furthermore, with the MinMerge semiring, it is easier to do blame assignment to the inferences inferring a poor item, since we can determine which is the offending inference in the poset of inferences. Trying to minimize the highest-cost inference in a poset of inferences is a similar idea to maximizing the minimum margin in the machine learning literature.

One should not confuse the MinMerge semiring with the k-Tropical semirings presented in [Mohri2002]. k-Tropical semirings are used to find k-shortest paths,

Algorithm 7 The multiplicative operator MERGE of the MinMerge Semiring

Input: sequences d_1, d_2
Output: sequence R

if $d_1 = \langle\infty\rangle$ or $d_2 = \langle\infty\rangle$ **then**
    return $\langle\infty\rangle$
$R \leftarrow \langle\rangle$
$i \leftarrow 0, j \leftarrow 0$
while $(d_1[i] > 0 \vee d_2[j] > 0) \wedge i < |d_1| \wedge j < |d_2|$ **do**
    if $d_1[i] \geq d_2[j]$ **then**
        $R \leftarrow R.d_1[i]$ ▷ . is a concatenation operator
        $i \leftarrow i + 1$
    else
        $R \leftarrow R.d_2[j]$
        $j \leftarrow j + 1$
while $i < |d_1|$ **do**
    $R \leftarrow R.d_1[i]$
    $i \leftarrow i + 1$
while $j < |d_2|$ **do**
    $R \leftarrow R.d_2[j]$
    $j \leftarrow j + 1$
return R

and are more closely related to the Viterbi n-best derivation semiring of [Goodman1999].

4.4 Search strategy

The supervisor, predictor, and the test-time translator all utilize a greedy uniform-cost search [Russell and Norvig1995] to explore the search graph. Without pruning, the uniform-cost search strategy finds a least-cost multitree when items are scored using a semiring whose domain is ordered, such as the Viterbi-derivation semiring or the MinMerge-derivation semiring. We order items using an *agenda* (e.g.
[Caraballo and Charniak1998]), sorted on an item's *figure-of-merit*. Following [Wellington et al.2006a], we use the item cost as the figure-of-merit, where the item cost is given by the MinMerge semiring, thereby yielding a best-first search strategy, where lower-cost items are expanded before higher-cost items. In addition to an agenda, the search strategy employs a *chart* to store locally optimal items. The chart consists of cells, in which inferred items are stored based on their spans.

For each item i popped from the agenda, the logic first memoizes i in the chart. It then searches for other chart items (called *matching items*) that, along with i and an appropriate grammar term γ, are antecedent terms to some inference. Once the logic makes an inference and the grammar scores it, the cost of the consequent is computed using the semiring equation, and the consequent is pushed onto the agenda. We present the pseudocode for a generalized parser utilizing a chart and an agenda in Algorithm 8.

Algorithm 8 Concrete Generalized Parsing algorithm

Input: Logic **L**, Grammar **G**, Semiring **R**, Termination Condition **D**, Agenda **A**, Chart **C**, Pruning Strategy **P**, Text tuple **T**

1: **for** each axiom item $w' \in \mathbf{L}$ corresponding to word w in **T** **do**
2: $V(w') \leftarrow \mathbf{1}_R$
3: Push($\mathbf{A}, w'$) ▷ Seed the agenda with the axioms of the Logic
4: **repeat** ▷ Main Loop
5: $x \leftarrow$ Pop($\mathbf{A}$)
6: **if** $\neg$Prunable($\mathbf{P}, x$) **then**
7: Memoize($\mathbf{C}, x$)
8: $M \leftarrow$ MatchingItems($\mathbf{C}, x$)
9: **for** $m \in M$ such that $\exists I = \dfrac{x, m, \gamma}{y} \in \mathbf{L}$ where γ is a grammar term **do**
10: $V(I) \leftarrow V(\gamma)$
11: $V(y) \leftarrow V(y) \oplus (V(x) \otimes V(m) \otimes V(\gamma))$
12: Push($\mathbf{A}, y$)
13: **until** x satisfies **D**

As argued in [Melamed and Wang2004], the worst-case computational complexity of Algorithm 8 under Logic MP or Logic T is $O(n^6)$. To keep the exploration of the search space manageable, we employ two *pruning strategies* to constrain the number of items the parsers expand. One places a hard limit on the number of items stored in each chart cell (a beam-pruning strategy, e.g. [Roark2001, Ratnaparkhi1999]). The other prunes away all items whose cost is greater than some multiple of the cost of the least-cost item in the same cell (e.g. [Burbank et al.2005]). Pruning has the disadvantage of introducing search errors, since an item constituting part of the globally optimal multitree might be pruned because it is not locally optimal with respect to items in the same chart cell.

One can reduce the risk of search error by employing better item cost estimates. A technique that is frequently used is to incorporate an estimate of *outside cost* to the figure-of-merit of an item. For example, [Klein and Manning2003] apply an A* search strategy to parsing by incorporating outside cost estimates that are monotonic upper bounds on the true cost of a final parse tree. In the ITG-constrained word alignment literature, [Zhang and Gildea2005] use IBM Model-1

costs in the outside cost estimates of word alignments falling outside the span being considered, [Haghighi et al.2009] use HMM alignment posteriors as outside cost estimates, and [Cherry and Lin2007] use fixed-link pruning to prune cells for spans that are inconsistent with the initial word alignment. We have not yet explored outside costs in our approach to SMT. We leave that to future work.

4.5 Grammars

The semiring presented in Section 4.3 derives the cost of an item inferred by an inference poset from the cost of each inference in the poset. The inference costs are given by the grammar, which can impose constraints on the inferences that are allowed to fire. If, for instance, we wish to disallow an inference completely, we can have the grammar assign it infinite cost. Grammars are a versatile way to control the inferences made by the parser according to many kinds of criteria.

4.5.1 Composite Grammars

We define a *grammar hierarchy* for use by the supervisor, predictor, and test-time translator, following [Wellington2007]. The grammar hierarchy uses *composite grammars* to incorporate many sources of information. A composite grammar is a mechanism that composes two or more subgrammars, each of which contributes to the cost of an inference. For instance, we can utilize a grammar as a filter that assigns 0 cost to some inferences and infinite cost to all others. Or we can utilize a grammar that allows only the grammar terms that are consistent with a set of word alignments and a monolingual parse tree. We can then score the inferences that received 0 cost using another subgrammar, for instance, a PCFG or a confidence rated binary classifier.

We define two kinds of composite grammars: *additive composite grammars* and *sequential composite grammars*. When an inference is scored by an additive composite grammar, the grammar assigns a cost to the inference that is the sum of the costs of its subgrammars. When an inference is scored by a sequential composite grammar, all but the last subgrammar act as a filter and specify whether the inference is allowed, and the last subgrammar supplies the scoring function for allowed inferences.

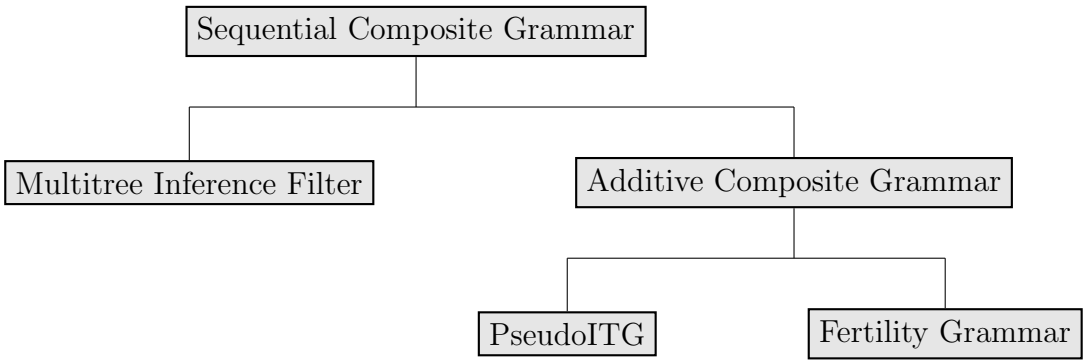
The subgrammars utilized by our composite grammars are:

- a multitree inference filter (MTIF) that filters out inferences that are inconsistent with a given multitree
- a *Pseudo Inversion Transduction Grammar* (or *PseudoITG*)
- an *inference lexicon*

- confidence-rated classifiers
- a *Gaussian fertility grammar* that models the probability $\Pr(\frac{n_t}{n_s}|X)$ that a target word sequence of length n_t is a translation of a source word sequence of length n_s dominated by a nonterminal label X. We discuss the Gaussian fertility model in Chapter 5 when discussing initialization of the translation model.

A sample grammar hierarchy used by a supervisor multiparser is shown in Figure 4.3. Under this grammar hierarchy, an inference is passed into the root sequential composite grammar. The composite grammar first consults the MTIF as to whether the inference is compatible with the MTIF's multitree. If it is, the inference is passed to the additive composite grammar, which adds the cost given to the inference by the PseudoITG to the cost given to the inference by the fertility grammar and returns it. Both the supervisor and predictor use a grammar hierarchy. The example presented here is used by the supervisor before any of confidence-rated classifiers have been parameterized.

Figure 4.3: A grammar hierarchy used by a multiparser in one of our experiments.



4.5.2 Multitree Inference Filter

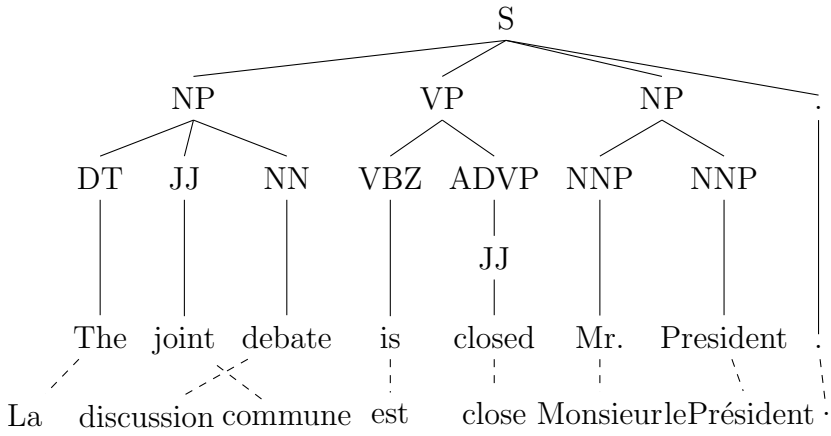
The Multitree Inference Filter (MTIF) is a grammar that contains a multitree for every sentence pair in the bitext. For a given sentence pair, the MTIF assigns infinite cost to any inference not consistent with the multitree and 0 otherwise. We use this grammar to replicate the experiments of [Wellington et al.2006a] in Chapter 5. The MTIF is used by the supervisor only, both for example generation and to constrain the exploration performed by the predictor.

4.5.3 Pseudo Inversion Transduction Grammar

For a given sentence, the PseudoITG allows inferences consistent with a set of monolingual syntactic constraints (possibly on both source and target dimensions) and a set of word alignments. It is a generalization of the grammar used for hierarchical alignment by [Wellington et al.2006a, Wellington2007]. The PseudoITG is so called because it evaluates all the inference types allowed by grammars in the class of ITG, but does not contain any production rules.

The PseudoITG takes two optional parameters: a word alignment and zero, one, or two monolingual parse trees. For instance, suppose the multiparser is parsing the sentence pair in Figure 4.4, and the logic attempts to compose the antecedents $A = \left[\begin{bmatrix}\text{JJ}\\ X^2\end{bmatrix}; [\ 1\ 2 \mid 2\ 3\ ]\right]$ and $B = \left[\begin{bmatrix}\text{NN}\\ X^1\end{bmatrix}; [\ 2\ 3 \mid 1\ 2\ ]\right]$, in inverted order. The PseudoITG will return a cost of 0 for the inference because the composition is consistent with the word alignments. If, however, the logic attempted a monotonic 2x2 composition with the antecedents, the PseudoITG would return infinite cost. Furthermore, if the logic attempts to compose the antecedents $C = \left[\begin{bmatrix}\text{JJ}\\ X\end{bmatrix}; [\ 4\ 5 \mid 4\ 5\ ]\right]$ and $D = \left[\begin{bmatrix}\text{NNP}\\ X\end{bmatrix}; [\ 5\ 6 \mid 5\ 6\ ]\right]$ in either monotonic or inverted order, the PseudoITG would return infinite cost because any composition of the two antecedents would be inconsistent with the source-side parse tree. By default, the PseudoITG gives 0 cost to any allowed inference, and infinite cost to any disallowed inference. The PseudoITG can also be configured to give non-zero cost to allowed inferences. The grammar can take weighted monolingual trees and weighted word alignments as parameters, and compute a cost for each inference from their weights.

Figure 4.4: A sentence pair with source-side monolingual parse tree and word alignment constraints



To support efficiently constraining the inferences to match the monolingual tree constituents, each item has a pointer to the node in the monolingual tree that immediately dominates the span. If that span matches a constituent boundary, the pointer will point to the non-terminal node of that constituent. If it does not match a constituent boundary, then the pointer will point to the nonterminal node of the smallest constituent that dominates the constrained span[2]. In the example above, for instance, antecedent A would contain a pointer to the JJ node dominating "joint", antecedent B would contain a pointer to the NN node dominating "debate", and the consequent of their composition would contain a pointer to the NP node dominating "joint debate".

When used by the predictor and test-time translator, the PseudoITG has only a monolingual parse tree on the source component as a constraint. Since the translator does not know the correct translation it also cannot know the word alignments. The PseudoITG used by the translator serves only to constrain the source side of the resulting multitree.

Unlike [Wellington2007], our use of the PseudoITG does not require the monolingual tree to strictly constrain the set of possible inferences. We include a hyperparameter in the grammar that allows us to specify how many source constituent boundaries (i.e. brackets) a multitree is allowed to violate. We refer to each violation as a crossed bracket, or CB. Doing so is necessitated by the fact that many idiomatic expressions are not valid syntactic constituents. For example, in the Europarl corpus [Koehn2005] the English phrase "we are talking about" is frequently translated as "il s'agit de." "We are talking about" is not a legal constituent according to most off-the-shelf monolingual parsers. However, for translation, it might be useful to generate an item that translates the English word sequence "we are talking about" to "il s'agit de" even though it breaks the constituent boundary. A PseudoITG can assign a cost to crossed brackets, allowing us to bias the supervisor against breaking constituency constraints but still permitting them when needed. Instead of the source-side parse trees acting as hard constraints on the inferences that can be fired, they are instead used as soft constraints. As we will see in Section 4.5.5.2, we add the syntactic constraints as features in our model.

There is evidence in the literature that soft syntactic constraints, rather than hard syntactic constraints, improve translation accuracy. [Marton and Resnik2008] apply syntactic features extracted from a treebank to Hiero [Chiang2005] and achieve accuracy gains over the baseline Hiero system. These features are expanded upon in [Chiang et al.2008]. Soft syntactic constraints appear to be useful in SMT subtasks as well. For instance [Cherry and Lin2007] use dependency features

[2]We can have an item that does not match a span if the monolingual tree is not binary. Because we allow only binary inferences, we might infer an item that is consistent with the monolingual tree, but does not match a complete span (e.g. if a non-terminal node has three constituent children, but the item composes antecedents corresponding to only two of the three).

to improve alignment accuracy in their discriminative word alignment algorithm, rather than a hard constraint on one or both of the components being aligned. To our knowledge, our use of loosely constraining monolingual trees is novel. Our approach combines the advantages of soft syntactic constraints by incorporating them as features as well as the advantage of increased parsing efficiency due to a smaller search space by still imposing a hard upper bound on the number of allowed CBs.

4.5.3.1 Binarization

As demonstrated in Figure 4.4 the monolingual parse tree constraints might not be binary. However, LMP and LT are binary. PseudoITG allows any binary inference consistent with the monolingual parse trees and word alignments. [Wellington et al.2006a] had the constraint that for a parent node with more than two children, the two or more children that were not the head of the parent were not allowed to be antecedents to a composition. A non-head child node can only compose with a node that subsumed the head child node of the parent. Our PseudoITG eliminates this constraint because we observed that it precludes many useful translation inferences from being fired. The PseudoITG does not have a preference between different binarizations

Binarization is not an issue for the other sub-grammars. The classifiers can score inferences regardless of the number of antecedents. The multitrees used by the MTIF should already be binary if they have been inferred under LMP or LT. The fertility grammar and inference lexicons only consider the yields of the consequent of an inference, without reference to the antecedents. Binarization for SMT has been studied extensively in the literature (e.g. [Zhang et al.2006, Wang et al.2007, Xiao et al.2009]). We can experiment with their binarization schemes in future work.

4.5.3.2 Role of PseudoITG in the item mapper

Recall that the item mappers implemented in Algorithms 4 and 5 are passed an inference generated under some Logic A to be mapped onto an inference generated under another Logic B. The algorithms first recursively find the mapping for the antecedent items of the input inference under Logic B. They then determine whether there are any inferences that contain the antecedents that are allowed by the grammar. To illustrate the grammar's role in the mapping function, suppose we have a predictor that passed the item $\left[\begin{bmatrix}\text{NN}^* \\ X\end{bmatrix}; [\ 1\ 3\ |\ ]; \text{commune discussion}\right]$ with antecedents $\left[\begin{bmatrix}\text{JJ} \\ X^2\end{bmatrix}; [\ 1\ 2\ |\ ]; \text{commune}\right]$ and $\left[\begin{bmatrix}\text{NN} \\ X^1\end{bmatrix}; [\ 2\ 3\ |\ ]; \text{discussion}\right]$. The antecedents are first mapped onto antecedents inferable under Logic MP. The mul-

tiparser's grammar then searches for an inference that has the antecedents that is allowed by the PseudoITG. In this case, the only composition allowed by the PseudoITG is an inverted 2x2 composition, which does not match the inference passed to the item mapper. Therefore, $\phi_{\mathrm{T}\to\mathrm{MP}}$ will return $\varnothing$ as the mapped item for $\left[\begin{bmatrix}\mathrm{NN}^* \\ X\end{bmatrix}; [\ 1\ 3\ |\ ]; \text{commune discussion}\right]$ since it is not inferable by the supervisor.

4.5.4 Inference Lexicon

The inference lexicon is a generalization of a translation lexicon or *phrase table* (e.g. [Koehn et al.2003, Och and Ney2004]). It constrains the set of monolingual compositions (both source and target) and bilingual compositions a parser is allowed to make by specifying sets of source and target word sequences that are yields of the antecedent items allowed to participate in these inferences. The inference lexicon is used by the predictor to restrict the hypothesis space it explores, and by the supervisor to ensure that it doesn't propose items that are not inferable by the predictor. An example inference lexicon is given in Table 4.6.

To facilitate fast look-up, the grammar terms of the inference lexicon grammar are keyed on the yields of the antecedent items to an inference. Each grammar term contains an *anchor* and a set of *extensions*. For monolingual compositions, the anchor is the active component yield of the left antecedent and the extension is the active component yield of the right antecedent[3]. For bilingual compositions, the anchor is the active component yield of the source monolingual item and the extension is the active component yield of the target monolingual item (essentially mapping a source fuse to a target one). For each item source or target yield, the inference lexicon stores a count for how many times it was observed in the data used to construct the inference lexicon. This frequency can be used to assign a cost to an inference and for pruning the inference lexicon.

To prune the inference lexicon, we use the following procedure (used by [Wellington2007] for word transduction inferences). For each anchor:

1. Compute the total number of extensions N for that anchor by accumulating the counts for each extension.

2. For each extension, if all of the following hold, we prune the extension:

   (a) the count $c < \frac{N}{5}$

   (b) the count $c < \frac{1}{2}$ of the count of the most frequent extension of the anchor

   (c) the count $c <$ threshold α

[3]We constrain it to use the order of antecedents in this manner to avoid spurious ambiguity in the lexicon.

Table 4.6: A sample inference lexicon for the phrase "we are talking about"

Anchor	Extension	Frequency
	Source Fuses	
we	are	58
we	are talking	3
we	are talking about	2
are	talking	8
are	talking about	5
talking	about	12
	Target Fuses	
il	s'	108
il	s' agit	81
il	s' agit de	11
s'	agit	179
s'	agit de	22
agit	de	22
	Bilingual Compositions	
we are	nous	22
we are	en	3
we are talking	nous parlons	1
we are talking	il s' agit de	2
talking	parler	2
talking	parlons	1

To disallow pruning, we set α to 1, so that no extension will be pruned. In our experiments, we set α to 3 for bilingual compositions, and 1 for source and target fuses (thereby not pruning any monolingual composition terms from the grammar).

4.5.5 Confidence Rated Classifiers

The training set synthesized by the parsing coordinator described in Chapter 3 is used to train the confidence-rated binary classifiers. For all our experiments, it is this subgrammar that is the primary mechanism by which the translator encodes a preference between different inferences and therefore different translations. This subgrammar first uses a feature generator to extract a vector of features from the inference, and then computes a score for the feature vector[4]. The confidence-rated binary classifiers return a real-valued confidence for each inference. The

[4]We partition inferences into types and induce classifiers for different inference types to allow us to train classifiers in parallel. The different classifiers can be seen as subtrees of one big decision tree where we hand-craft the top split of the tree on the inference type.

confidence can be either positive or negative. Rather than assigning a confidence to an inference i, we assign a *cost*, which is the logistic function of the confidence $\mu_\Theta(i)$:

$$V(i) = \ln(1 + \exp(-\mu_\Theta(i))) \tag{4.2}$$

The confidence rated classifier is the only subgrammar whose parameters are re-estimated on line 8 of Algorithm 2. When we refer to the model in this work, unless otherwise noted, we are referring to the confidence rated classifiers.

A classifier $h_\Theta(i)$ is a linear real-valued cost function parameterized by vector Θ of weights for the features of inference i:

$$h_\Theta(i) = \Theta \cdot X(i) = \sum_{f=1}^{|\Theta|} \Theta_f \cdot X_f \tag{4.3}$$

where $X(\cdot)$ is a vector of boolean features for inference i with 1 representing true and 0 representing false. The classifiers are trained by finding the parameter vector Θ that minimizes the value of a *local objective function*[5]. The sign of $h_\Theta(i)$ indicates whether the classifier believes the inference is correct. The magnitude of h_Θ is the confidence of the classifier in its prediction.

4.5.5.1 Gradient Boosted Decision Trees

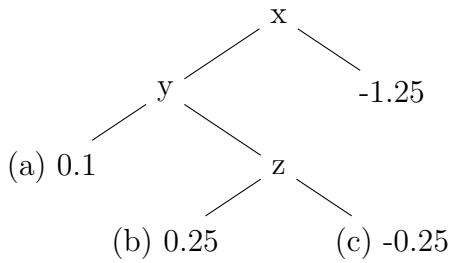
We represent the hypothesis function for each inference type as an ensemble of gradient boosted decision trees (GBDTs), trained using the discriminative approach described in [Turian et al.2006]. In this section, we briefly review GBDTs and their training.

The linear hypothesis function $h_\Theta(i) = \Theta \cdot X(i)$ is represented by an ensemble of confidence-rated decision trees [Schapire and Singer1999]. Each internal node splits examples on whether an example satisfies the atomic boolean feature represented at the node. The path from the root to a leaf in each tree represents a *compound feature*, a conjunction of the atomic features at the nodes on the path. For instance, in the decision tree represented in Figure 4.5, the compound feature represented at node (a) is $x \wedge y$ with parameter value 0.1; the compound feature represented at node (b) is $x \wedge \neg y \wedge z$ with parameter value 0.25; and the compound feature represented at (c) is $x \wedge \neg y \wedge \neg z$ with parameter value -0.25. Each leaf n in each decision tree t is parameterized by a vector $\theta(n)$ where all entries are 0 except for the compound feature terminating at the leaf, so that the parameter vector for each classifier is the sum of these vectors:

$$\Theta = \sum_{t \in T} \sum_{n \in t} \theta(n)$$

[5]We use the term "local objective function" to distinguish the objective of each classifier from the global objective of learning whole paths through the search space. We expand on local versus global objectives in Section 4.8.

Figure 4.5: An example decision tree



To compute the cost of an inference, the inference is percolated down to a leaf of each tree in the ensemble, by following the sequence of nodes in the tree corresponding to features present in the inference's feature vector.

Each classifier is trained to minimize its expected generalization error, which is estimated by the empirical risk objective function $R_\Theta(I)$ over a labeled training set of inferences I. The empirical risk has two terms, a *loss function* $L_\Theta(I)$, and a *regularization term* Ω_Θ.

$$R_\Theta(I) = L_\Theta(I) + \Omega_\Theta \tag{4.4}$$

The loss function is a sum of *biased example-wise loss* functions:

$$L_\Theta(I) = \sum_{i \in I} b(i) \cdot l_\Theta(i) \tag{4.5}$$

where, $b(i)$ is the *bias* of the example, and l_Θ is the log-loss function [Collins et al.2002]:

$$l_\Theta(i) = \ln(1 + \exp(-\mu_\Theta(i))) \tag{4.6}$$

$\mu_\Theta(i)$ is the *margin* of example i:

$$\mu_\Theta(i) = y(i) \cdot h_\Theta(i) \tag{4.7}$$

where $y(i)$ is the label of example i.

The Ω_Θ term is the *regularizer* which penalizes complex models to reduce overfitting. We follow [Turian et al.2006] and use the ℓ_1 regularizer:

$$\Omega_\Theta = \sum_{f=1}^{|\Theta|} \lambda \cdot |\Theta_f| \tag{4.8}$$

ℓ_1 regularized GBDTs have many advantages for natural language classification tasks, including efficient exploration of an exponentially-sized feature space [Ng2004]; the ability to jointly parameterize atomic features, thus capturing atomic feature inter-dependencies; and the ability to model non-linear separability while maintaining most of the simplicity of a linear model.

The λ hyperparameter (or *regularization penalty*) sets the strength of the regularizer. We use the same regularization penalty for all trained classifiers. We select the regularization penalty by translating a tuning set of sentences with several different penalties, and selecting the penalty yielding highest translation accuracy over the set. In pilot experiments, we attempted to set different regularization values for different inference types, namely one regularizer for bilingual composition inferences and one for 2x2 composition inferences. This strategy did not yield significant improvements in translation accuracy, but greatly increased training time because we had to use a grid search instead of a line search. When we were experimenting with the extension logic described in Section 4.2.3 under the Viterbi-derivation semiring, we also attempted to estimate two regularization penalties, one for inferences yielding unfinished consequents, and one for inferences yielding finished consequents. Applying this strategy lead to slightly increased translation accuracy at the expense of increased training time.

The training algorithm for an ensemble is as follows. Given a labeled training set of inferences I with size $|I|$, we initialize the bias of each example $b(i) = \frac{1}{|I|}$. We initialize the regularization penalty λ to a high value, and start constructing trees. On each iteration of the GBDT training algorithm we build a tree (using the method detailed in the next paragraph). We determine whether the tree reduces the risk over the training set (Equation 4.4) by a given threshold. If it does, we add the tree to the ensemble, and re-weight the examples in the training set. Otherwise, we decay λ by some fixed factor η. This procedure continues until $R_\Theta(I)$ converges to a sufficiently small threshold. This approach applies the regularization path method of [Park and Hastie2007] to gradient boosted decision trees.

We normalize the bias of each example to ensure that during training all the classifiers are partitioning the same weight mass regardless of the size of their training set. We found that doing so brings the costs assigned by the classifiers in line with one another. Initially, we attempted to make do without example bias. We observed, however, that the magnitudes of the confidences given by classifiers trained on fewer examples tended to be lower than the magnitudes of the confidences given by classifiers trained on more examples. This effect means that when we had two alternative inferences, and their respective classifiers both assigned negative confidence, the confidence given by the classifier trained on more examples tended to have disproportionately higher magnitude than the classifier trained on fewer examples. Because most of the examples on which the classifiers were trained were negative, more often than not, the highest confidence of an inference within a set of competing inferences was negative. Thus, the classifiers had an inherent bias towards inference types that were not seen as often in training data. We initially attempted to solve this problem by standardizing the confidences. We did so by estimating the standard deviation of costs given by each classifier on a

held out development set, and standardizing the confidences by the standard deviation. There were several complications with this approach, however. We had to deal with standard deviation estimates of 0 (since some classifiers might have only been applied once in the development set). We also had to re-estimate the optimal regularization penalty given the standardized confidences followed by having to re-estimate the standard deviations given the new regularization penalty.

To construct the decision trees we use greedy feature selection. Initially, a root node is constructed in which all training examples fall. If the root is not split, the parameter value it is assigned is the bias term of Θ. The tree construction algorithm selects an atomic feature on which to partition the example set by finding the feature that maximizes the *expected gain*:

$$G_\Theta(I; f) = \max\left(0, \left|\frac{\partial L_\Theta(I)}{\partial \Theta_f}\right| - \lambda\right) \tag{4.9}$$

Such a split creates two new leaves in the tree, one for the subset of examples that satisfy feature f and one for the subset of examples that does not. We then select the confidence $\Delta\Theta(n)$ at each node n that minimizes $R_\Theta + \Delta\Theta(n)(I_n)$, where I_n is the set of examples falling into node n, using a line search.

Features pertain to inferences. Our features can encode information on antecedent items, the grammar term, the consequent item, and the entire source sentence and monolingual parse tree. Since GBDTs do feature selection for us, in contrast to most other work on SMT, all our experiments use the same set of features, which include: language model features, window features, dependency features, production features, and other miscellaneous features.

[Wellington et al.2006a] were able to include some scalar-valued atomic features by decomposing them into binary features. For example, to represent a feature that some word x was within two positions to the left of a constituent, they added two binary features: "word x at position ≥ -1" and "word x at position ≥ -2". Then the training algorithm can select either of the features to split on, depending on whether x at position -1 or at position or -2 had more discriminative power (i.e. greater gain). This approach, however, is limited only to scalar features whose values are discrete and whose range is bounded. They could not, for example, add a language model feature to their feature set, since the values of n-gram probabilities are continuous. Furthermore, pre-encoding the features increases the number of atomic features we would have to define manually. To overcome this limitation, we have implemented the GBDT training algorithm to convert features with scalar values into binary features, by finding the scalar value that partitions the training examples in such a way as to maximize the gain. The node partitions examples on whether the feature value is greater than the value at the split.

4.5.5.2 Features

Language Model (LM) Due to our modification to GBDT training that allows the model to express continuous scalar-valued features, we can incorporate various language model features into our model. Instead of computing smoothed probability distributions over n-grams, we directly utilize the frequencies of n-grams found in a target-language training set. Because the n-gram frequencies are not used in h_Θ (since all the features parameterized by the model are binary), there is no need for normalization – all the model needs to know is whether the n-gram count is above or below a certain value. Doing so removes a potentially costly step from our training pipeline, in that we no longer need to normalize and smooth n-gram counts. We incorporate the following LM features for each inference:

- Frequency of the least frequent unigram, bigram, and trigram in the target yield of the consequent of the inference
- Mean frequency of all unigrams, bigrams, and trigrams in the target yield of the consequent of the inference
- A measure of the "informativeness" of the minimum n-gram counts. The "informativeness" is a measure of how significant the n-gram is relative to its longest constituent $(n-1)$-gram or 1-gram. For an n-gram $(w_1, \ldots, w_n)$ with count $c(w_1, \ldots, w_n)$ the "informativeness" is defined as:

$$i(w_1, \ldots, w_n) = \frac{c(w_1, \ldots, w_n)}{\max(\min(c(w_1, \ldots, w_{n-1}), c(w_n)), \min(c(w_1), c(w_2, \ldots, w_n)))} \tag{4.10}$$

The intuition behind the "informativeness" feature is that we wish to determine the relevance of each n-gram. Knowing the count is not enough. The distribution of n-grams is long-tailed, so that most n-grams, especially higher order ones, will be rare. We can determine how informative an n-gram is by measuring the ratio of its count to the count of its constituent $(n-1)$-gram. To illustrate why, suppose that a consequent yield's least-frequent bigram is "colonne vertébrale". There are 6 instances of this bigram in the language model. There are 7 instances of the unigram "colonne" and 6 instances of the unigram "vertébrale". The bigram is relatively uncommon. When a node splits on the feature, it is likely that the split value will be greater than 6, so that inferences with minimum bigram count less than 6 receive a negative confidence. However, in addition to knowing that the minimum bigram count is 6, we know that the minimum constituent unigram count is also 6. The high ratio of the two provides a signal that despite the low frequency, the yield is still likely to be valid in the target language, since the two words in the bigram co-occur almost as often as they occur separately.

Window features Window features include all the source words and their part-of-speech (POS) tags within a two-word window around the consequent of the inference along with their relative positions. In addition, we include the source words on the left and right ends of the consequent, their POS tags, and their candidate translations.

Dependency features Dependency features express the syntactic dependency structure implied by the lexicalized constituency parse rooted at the consequent of the inference. These features include:

- the source-language head word of the consequent, the target-language candidate translation of the head word, and the POS tag of the head word
- each dependent of the source-language head word, their POS tags, and the nonterminal labels of the items of which they are the head
- the nonterminal label of the maximal projection of the lexical head
- the nonterminal label, POS tag, and lexical head of the parent of the maximal projection
- the lexical heads and nonterminal labels of the siblings of the maximal projection

Production features Production features include the monolexicalized, bilexicalized, and and delexicalized grammar constraint terms utilized in the inference. These are similar to the soft syntactic constraint features of [Marton and Resnik2008].

Miscellaneous features In addition to the features above, we also include the following features: the ratio of source-span length to target-yield length of the consequent; the identity and part-of-speech tag of the last word of the source sentence (which is frequently a punctuation mark); and features denoting the type of inference inferring the antecedents (i.e. source fuse, target fuse, bilingual composition, monotonic 2x2 composition, or inverted 2x2 composition).

4.6 Termination Conditions

There are four termination conditions relevant to this work. The first is when we infer a multitree whose source component covers the input sentence. This is the termination condition for the translator when it's used for tuning the regularization parameter (discussed in Section 4.5.5.1) and when it's used for testing. The

second termination condition is when the predictor infers a multitree that is consistent with the source/target sentence pair in the bitext[6]. This is the termination condition for the predictor as used by the parse coordinator (discussed in Chapter 3). The third termination condition is a timer that terminates a parse when it has gone on for more than a set period of time. In our experiments, we set this time to 30 minutes per sentence. In practice, this termination condition very rarely fired in our experiments. The fourth termination condition is when the supervisor's agenda is empty. This termination condition is satisfied somewhat often during training, because oftentimes the supervisor is not capable of inferring a multitree over a sentence pair due to the constraints of the monolingual parse trees and word alignments utilized by its PseudoITG (refer to the discussion in [Wellington2007] for details). When this happens, we have no way to infer a full parse. Instead, when this occurs we extract a *maximum cover* for the sentence. A maximum cover is a minimal set of items representing the fewest number of trees needed to cover the sentence. When we can extract multiple maximum covers, all of which have the same number of trees, we select the cover whose trees have least cost. We use a maximum cover to get the largest possible multitrees covering the sentence. To extract a maximum cover, we consider all the items inferred by the parser. We order them according to the number of source and target words covered by each one, followed by their cost as a secondary criterion. We greedily select the largest one, and filter out all the other multitrees that overlap it. We then select the next largest one that is consistent with the one already selected, and further filter down the set. We repeat this until we have a set of items covering every word in the sentence pair. If we want to extract a set of maximum covers, we extract one maximum cover, remove its items from the set, extract another maximum cover, and repeat.

4.7 Parsing Coordinator Revisited

By framing translation and multiparsing and therefore the predictor and supervisor as generalized parsing algorithms, we can now fully specify the concrete details of the parsing coordinator described in Chapter 3.

The concrete parsing coordinator algorithm is given in Algorithm 9. First, the agendas of the supervisor and predictor are initialized in lines 2 and 3, and the returned set of examples is initialized to empty in line 4. The main loop in lines 5 – 27 iterates until the predictor has inferred an item satisfying the termination condition of the supervisor or until the supervisor runs out of items to expand. In

[6]The predictor does not have knowledge of the reference. This termination condition, however, is used by the parsing coordinator, which can retrieve the relevant information from the supervisor.

Algorithm 9 Concrete generalized parsing coordinator

Input: Supervisor Logic $\mathbf{L^S}$, Predictor Logic $\mathbf{L^P}$,
Supervisor Grammar $\mathbf{G^S}$, Predictor Grammar $\mathbf{G^P}$,
Supervisor Agenda $\mathbf{A^S}$, Predictor Agenda $\mathbf{A^P}$,
Supervisor Chart $\mathbf{C^S}$, Predictor Chart $\mathbf{C^P}$,
Supervisor Pruning Strategy $\mathbf{P^S}$, Predictor Pruning Strategy $\mathbf{P^P}$,
Supervisor Termination Condition $\mathbf{D^S}$,
Semiring $\langle D, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$,
Item mappers $\phi_{\text{MP}\to\text{T}}$ and $\phi_{\text{T}\to\text{MP}}$,
Exploration threshold ψ.

Output: Training set I

1: **procedure** GenerateTrainingExamples($(\mathbf{s}, \mathbf{t})$)
2: Seed Agenda $\mathbf{A^S}$ with axioms from $\mathbf{s}, \mathbf{t}$, and $\mathbf{G^S}$
3: Seed Agenda $\mathbf{A^P}$ with axioms from $\mathbf{s}$ and $\mathbf{G^P}$
4: $I \leftarrow \{\}$
5: **repeat**
6: **if** $\mathbf{A^P}$.Empty() $\wedge \neg \mathbf{A^S}$.Empty() **then**
7: $x^S \leftarrow$ NextSupervisorItem($\mathbf{L^S}, \mathbf{G^S}, \mathbf{A^S}, \mathbf{C^S}, \mathbf{P^S}$)
8: **if** $x^S \neq \varnothing$ **then**
9: $\mathbf{A^P}$.Push($\phi_{\text{MP}\to\text{T}}(x^S)$)
10: **if** $\mathbf{A^P}$.Empty() $\wedge$ $\mathbf{A^S}$.Empty() **then**
11: **break**
12: $x^P \leftarrow \mathbf{A^P}$.Pop()
13: $x^S \leftarrow \phi_{\text{T}\to\text{MP}}(x^P)$
14: **if** $x^S \neq \varnothing$ **then** ▷ $x^S \neq \varnothing$ iff it is supervisor approved
15: $\mathbf{A^S}$.Push(x^s)
16: $I \leftarrow I \cup$ MakeTrainingExample(x^P, x^S)
17: **if** $d(x^P) < \psi$ **then** ▷ $d(\cdot)$ is the exploration distance of the item
18: **if** $\neg \mathbf{P^P}$.Prunable(x^P) $\wedge\, x^P \notin \mathbf{C^P}$ **then**
19: $\mathbf{C^P}$.Memoize(x^P)
20: $M \leftarrow \mathbf{C^P}$.MatchingItems($\mathbf{L^P}, x^P$)
21: **for** $m \in M$ **do**
22: **if** $\exists \gamma \in \mathbf{G^P} \wedge \exists I = \dfrac{x^P, m, \gamma}{y} \in \mathbf{L^P}$ for some y **then**
23: **if** $y \notin \mathbf{C^P} \wedge y \notin \mathbf{A^P}$ **then**
24: $V(y) \leftarrow \mathbf{0}$
25: $V(y) \leftarrow V(y) \oplus (V(x^P) \otimes V(m) \otimes V(\gamma))$
26: $\mathbf{A^P}$.Push(y)
27: **until** x^S satisfies $\mathbf{D^S}$ or $\mathbf{A^S}$ is empty
28: **return** I

Algorithm 10 Procedure that returns a supervisor item for the predictor to expand

Input: Logic $\mathbf{L^S}$,
Grammar $\mathbf{G^S}$,
Agenda $\mathbf{A^S}$,
Chart $\mathbf{C^S}$,
Pruning Strategy $\mathbf{P^S}$

1: **procedure** NEXTSUPERVISORITEM
2: **while** $\neg\mathbf{A^S}$.EMPTY() **do**
3: $x^S \leftarrow \mathbf{A^S}$.POP()
4: **if** $\neg\mathbf{P^S}$.PRUNABLE$(x^S) \wedge x^S \notin \mathbf{C^S}$ **then**
5: **break**
6: $x^S \leftarrow \varnothing$
7: **if** $x^S = \varnothing$ **then**
8: **return** $\varnothing$
9: $\mathbf{C^S}$.MEMOIZE(x^S)
10: $M \leftarrow \mathbf{C^S}$.MATCHINGITEMS$(x^S)$
11: **for** $m \in M$ **do**
12: **if** $\exists\gamma \in \mathbf{G^S} \wedge \exists I = \dfrac{x^S, m, \gamma}{y} \in \mathbf{L^S}$ for some y **then**
13: **if** $y \notin \mathbf{C^S} \wedge y \notin \mathbf{A^S}$ **then**
14: $V(y) \leftarrow \mathbf{0}$
15: $V(y) \leftarrow V(y) \oplus (V(x^S) \otimes V(m) \otimes V(\gamma))$
16: $\mathbf{A^S}$.PUSH(y)
17: **return** x^S

lines 6 – 11 the coordinator calls NEXTSUPERVISORITEM (listed in Algorithm 10) so that the predictor has some elements on its agenda if its agenda is empty at the beginning of the main loop. If, in line 10, the predictor's agenda is still empty after having called NEXTSUPERVISORITEM in line 7 (i.e. if NEXTSUPERVISORITEM returns $\varnothing$, meaning that the supervisor's agenda is empty as well), the algorithm breaks out of the main loop and returns the example set in line 28. Regardless of whether the coordinator requested an item from the supervisor, it pops the next item from the predictor's agenda, maps it to a supervisor item, and then makes a labeled training example in line 16. The conditional statement in line 17 checks whether the distance between the predictor's item and the last supervisor-approved item is below a certain threshold. If it is and if the predictor does not prune it (line 18), the predictor memoizes the item in its chart in line 19. It then finds matching items in line 20, according to the constraints imposed by the inference types of the logic. For each matching item, it sees whether it can participate as an antecedent

in an inference in line 22. For each legal inference, the item cost is computed in line 25 and the item is pushed onto the agenda in line 26. The NextSupervisorItem (Algorithm 10) first pops items from the supervisor's agenda until it finds one that it has not yet memoized or pruned (lines 2 – 6). It then memoizes the item, finds matching items, and proposes inferences, in lines 11 – 16. In line 17 it returns the item that was popped from the agenda in line 3.

4.8 Global Objectives

The criteria by which we select and label the training examples, or global objective, for the model can be fully determined by the configuration of the supervisor multiparser, the details of the MakeTrainingExample function (line 16 of Algorithm 9), and the amount of exploration the predictor is allowed to perform. The configuration of the supervisor determines the hidden correspondence structure $\mathbf{c}$ that the algorithm tries to learn. The details of MakeTrainingExample determine how we label the training examples output by the coordinator. The exploration distance we specify determines the number and type of incorrect inferences the model will be trained to avoid.

4.8.1 Supervisor Configuration

In the experiments of [Wellington et al.2006a] the authors update towards a fixed multitreebank over the training corpus by labeling all inferences contained in the multitree as positive examples, and all other ones generated by their example generation algorithm as negative examples. Doing so limits the generalization of the model. We can decide whether or not to commit to a particular multitree by configuring the supervisor with different grammars. To commit to a single multitree as [Wellington et al.2006a] do, we can apply a Multitree Inference Filter (Section 4.5.2) so that the only translator inferences approved by the supervisor are those comprising the multitree. We refer to this kind of supervision as a *single tree update* strategy. [Liang et al.2006] also commit to updating towards a single hidden correspondence structure, in their case a word alignment, on each iteration of training, but the alignments change across iterations according to the preference of each iteration's model. Since a sentence pair may have many possible correspondence structures (i.e. multitrees), we may not want to update towards a single fixed multitree. Instead, we may wish to label all inferences that may be part of the *multiforest* covering the source sentence/reference translation pair as positive examples and include them in the inference lexicon. By directing the model to learn a set of possible multitrees for each bitext sentence pair, during test time the model can decide which one it prefers. To implement this update strategy, the only configuration we need to change is to remove the Multitree Inference Filter from

the supervisor's grammar hierarchy so that it can utilize any inference consistent with the source-side monolingual parse tree and the word alignments – constraints implemented by the PseudoITG. We refer to this strategy as a *forest update*. [Mi et al.2008] also utilize a multiforest in training their translation model, from which they estimate the statistics for the sub-models in their mildly discriminative model. The structures that the supervisor is constrained to infer determine the *density of the search space* of the predictor. When the supervisor is constrained to infer a single multitree, the density of the predictor's search space is low, since the supervisor can guide the predictor only with the set of inferences predicting the single tree, and since the inference lexicon will have few elements relative to the inference lexicon induced using a multiforest. When the constraint is removed, the density becomes much higher.

4.8.2 Example Labeling

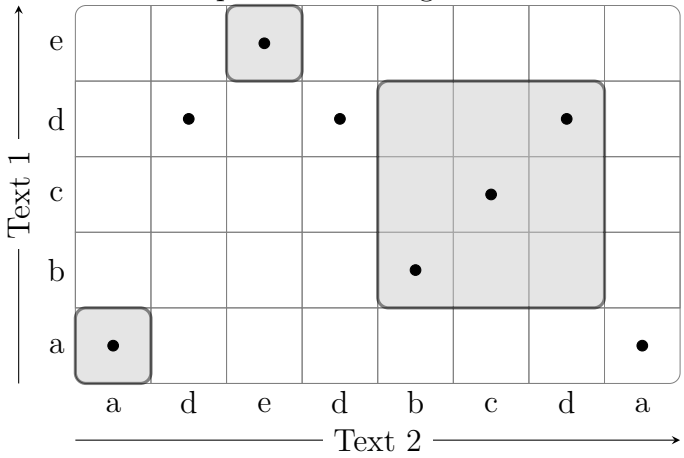
The scheme used to label training examples also affects what the model will learn. One labeling scheme, the *strict labeling strategy*, labels every inference approved by the supervisor as a positive example, and all others as negative examples. In contrast, the *permissive labeling strategy* attempts to train the model to recover from mistakes. Both labeling strategies can be used with either the single tree update or forest update strategies described in the previous section. We refer to these as *single tree strict update*, *forest strict update*, *single tree permissive update*, and *forest permissive update*.

Strict labeling The strict labeling strategy is analogous to the "bold update" strategy of [Liang et al.2006]. If we conceptualize it in terms of our parsing coordinator, this strategy treats every inference approved by the supervisor as correct, and all other inferences fired by the predictor as incorrect. A major disadvantage of the strict labeling strategy is that the translator cannot learn to recover from mistakes. To illustrate, suppose we have a set of items X, all of which cover the same source side span σ, and a set of items Y all of which cover the same source side span τ such that τ is adjacent to σ. Suppose the target yield of a single $\hat{x} \in X$ and $\hat{y} \in Y$ exactly matches part of the reference, and that the yield of the composition of $\hat{x}$ and $\hat{y}$ does as well. Also, suppose that $\exists \dot{x} \in X$ where $\dot{x}$ such that $\dot{x} \neq \hat{x}$, that $\dot{x}$ is not approved by the supervisor, and that $V(\dot{x}) << V(\hat{x})$. Furthermore, suppose that $\hat{x}$ falls outside the beam width and is pruned out. Under the strict labeling strategy, any inference that takes $\dot{x}$ as an antecedent will receive label -1. Because none of the subsequent inferences will be consistent with the fixed multitree, the model makes no distinction between composing $\dot{x}$ with $\hat{y}$ and composing it with any other $y \in Y$. The model may assign a high cost to the composition of $\dot{x}$ with $\hat{y}$, and a lower cost to the composition of $\dot{x}$ with some

other y, so that the latter inference becomes part of the final multitree, leading to a decrease in translation quality. In other words, as soon as the translator makes an incorrect inference, it may have too little information to determine which of the subsequent inferences yield better translations. Under the strict labeling strategy, they are all equally bad.

Permissive labeling To learn to recover from mistakes we shall define an objective function that measures the change in translation accuracy of a consequent relative to that of its antecedents. If the quality of the consequent is better than that of its antecedents, we label the example as positive. If the quality of the consequent is worse than at least one of the antecedents, we label the example as negative. Otherwise, if there is no change in translation quality of the consequent relative to its antecedents, the parsing coordinator does not include the inference in the training set.

Figure 4.6: An example of a bitext grid with hits and runs



Before we explain our approach, we first detail the primary automatic evaluation measures we use in our experiments, an application of precision, recall, and f-measure for SMT [Melamed et al.2003]. Precision and recall are defined in the standard way for translation $\mathbf{t}$ and reference $\mathbf{r}$: $\textsc{Precision}(\mathbf{t}|\mathbf{r}) = |\mathbf{t} \cap \mathbf{r}|/|\mathbf{t}|$ and $\textsc{Recall}(\mathbf{t}|\mathbf{r}) = |\mathbf{t} \cap \mathbf{r}|/|\mathbf{r}|$. By appropriately defining the intersection operator, we determine how the translation is evaluated. For unigram measures, the intersection operator counts the number of matching words called *hits* in $\mathbf{t}$ and $\mathbf{r}$ with no double-counting of hits. For higher-order n-gram measures, the intersection operator counts the weighted number of matching *runs*, a continuous diagonal sequence of hits. Each run can be represented as a block in a bitext grid. The weight of each run is proportional to the area of the smallest enclosing block (refer to Figure 4.6). The weight of all runs r in the bitext grid M is given by the *maximum match*

score (MMS):

$$\text{MMS}^{\alpha}(\mathbf{t}|\mathbf{r}) = \sqrt[\alpha]{\sum_{r \in M} |r|^{\alpha}} \tag{4.11}$$

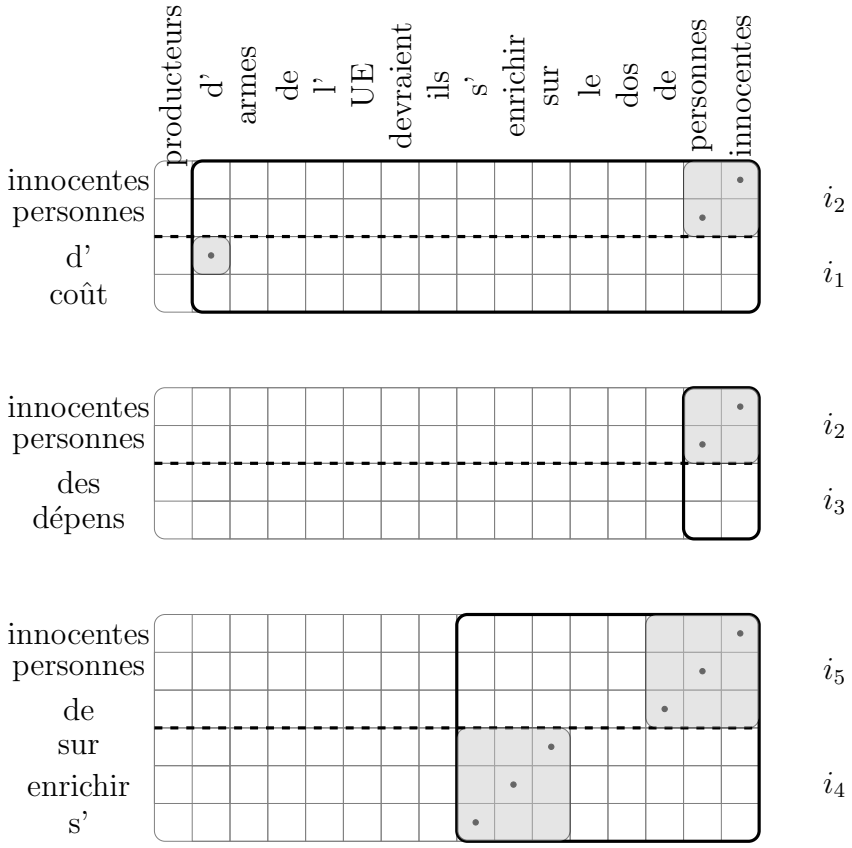
Precision and recall are given by: $\textsc{Precision}^{\alpha}(\mathbf{t}|\mathbf{r}) = \text{MMS}^{\alpha}(\mathbf{t}|\mathbf{r})/|\mathbf{t}|$ and $\textsc{Recall}^{\alpha}(\mathbf{t}|\mathbf{r}) = \text{MMS}^{\alpha}(\mathbf{t}|\mathbf{r})/|\mathbf{r}|$. The F^{α} measure is the harmonic mean of $\textsc{Precision}^{\alpha}$ and $\textsc{Recall}^{\alpha}$. By setting the exponent α to a value greater than 1 the measure rewards longer matching substrings. When $\alpha = 1$, the precision and recall are equivalent to bag-of-words precision/recall measures.

As a motivating example, suppose that the training corpus contains the sentence pair "So why should EU arms producers profit at the expense of innocent people ?"/"Pourquoi donc les producteurs d' armes de l' UE devraient ils s' enrichir sur le dos de personnes innocentes ?". Suppose that the predictor has inferred a set of items with the following source and target yields: i_1 = ["expense of","coût d' "], i_2 = ["innocent people","personnes innocentes"], i_3 = ["expense of","dépens des"], i_4 = ["expense of","s' enrichir sur"], and i_5 = ["innocent people","de personnes innocentes"]. No composition of these items can yield an item with the gold standard reference "sur le dos de personnes innocentes", which is the correct (but non-literal) translation of "expense of innocent people". This can happen due to a word-alignment error in which "expense" is aligned to "enrichir" rather than "dos". Under the strict labeling strategy, all compositions involving these antecedent items are marked as negative examples, and we would get no positive examples for learning how to translate "expense of innocent people."

To elaborate on the example, suppose that the training algorithm considers three compositions in particular: the monotonic composition of i_1 and i_2, inferring consequent with yields c_1 =["expense of innocent people","coût d' personnes innocentes"]; the monotonic composition of i_3 and i_2, inferring consequent with yields c_2 =["expense of innocent people", "dépens des personnes innocentes"]; and i_4 and i_5, inferring consequent with yields c_3 =["expense of innocent people","s' enrichir sur de personnes innocentes"]. The bitext grids for these items are shown in Figure 4.7. The target yields of i_1 and i_3 each have a single hit against the reference. The yield of i_2 has two hits, and a run of length 2, and the yields of i_4 and i_5 each have three hits and runs of length 3. The precision and recall measures use the sum of weights of the runs, that is, the MMS, to compute the quality of the output. Higher MMS values indicate closer matches to the reference. According to this criterion with $\alpha = 2$, of the three translations c_3 has highest accuracy, with a run weight of 4.2426, followed by c_1 with run weight 2.2361, and c_2 as the worst translation with run weight 2.

Ideally, when we have sets of competing inferences, we would compute the F-measure of each output with respect to the reference, select the one(s) with

Figure 4.7: Matching runs of translation candidates against the reference translation. Bold outlined boxes represent the portion of the reference used for computing the MMS. Grey boxes represent runs, and dots represent hits.



highest score as the positive examples, and the ones with lowest as negative[7]. For models such as the ones in our setup, however, doing so is not feasible, because we have no clear way to construct sets of competing inferences. In the case we just illustrated, the set is clear. However, if the translator also proposes an inference with a consequent covering the source text "at the expense of innocent", it is no longer clear which inferences belong in the competing set.

To take full advantage of permissive labeling, we need a *whole inference quality*, or WIQ criterion, a measure that takes into account not only the quality of the consequent, but also the quality of its antecedents. The measure we utilize should: 1) reward compositions that concatenate consecutive runs; 2) penalize inferences composing runs that are not in the same order as seen in the reference; 3) penalize the consequents in proportion to the length and number of non-matching token

[7]i.e. The update strategy undertaken by global models, such as [Liang et al.2006, Chiang et al.2008]

sequences they propose; and 4) penalize the consequents in proportion to the length and number of reference token sequences that are not in the consequent's yield. This last criterion is difficult to evaluate, because we need to decide which parts of the reference translation the output translations are missing. Since we have no *a priori* knowledge of the optimal segmentations of the sentence pair, there are no partial reference translations we can compare against.

We choose to give each item the benefit of the doubt, by comparing it against the subsequence of the reference translation that minimizes the number and length of missing runs. For example, to evaluate item i_1 in Figure 4.7, we compare it only against the word "d' " in the reference. We treat "coût" as an incorrect word proposed by the translator, and assume that the translator did not miss any word sequences in the reference. For item c_3, we compare against the subsequence "s' enrichir sur le dos de personnes innocentes". Because our logic does not allow discontinuous constituent items, there is no subsequent composition that will allow us to propose an item yielding that string. Therefore, we mark the item as having missed a reference subsequence of length 2. If later on we compose c_3 with another item that has the target yield "le dos", the consequent of that composition will not have any missing reference runs, although it should incur a penalty for having a run that is out of order with respect to the reference.

To measure the quality of consequents relative to the quality of the antecedents in an inference, we define a measure called WIQ1 for inference $f = \frac{a_1, a_2, \gamma}{c}$:

$$e(i|\mathbf{r}) = \sqrt[\alpha]{\sum_{r \in M_{\mathbf{t}\cap\mathbf{r}}} r^\alpha} - \frac{\sqrt[\beta]{\sum_{\bar{r} \in M_{\mathbf{t}}} \bar{r}^\beta} + \min_{M_{\mathbf{r}}} \sqrt[\gamma]{\sum_{\bar{r} \in M_{\mathbf{r}}} \bar{r}^\gamma}}{2} - \chi^\delta \tag{4.12}$$

and

$$\text{WIQ1}(f|\mathbf{r}) = e(c|\mathbf{r}) - \max(e(a_1|\mathbf{r}), e(a_2|\mathbf{r})) \tag{4.13}$$

where: $M_{\mathbf{t}\cap\mathbf{r}}$ is the set of matching runs between output and reference; $M_{\mathbf{t}}$ is the set of runs in the output translation that do not match any subsequence of the reference; $M_{\mathbf{r}}$ is the set of non-matching runs in the reference such that $e(i|\mathbf{r})$ is maximized; and χ is the number of crossed alignments. In this work, $\alpha = \beta = \gamma = 2$, and $\delta = 1$. As a base case, monolingual items get e score of 0, so as to ensure that bilingual items will be added to the training set.

The number of crossed alignments is the minimum number of runs that are not in the correct order relative to the reference. WIQ1 uses MMS as the first term (Equation 4.11), which rewards the yield of each item for each matching run it has, with a higher reward for longer matches compared to shorter matches. WIQ1 penalizes the item for each run in its yield that does not match the reference and for each run in the reference that does not match the item's yield. The last term

imposes an additional penalty for crossing matching runs relative to the reference. The other terms do not use this information. Suppose we are given a reference xay, where x, a, and y are all word sequences. Suppose that a consequent has a target yield xy and is missing the sequence a. That consequent would have the same e score as another consequent with target yield yx if the WIQ1 measure did not have a χ term, meaning that both consequents would have the same WIQ1 score, when one of the consequents is better than the other. When dividing the weights of non-matching candidate and reference runs by 2, we average over the non-matching runs in the candidate and the reference.

Returning to the example in Figure 4.7, c_1 will be added as a negative example because $(e(c_1|\mathbf{r}) = -4.26393) - \max(e(i_1|\mathbf{r}) = 0.5, e(i_2|\mathbf{r}) = 2) < 0$, c_2 will be added as a negative example because $(e(c_2|\mathbf{r}) = 1) - \max(e(i_3|\mathbf{r}) = -1, e(i_2|\mathbf{r}) = 2) < 0$, and c_3 will be added as a positive example, since $(e(c_3|\mathbf{r}) = 3.24264 - \max(e(i_4|\mathbf{r}) = 3, e(i_5|\mathbf{r}) = 3) > 0$.

The permissive labeling strategy can label inferences inferring the same consequent differently for different sets of antecedents. For example, consider a consequent item whose target yield is "sur Union européenne". If the reference translation is "sur l' Union européenne," and if the yields of the antecedent items of the inference are "sur" and "Union européenne", the inference will be labeled as a negative example, since $e(\text{"sur Union européenne"}|\mathbf{r}) = \sqrt{1^2 + 2^2} + \frac{1+0}{2} + 0 = 1.736$, and $e(\text{"sur"}|\mathbf{r}) = \sqrt{(1^2)} + \frac{0+0}{2} + 0 = 1$ and $e(\text{"Union européenne"}|\mathbf{r}) = \sqrt{(2^2)} + \frac{0+0}{2} + 0 = 2$. However, if the yields of the antecedent items are "sur Union" and "européenne", the inference will be labeled as a positive example, since $e(\text{"sur Union"}|\mathbf{r}) = \sqrt{(1^2 + 1^2)} + \frac{1+0}{2} + 0 = 0.914$ and $e(\text{"européenne"}) = \sqrt{(1^2)} + \frac{0+0}{2} + 0 = 1$.

Initially, we attempted to use F^2-measure directly as a WIQ measure. Doing so was challenging because it was unclear how to normalize the MMS. We tried normalizing the precision using the length of the output translation and normalizing the recall using the length of the reference. However for most inferences the precision and recall measures were unbalanced since their consequents did not cover the entire source sentence. The lack of balance tended to degrade translation quality by adding too many positive examples to the training set, because the consequent could never have lower recall than the antecedents. We then attempted to compute the recall of the yield of each consequent with respect to the same reference we used to compute WIQ1, however we found that doing so ended up not giving us enough positive examples. We also experimented with other WIQ functions to compare the e value of the consequent to those of its antecedents. We tried using the average e-value of the antecedents rather than the maximum which led to a too-permissive labeling strategy.

Using whole inference quality is critical for learning to search. Using a WIQ, not necessarily WIQ1, allows the translator to learn whether making a particular inference improves or damages the quality of the resulting translation. The permis-

sive labeling strategy utilizing WIQ gives the learning algorithm some information about which incorrect inferences[8] are better than others, potentially giving the model more discriminative power when the translator finds itself in an incorrect state. To our knowledge, we are the first to utilize such a criterion in learning to search hypergraphs.

The example labeling strategy should not be confused with exploration of the search space. Exploration refers to the inferences generated by the predictor that the model can use to learn. The labeling strategy refers to determine whether we label an inference as a positive or negative example for classifier training. Exploration and the labeling strategy do interact. With a strict labeling strategy, increasing the amount of exploration adds only negative examples for training. We conjecture that with very high amounts of exploration, the models will lose discriminative power due to being overwhelmed with negative examples. Under the permissive labeling strategy, however, allowing more exploration might not negatively impact the quality of the model. Because some of the "incorrect" examples will be positive, the learning algorithm has some information as to the relative quality of examples, allowing the algorithm to determine whether an inference helps or hurts when the translator is unable to infer the reference translation.

4.9 Comparison to other work on tree-structured SMT

Tree-structured machine translation has been studied extensively in the research literature. [Yamada and Knight2001, Yamada and Knight2002] presented a generative model that transforms a target language tree into a source language sentence. [Alshawi et al.2000] built a finite-state transducer model over dependency trees, with a transducer for each head node in the source dependency tree. [Graehl and Knight2004] formalized the use of tree transducers for tree-structured MT and present an algorithm for parametrizing stochastic tree transducers (e.g. [Eisner2003]). They presented an EM algorithm for parameterizing their STSG model based on the inside/outside algorithm [Baker1979]. The tree transducers of [Alshawi et al.2000, Graehl and Knight2004] operate under a top-down logic. Data-Oriented Translation [Poutsma2000, Hearne and Way2003] employs tree-to-tree translation by parameterizing a maximum-likelihood synchronous tree-substitution grammar (STSG) from a bitext whose source and target components are parsed.

[Chiang2005] presents a hierarchical phrase SMT system that, like our parsers, is a bottom-up synchronous CFG parser with beam pruning. They present heuristic methods to extract hierarchical phrase rules from a word-aligned bitext. They

[8]Incorrect inferences are those that do not infer a contiguous subsequence of the reference translation linked to the correct source segment.

use maximum likelihood estimation to parameterize the phrase rules, and apply MERT to further weight the MLE parameters. While tree-structured, their model makes no use of a treebank or any other linguistic annotations in training their model. The SCFG rules they extract combine terminals and nonterminals on the RHS, allowing them to model discontinuous phrases, such as translating the English "not X_1" to French "ne X_1 pas", unlike our system which does not yet allow for discontinuous items. [Chiang et al.2008, Chiang et al.2009] augment the model of [Chiang2005] with additional features, including source and target-language syntactic features, source word context features, node count features, and word insertion features, bringing about an improvement in translation accuracy, made possible by adopting MIRA as their training algorithm.

In contrast to [Chiang2005], [Galley et al.2004] present a heuristic method to extract SCFG rules from bitext with one component parsed. Just like the rules extracted by Chiang, [Galley et al.2004] give SCFG rules that combine terminals and nonterminals on the right hand side, allowing for their system to model discontinuous constituents. [Galley et al.2006] build upon [Galley et al.2004] by constructing and parameterizing what they call composed rules from the "minimal rules" of [Galley et al.2004]. They then apply these composed rules for translation, and show that they lead to substantial improvement in translation accuracy over minimal rules, since by translating using larger tree fragments, their rules capture more of the context, at the expense of a significantly larger grammar and possibly fragmenting the training data. By incorporating composed rules into their model, they are able to model phrase translations that are not constrained by a treebank. The models of [Galley et al.2004, Galley et al.2006] both translate a source-language string into a target-language tree.

The tree-to-string alignment template of [Liu et al.2006] is similar to [Galley et al.2004], except where [Galley et al.2004] constructs the rules from a target-language treebank, they construct SCFG rules from source-language trees, putting it more in line with the tree transduction approach of [Graehl and Knight2004]. Just as [Galley et al.2006] construct string-to-tree models with tree fragments, [Huang et al.2006] construct tree-to-string models with tree fragments rather than "minimal rules." They learn two separate models: one for phrase translations and one for SCFG rules learned from the treebank. [Mi et al.2008] notice that the fact that the tree-to-string systems translate a single tree leaves the translator susceptible to source-side parse errors, as well as underspecification and uncertainty in the source-language syntactic model of the monolingual parser. They present a forest-to-string model where the input to the translator is a packed forest (e.g. [Huang2008] and references therein) allowing them to learn from and translate using many possible parses over the source sentence. Conceptually, our PseudoITG is most similar to the forest-to-string model of [Mi et al.2008], in that the PseudoITG allows us to infer multitrees over an implicit forest of source-side

parses.

Finally, some works have attempted to build synchronous grammars that incorporate both source and target-language trees. These tree-to-tree models are synchronous CFGs or tree substitution grammars constructed from word aligned bitext constrained by both source and target parse trees. [Cowan et al.2006] borrow the concept of extended projections from the lexicalized tree adjoining grammar (LTAG) formalism and define and parameterize a discriminative model mapping source language parse trees to aligned extended projections. They parameterize the model using the averaged perceptron algorithm [Collins2002]. [Zhang et al.2008] learn a synchronous tree-sequence-substitution grammar (STSSG) from bitext with both source and target components covered by parse trees, where entire tree sequences are jointly substituted at nonterminal nodes, rather than independently as in STSGs. They then apply a CKY-style parsing algorithm with beam search to translate. [Liu et al.2009] note that the disadvantages of 1-best tree-to-string translations are exacerbated in tree-to-tree translation, since issues such as parser errors, underspecification, and uncertainty now apply to both source and target languages. They present a method to extract rules from packed-forests on both source and target languages, following [Mi et al.2008]. [Chiang2010] present a fuzzy tree-to-tree extraction method that commits to single parses on both source and target sentences, but allow tree substitutions at any nonterminal node, rather than matching nonterminal nodes. This approach to substitutions allows them, for instance, to substitute a subtree rooted at nonterminal VBZ (present-tense verb) at a site labeled VBD (past-tense verb) in another tree.

Synchronous dependency models have been applied to tree-structured MT as well. [Ding and Palmer2005] utilize a probabilistic synchronous dependency grammar in their tree transducer where tree nodes are individual words. [Quirk et al.2005] apply synchronous dependency treelets rather than the dependency relations of [Ding and Palmer2005], thus modeling both continuous and discontinuous phrases. During translation, they synchronously parse the input bottom-up, where the partial hypotheses are constrained by a dependency parse.

Synchronous grammars of various types have been used for MT related subtasks as well. Synchronous grammars have been widely used in word and phrase alignment. Stochastic inversion transduction grammar (ITG) [Wu1997] has been used to constrain the search space for word and phrase alignment [Zhang and Gildea2005, Cherry and Lin2007, Haghighi et al.2009, Liu et al.2010, Pauls et al.2010, Huang et al.2011].

Many of the SMT systems described above can be viewed as configurations of our system. The translator of [Yamada and Knight2001] can be implemented as a generalized parser with an appropriate logic, using a PseudoITG with target-side monolingual trees. [Poutsma2000, Hearne and Way2003] can be implemented as a generalized parser using a generalized multitext grammar [Melamed et al.2004]

due to a mapping they use between treelets and SCFG rules. The system of [Chiang2005] can be implemented by a generalized parser with a logic allowing discontinuous constituents and a PseudoITG unconstrained by source or target monolingual parse trees. The approach of [Galley et al.2004, Galley et al.2006] can be implemented using a generalized parser using the same configuration as the parser for the approach of [Chiang2005], except that the PseudoITG should be configured with monolingual parse trees.

The use of a multiparser and a translator to infer translation forests is also found in the literature on using latent variable models for discriminative SMT. [Blunsom et al.2008] populates two charts, one for trees that are constrained by both source and reference sentences, and one for trees that are constrained only by source sentence. Unlike [Blunsom et al.2008], we utilize the supervisor and predictor for more than populating their respective charts. Our architecture allows the coordinator to use the supervisor to guide the predictor's inference process. Furthermore, the coordinator can use the supervisor to limit the amount of exploration performed. Finally, in order to determine whether a predictor's inference is approved by the supervisor, the supervisor need not have memoized a corresponding consequent item first. All it has to do is to check whether its logic and grammar admits the inference.

Chapter 5

Experiments

In this chapter we describe our experiments. We first describe our datasets, and the independent variables whose settings give rise to the five systems we train. We then present various measures of accuracy of the translations produced by the five training configurations. Additionally, we present the results of a lesioning experiment comparing the translation accuracy of a translator using the Viterbi-derivation semiring and the accuracy of one using the MinMerge-derivation semiring. Finally, we present an analysis of the behavior of the systems on a set of sentences from our development set, to show how the independent variables affect the preferences of the translator.

5.1 Data

In this work we experiment with translation of two language pairs: English to French and English to Hindi. We focus on translating from English due to the wide availability of bitexts with an English component. We selected English to French translation because we understand French, making it easier to debug and analyze the outputs of our system. We selected Hindi because the grammatical structure of Hindi is quite different from that of English, and we wanted to show that our methods work on grammatically dissimilar language pairs.

English-French: We extracted 10,000 randomly chosen sentence pairs from the English-French section of the Europarl corpus [Koehn2005] whose source and target lengths were at most 40 words. The Europarl corpus contains some errors in sentence alignments. Some sentence pairs are clearly not translations of one another, and other sentence pairs are non-literal translations. Of the 10,000 sentences, we filtered out those whose source-to-target length ratio was greater than 2:1 or less than 1:2, leaving us with 9,830 sentences for training. Many other approaches, such as [Koehn et al.2003], do not filter sentence pairs. Instead, they

select the parts of the sentence pair which are useful for constructing the phrase table, and disregard the rest. In future work, we hope to be able to incorporate such a selection strategy. We selected an additional 250 sentence pairs for tuning the regularization parameter, 1000 sentence pairs as a development set for exploratory data analysis, and 1000 sentences as the final test set. Unlike the training set, we did not filter the tuning, development, and test set.

English-Hindi: The English-Hindi data comes from the EILMT Tourism Corpus used in [Venkatapathy and Bangalore2009][1]. The data set consists of 11,300 sentence pairs for training, as well as 250 sentence pairs for tuning the regularization penalty, and 500 sentence pairs for testing. We randomly selected 1000 sentences from the training set as a development set for exploratory data analysis. Our final training set consists of 10,300 sentence pairs. We did not filter out any of the sentences in this training set.

Data preparation The data preparation steps detailed in this section are used in all our experiments. For both language pairs, we first deleted all dashes, single quote tokens, and double quote tokens. We parsed the English side of all the bitexts (training, development, tuning, and test) using the Berkeley syntactic parser [Petrov and Klein2007]. We then lexicalized the resulting parse trees using Collins' head finding rules [Collins1999]. Finally, we induced word-alignments on the training sets using the Berkeley Aligner [Haghighi et al.2009], constraining the English side with their parse trees. We used the Berkeley aligner rather than Giza++ [Och and Ney2003] because Giza++ produces alignments that are often inconsistent with syntactic parse trees.

Giza++ produces unidirectional alignments, and so state-of-the-art systems such as Moses [Koehn et al.2007] use alignments generated by the following algorithm:

1. Generate source-to-target alignments and target-to-source alignments.
2. Take the intersection and the union of the two alignment sets.
3. Augment the intersection set with alignments in the union set using the grow-diagonal-and-final heuristic [Koehn et al.2003].

The alignments the Berkeley aligner produces are bidirectional, and, with the default parameters, tend to be sparse, like the intersection of the two alignment sets from Giza++. We therefore generate our alignments with the following algorithm, motivated by the alignment algorithm used by Moses:

[1]Thanks to Dan Melamed for running these experiments at AT&T Laboratories.

1. Generate a bidirectional alignment with the Berkeley aligner using 0.5 as the posterior threshold, which is the default setting in the aligner.

2. Generate a bidirectional alignment with the Berkeley aligner using 0.35 as the posterior threshold. This threshold was tuned by trying to match the density of the union of Giza++ alignments on the English-French training set.

3. Augment the sparser alignment set with alignments from the denser set using the grow-diagonal-and-final heuristic [Koehn et al.2003].

To estimate the French language model, we selected 100K sentences from the French side of the Europarl corpus, where we excluded all sentences that were present in our 10K sentence training bitext. By excluding the sentences found in our training bitext from the language model training corpus, we ensure that the corpus used to train the model will have sentences containing unseen n-grams. The classifiers can then learn that the translator can propose a correct inference whose minimum n-gram count is 0, which they otherwise wouldn't learn since there would be no positive examples with unseen n-grams. To estimate the Hindi language models we could not follow this approach because of lack of training data. We therefore extracted the language model from the same 10,300 Hindi sentences used for training the translation model. For both French and Hindi, we set a maximum n-gram length of 3, and did not use a lower-bound frequency cutoff.

5.2 Shared experimental conditions

One of the advantages of using the parsing coordinator and GenPar is the ease with which we can control the experimental variables. The independent variables of our experiment are set by configuring the components of our training pipeline. All our experiments have more-or-less the same data flow. The data-flow diagram for all our experiments is shown in Figure 5.1. The following configuration details are constant across all our experiments.

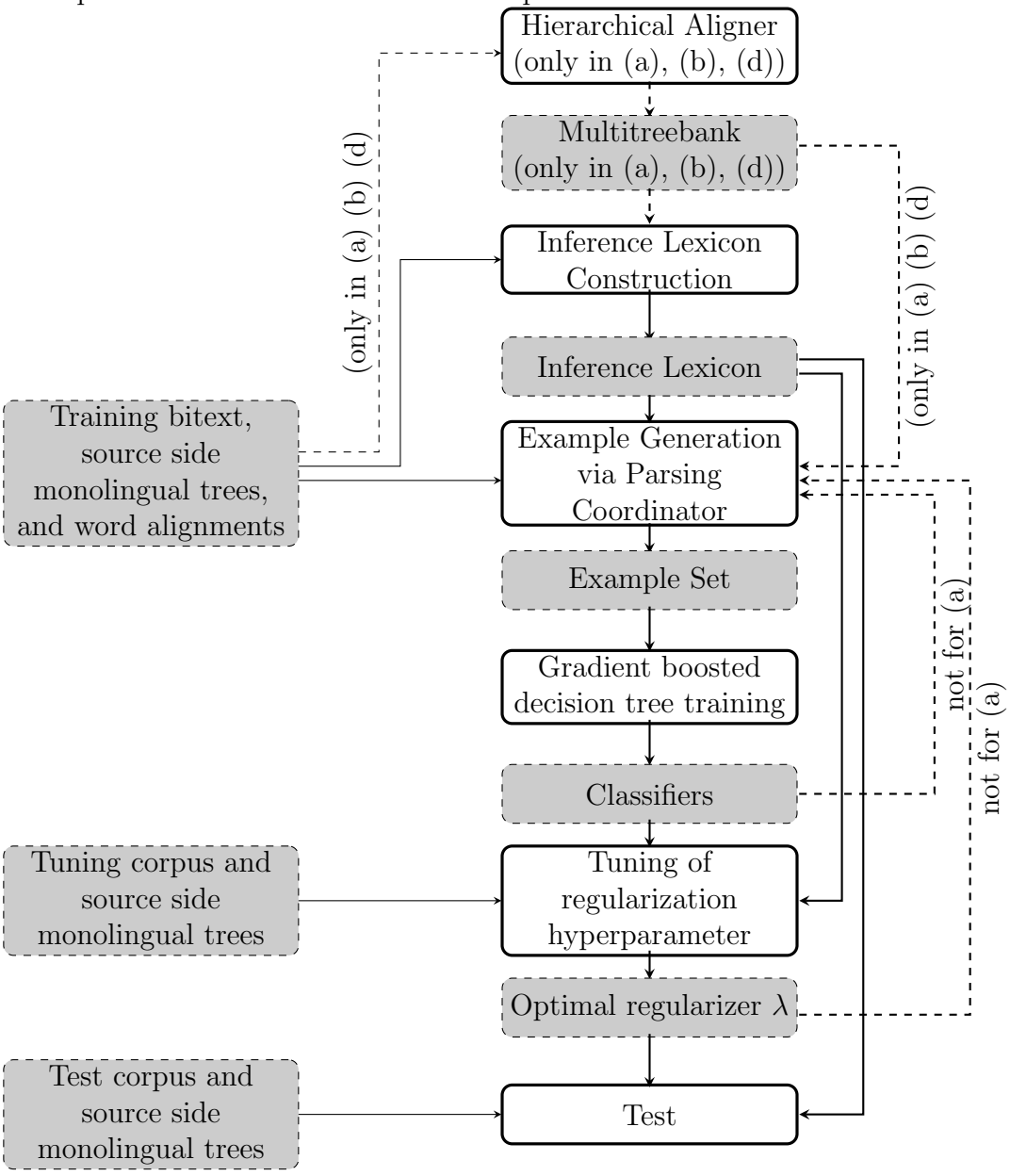
5.2.1 Grammars

All the processes share a common set of grammars – a PseudoITG, fertility grammar, multitree inference filter, inference lexicon, and set of confidence-rated classifiers.

PseudoITG

The PseudoITG is used in all the processes in our data flow that involve parsing (i.e. all the processes except for gradient boosted decision tree training).

Figure 5.1: The data-flow diagram for all our experiments. Steps that are only applicable to certain experiment configurations are specified in their boxes. The five experiments using this data flow are: (a) straw-man, (b) single tree with strict labeling, (c) forest with strict labeling, (d) single tree with permissive labeling, (e) forest with permissive labeling. White boxes represent processes, grey boxes represent data, solid lines represent data flows used in all experiments, and dashed lines represent data flows used in some experiments.



The PseudoITG is configured with three sets of constraints and scoring functions – a maximum length on fuses, a set of weighted word alignments and a monolingual English treebank containing a tree for every English sentence. When used in the hierarchical alignment process, the inference lexicon construction process, and by the supervisor in the example generation process, the PseudoITG is constrained by source monolingual trees and with weighted word alignments. When used by the predictor in the example generation process, and in the tuning and test processes, the PseudoITG is constrained only by source monolingual trees. All bilingual inferences get a cost that is the sum of the costs of the word alignments consistent with the source/target yields of the consequent of the inference. If a bilingual composition inference violates a word alignment constraint, it gets infinite cost. We chose not to smooth the costs of word alignment constraint violations (unlike CBs) because doing so had a substantial adverse effect on the running time of our experiments. In all the processes, the PseudoITG has the constraint that fuses are allowed a maximum length of 3. This constraint is imposed to limit the size of the search space for efficiency and to reduce training data fragmentation[2].

The PseudoITG uses the English trees to constrain the bilingual and 2x2 composition inferences. The PseudoITG is configured to allow an inference to violate up to three source-side constituency boundaries (i.e. brackets)[3]. An inference gets assigned a cost of 10 for each CB, which is higher than all other costs assigned by the grammars. Under this configuration, the PseudoITG biases the supervisor against items that violate the constraints imposed by the English trees, but will still allow them when needed to infer complete multitrees.

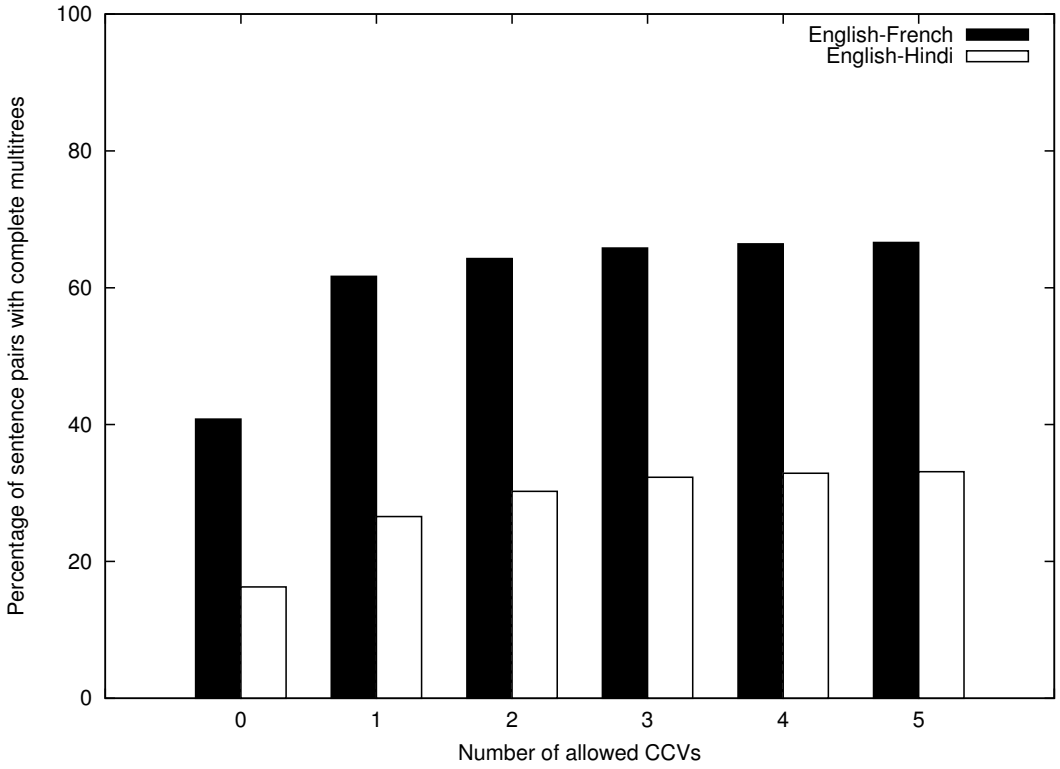
Allowing CBs has an effect on the number of sentences in the bitext for which we can construct complete multitrees. [Wellington et al.2006b] showed that due to constraining word alignments, monolingual parse trees, and lack of discontinuities allowed by a logic, one will sometimes not be able to infer a complete multitree over a sentence pair. We have observed that in the bitexts we use for training in this work, allowing CBs increases the proportion of fully multiparsed sentence pairs. Figure 5.2 contains a histogram measuring the number of allowed CBs vs. the percentage of fully parsed sentences in the English/French and English/Hindi bitexts, using the monolingual trees and word alignments described in Section 5.1.

The PseudoITG does not encode a preference among different binarizations of n-ary nodes for $n > 2$. When the supervisor has more than one way to binarize

[2]By allowing longer fuses, the classifiers will have to learn bilingual composition inferences for more source/target pairs. Longer fuses will have fewer training examples due to data sparsity, and so translation quality will degrade. In pilot experiments we observed a rather dramatic degradation of translation quality when we increased the maximum fuse length to 5 for both source and target fuses.

[3]cf. the discussion in Section 4.5.3

Figure 5.2: The number of allowed CBs vs. percentage of completely parsed English/French and English/Hindi bitext sentences



a node, it will select randomly. Doing so may be suboptimal, especially when we constrain the supervisor to infer a single pre-specified multitree for each sentence pair in the corpus, since we have no way to guarantee that the binarizations will be consistent across different sentence pairs. On the one hand, inconsistent binarizations would increase the entropy of the training set due to spurious ambiguity. On the other hand, having higher entropy may be better than learning a bad bias if we were to choose a binarization scheme that is unhelpful for translation. This problem may be somewhat mitigated in the experiments where we don't constrain the supervisor to allow only a single multitree for each sentence pair.

Fertility grammar

The fertility grammar provides an inductive bias on bilingual composition inferences. It is used when we do not yet have classifiers trained to score inferences. The parameters of the Gaussian distributions comprising the fertility grammar are estimated using the bitexts, the word alignments, and monolingual parse trees. We estimate the parameters of each Gaussian distribution $\Pr(\frac{n_t}{n_s}|X)$ as follows.

For each pair of source/target spans that are consistent with the alignment, we compute and record the ratio of their lengths. Then, for each source nonterminal label X, we estimate the maximum likelihood parameters of a Gaussian from the sample of span pairs whose source non-terminal label is X. In doing so we do not constrain the number of CBs. Due to the lack of this constraint, the models are deficient, in that they assign non-zero probability to events (i.e. bilingual compositions) that will be disallowed by the PseudoITG.

To see why a fertility model is needed, consider the tree-constrained word-aligned sentence pair in Figure 5.3(a). The target terminal "y" is unaligned. In multiparsing, y can be grouped with either x or z, leading to two possible multitrees, as in Figures 5.3(b) and 5.3(c). Additionally, the multiparser can decide that the string "a b" should be translated as a unit to "x y z", leading to the multitree in Figure 5.3(d). We cannot rely solely on the alignment probabilities assigned by the PseudoITG to disambiguate between these multitrees, since all three multitrees would get the same cost. Under the Viterbi-derivation semiring the sum of the inference costs is the sum of the alignment log probabilities. When using the fertility model under the Viterbi-derivation semiring, the cost of the multitree in Figure 5.3(b) would be $-\log(p_\ell(z,a)) + -\log(p_\ell(x,b)) + -\log(p_\phi(2|A)) + -\log(p_\phi(1|B))$, where p_ℓ is the alignment probability and p_ϕ is the fertility probability; the cost of the multitree in Figure 5.3(c) would be $-\log(p_\ell(z,a)) + -\log(p_\ell(x,b)) + -\log(p_\phi(2|B)) + -\log(p_\phi(1|A))$; and the cost of the multitree in Figure 5.3(d) would be $-\log(p_\ell(z,a)) + -\log(p_\ell(x,b)) + -\log(p_\phi(1.5|X))$.

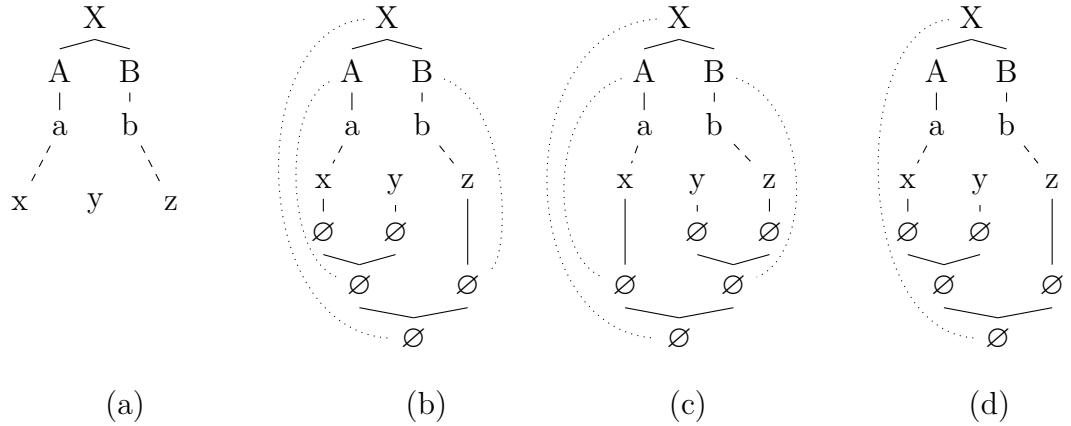
Fertility models are used in other work on SMT as well. The IBM models of [Brown et al.1993] use fertility models that model the probability that a target word t_i has generated a number of source words ϕ_i: $p(\phi_i|\mathbf{t_i})$. Their fertility models are one-to-many, not many-to-many as ours is, are discrete rather than continuous, and are conditioned only on lexical information, rather than the syntactic category dominating the source word sequence.

The fertility grammar is utilized by the hierarchical alignment process, the inference lexicon construction process, and the supervisor and predictor on the first iteration of the example generation process.

Multitree Inference Filter

The MTIF is used in the experiments where we constrain the search space to have low density by constraining the supervisor to infer a single pre-selected tree. The MTIF uses the multitrees output by the hierarchical alignment process. The MTIF is used by the inference lexicon construction process to constrain the monolingual and bilingual inferences that populate the lexicon, and by the supervisor in the example generation process.

Figure 5.3: Four possible source/target fuse pairs with different fertilities



Inference Lexicon

The inference lexicon is used to constrain the monolingual and bilingual composition inferences fired by a translator. It is used by the predictor in the example generation process, and by the translators in the tuning and test processes. The inference lexicon is constructed by a parser constrained by a PseudoITG and fertility grammar. The construction process can be constrained by an MTIF as well. When the parser is constrained by an MTIF, the inference lexicon that will be generated will contain a grammar term for every monolingual and bilingual composition in the multitrees used by the MTIF. Under this configuration, used in three of the experiments, the English-French inference lexicon contained 20,872 source-language fuses, 33,619 target-language fuses and 51,517 source/target pairs. The English-Hindi inference lexicon contained 26,881 source-language fuses, 38,413 target-language fuses and 55,925 source/target pairs.

When the multiparser used in constructing the inference lexicon is not constrained by an MTIF, we use the multiparser to infer a multiforest. The multiforest is not a complete forest over the sentence pair, since we employ beam pruning with a beam-width of 10. We populate the inference lexicon with all the monolingual and bilingual inferences that are part of any complete multitree in the inferred forest. Furthermore, for each monolingual or bilingual composition inference that ends up being part of a final tree, if the multiparser fired an inference whose consequent yields were subsequences of the yields of the consequent that ended up being part of the final tree, that inference got added to the lexicon as well. For instance, if a final parse tree contained an item yielding [a b c | x y z], and the parser has fired inferences with consequents yielding [a b | x y], [a | x], [b | y] and [c | z], all four will be added to the inference lexicon as well, even though they are not part of a complete parse. If no complete parse tree is found by the multiparser, we instead

compute 10 max covers (see Section 4.6) from the chart and proceed to populate the inference lexicon as described above. Under this configuration, the inference lexicon for English-French contained 82,115 source fuses, 125,701 target fuses and 284,073 source/target pairs. The lexicon for English-Hindi contained 72,477 source fuses, 104,900 target fuses, and 202,555 pairs.

The inference lexicon is a model that can be learned from training data. At this point, we describe an elegant but ultimately infeasible training routine for the inference lexicon, as a caution to others who might attempt it. For the purposes of illustration, suppose our training target is a multiforest over each sentence pair in the bitext, and we use a strict labeling strategy (although the following discussion applies to single tree updates and permissive labeling as well). The input to the training algorithm is a bitext with source-side parse trees, weighted word alignments, and a Gaussian fertility model. The predictor's grammar hierarchy consists of a PseudoITG, inference lexicon, and set of classifiers. On the first iteration, the supervisor's grammar hierarchy consists of a PseudoITG with weighted alignments and a Gaussian fertility model. On subsequent iterations it consists of a PseudoITG with boolean-valued alignments and the set of classifiers.

On the first iteration the inference lexicon is empty and no classifiers have been trained, so the predictor cannot propose inferences. Because the agenda is perpetually empty, on each iteration of the parsing coordinator the NextSupervisorItem() method will be called (Line 7 of Algorithm 9). The supervisor will propose inferences and score them using the alignment weights and Gaussian probabilities. The predictor will pop the item from its agenda. The MakeTrainingExample() function in Line 16 will label the example according to the update strategy and try to expand it. On the first iteration, all examples will be positive since they are all approved by the supervisor. Since there are no classifiers the grammar will assign infinite cost to every inference for which the item is an antecedent. Therefore the predictor will not push anything onto its agenda. Instead, it will keep asking the supervisor for items to expand. After each sentence in the bitext has been processed by the parse coordinator, the training set consists only of positive examples. The coordinator constructs an inference lexicon from the monolingual and bilingual compositions the predictor has been given, and the classifiers are trained on the positive examples. Because there are no negative examples for training, no features apart from the bias term will be used in the classifier.

On the second iteration, the predictor has an inference lexicon and classifier, so it can propose its own inferences. It will propose many inferences that are not approved by the supervisor that will be labeled as negative examples. Furthermore, the predictor will sometimes be unable to infer a complete parse yielding the reference translation, so the supervisor will have to provide new inferences for the predictor to make, based on the current model. Because the supervisor is con-

figured to generate inferences for which we do not have classifiers[4] the inference lexicon generated on the second iteration will have new bilingual compositions with corresponding classifiers to score them.

On the third iteration, the model will contain many classifiers that are trained to discriminate between inferences. However, for the newly generated inferences for which we had no classifiers in the second iteration, the new classifiers will give very low cost to all inferences (since they were only trained on positive examples). On this iteration, the model will be biased towards those inferences. This process will continue until convergence (i.e. until no new classifiers for new inference types are constructed).

Unfortunately, waiting for convergence greatly increases the amount of time spent training. In our preliminary experiments on English-French data, after fifteen iterations (making 3 passes over the bitext over 11 days), approximately 7% of the classifiers updated on that iteration were completely new. Furthermore, due to slow convergence, as we traced the translation accuracy of the model on the development set as training progressed, the accuracies were quite volatile across the iterations. Decreasing the learning rate wouldn't have helped, because whenever we add a new fuse pair to the inference lexicon, it will only add corresponding positive examples to learn from. The classifier will therefore give a positive confidence to any subsequent inference of the same type, which will in all likelihood be the least-cost item proposed by the translator (since most other competing bilingual inferences will have negative confidences, due to the tendency of the training set to have more negative than positive examples. Decreasing the magnitude of the confidence assigned to the new inference will not help.

Given the long training time, we were unable to run this setup to completion. Therefore, we construct the inference lexicon as a preprocessing step by using a multiparser configured with a grammar to encode preferences among the trees (namely a PseudoITG and fertility grammar) and generating an inference lexicon from a single multitree or multiforest. On subsequent iterations we utilize the model to discriminate between the source/target fuse pairs the lexicon allows.

Confidence-rated classifiers

The confidence rated classifiers are used to score inferences in the example generation process, and in the tuning and test processes. They are utilized by the supervisor and predictor in the example generation process only after the first iteration.

[4]The supervisor is configured to generate the inferences by assigning a high finite cost when the classifiers return infinite cost for the inference

5.2.2 Parsing coordinator

The parsing coordinator is used in the example generation process of the data-flow shown in Figure 5.1. The parsing coordinator iterates over the bitext, and for each sentence pair it coordinates the search procedure of the supervisor and the predictor.

We refer to each complete pass over the bitext as an *epoch*. An epoch is not necessarily the same as an iteration. Rather than reestimating the model after each pass through the bitext, we can partition the corpus so that we generate examples from part of the corpus, reestimate the model, generate examples from the next part of the corpus, etc. Doing so gives us the advantage of updating the model more frequently allowing the training process to learn to correct translation errors made by the model more often[5]. In all our experiments (apart from the non-iterative straw-man experiment) we partition the bitext into five partitions, yielding five iterations for each epoch. We partitioned the bitext into five parts because we wanted to allow more frequent model updates while not substantially increasing training time. If we were to update the model after every sentence, for example, the training time would have increased by a prohibitive amount.

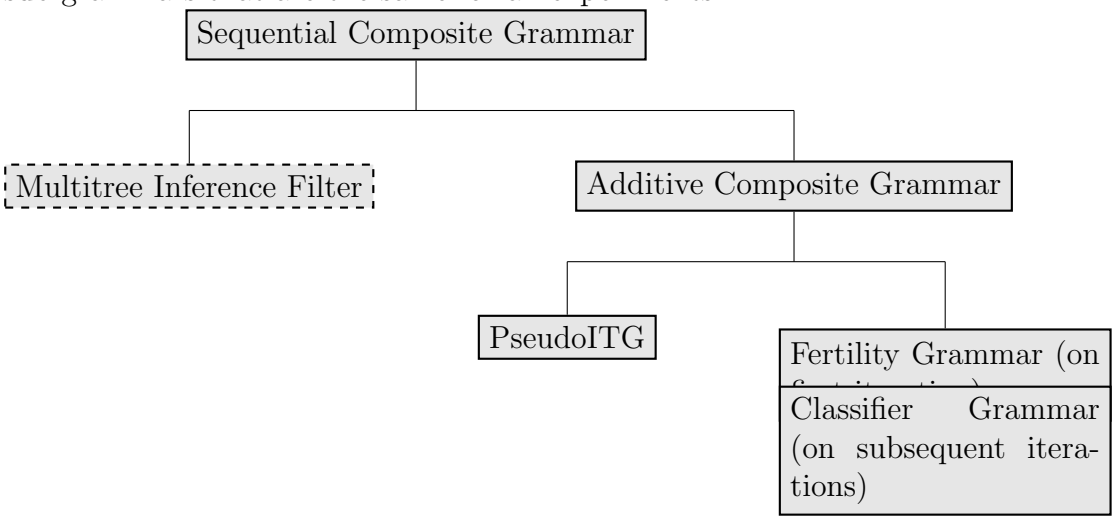
The amount of exploration done by the predictor is limited by the exploration threshold set by the parsing coordinator. The exploration threshold determines how many states the predictor will explore. With exploration threshold 0, it will explore only states allowed by the supervisor. As we increase the exploration threshold the predictor is able to explore more and more states, most of them inconsistent with the multitree or multiforest inferable by the supervisor.

The predictor is never able to enumerate all the states it can explore under a high threshold. The coordinator has to select the ones that will be most useful for learning. One criterion for selecting the inferences to use in the training set is according to the model's own biases. The incorrect states that the predictor will find itself in are those that are given low cost by the model. The distribution of inferences fired by the predictor during training are likely to have a similar distribution to those of the inferences fired during test, and therefore we wish to learn from them to avoid making bad inferences.

Rather than setting a fixed exploration threshold for training we gradually increase the threshold for each epoch. On the first epoch we set the exploration threshold to 0. For each epoch, the exploration threshold is incremented. For all our iterative experiments, we run the coordinator through five epochs, thereby setting the maximum exploration threshold to 4. By gradually increasing the exploration radius, we build a series of models that the predictor uses for state exploration. This idea is similar to *curriculum learning* [Bengio et al.2009], where

[5]More frequent model updates in this manner are similar to early updating [Daumé and Marcu2005].

Figure 5.4: The supervisor's grammar hierarchy. Boxes with solid edges represent sub-grammars that are the same for all experiments.



the idea is to start to learn a simple model, and gradually make the learning task more difficult.

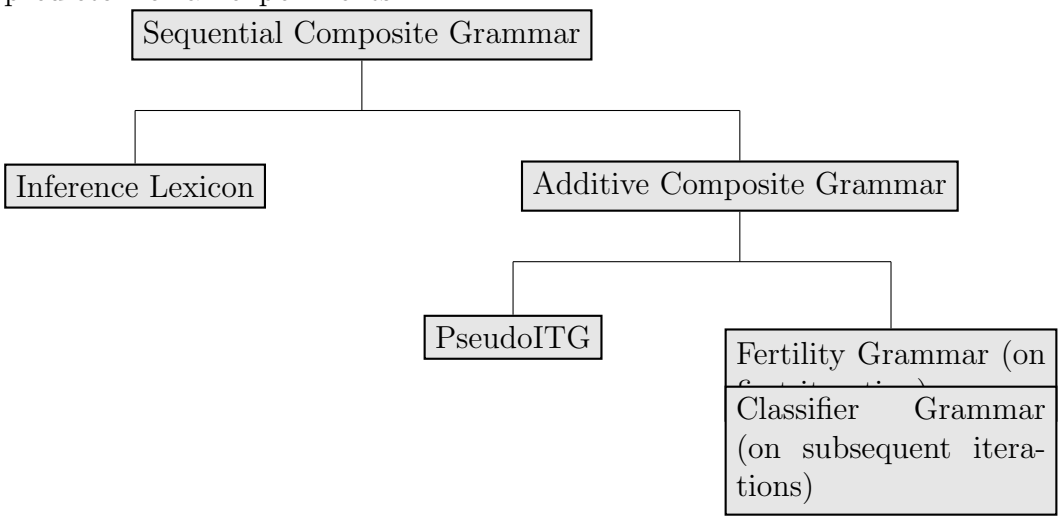
5.2.3 Supervisor configuration

Recall that the supervisor is used to evaluate inferences fired by the predictor and to guide the predictor when it is unable to infer a multitree yielding the reference translation. The supervisor is a multiparser and utilizes Logic MP. The grammar hierarchy of the supervisor, shown in Figure 5.4, has three subgrammars that are constant for all experiments – a PseudoITG, confidence-rated classifiers and fertility grammar.

5.2.4 Predictor

The predictor is a translator, and is parameterized by Logic T. Its grammar hierarchy, shown in Figure 5.5, is nearly identical to that of the supervisor. Unlike the supervisor the predictor's PseudoITG contains only monolingual tree constraints. The predictor has a component in its grammar hierarchy that replaces the MTIF, namely the inference lexicon. The inference lexicon assigns a cost to fuse inferences inversely proportional to their frequency in the lexicon – namely for fuse inference i, $V(i) = \frac{\alpha}{n}$ where n is the frequency of the consequent of i in the lexicon, and α is a manually selected value. We found that $\alpha = 2$ gave the best results in pilot experiments. In practice, the scores given by the inference lexicon tended to be the lowest cost in the inference sequence (by design), and were used to order the

Figure 5.5: The predictor's grammar hierarchy. The hierarchy is used by the predictor for all experiments.



1D fuses on the agenda.

5.3 Independent variables

In our experiments, there are three independent variables – the amount of exploration allowed by the coordinator, the density of the search space, and the labeling strategy. For our first experiment, we wished to confirm the utility of allowing exploration by using the training method of [Wellington et al.2006a] as a straw-man condition, although with our logic and feature set. After confirming the utility of exploration, we are left with the density of the search space and labeling strategy with which to experiment. We have five main experimental conditions that we run – the straw-man condition without exploration; the baseline single-tree update using the strict labeling strategy; the single-tree update using the permissive labeling strategy; the forest update using the strict labeling strategy; and the forest update using the permissive labeling strategy.

5.3.1 Conditions with single-tree constraints

The first step in these three conditions is to infer a multitreebank over the bitexts. We used a multiparser called a *hierarchical aligner* to infer the multitreebank over the bitext[6]. This step is denoted in the first box of Figure 5.1. To be a hierarchical aligner, the multiparser was configured in the same manner as the

[6]Refer to [Melamed and Wang2004] for a discussion of hierarchical alignment.

supervisor parser described above. The grammar hierarchy was the same as the supervisor's, except that it did not include a multitree inference filter, and did not use the classifiers – it utilized only the PseudoITG and fertility grammar as described in the previous section.

We then constructed the inference lexicon from the multitreebank. We created a lexicon entry from each source and target fuse inference and bilingual composition inference in each multitree, and pruned the resulting inference lexicon according to the criteria laid out in Section 4.5.4.

The only difference between the straw-man and the baseline single-tree update experiments was that the straw-man used a maximum exploration threshold of 0, thereby requiring only a single epoch. When we limit the exploration threshold to 0, all the inferences that the parser is able to fire will be added to the training set in one epoch. Increasing the number of epochs will not add any new inferences to the training set. The coordinator for the other two single-tree experiments ran for five epochs, therefore using a maximum exploration threshold of four. The only difference between the single-tree strict labeling update and the single-tree permissive labeling update was the labeling strategy. We utilized the same multitreebank and inference lexicons for all three experiments.

5.3.2 Conditions with multiforest constraints

Constraining the training routine to update towards a single multitree limits the coverage of our grammar and forces the learning algorithm to update towards a path which may be difficult for it to learn[7]. Some other approaches to SMT training do not impose such a constraint. The models utilized by the finite-state translation systems of [Koehn et al.2003, Och2003, Och and Ney2004] allow phrases consistent with many different segmentations of each training sentence pair, rather than committing to learning a single segmentation. Similarly, the hierarchical phrase method of [Chiang2005, Chiang et al.2008] estimates hierarchical phrase rules from a multiforest over each training sentence pair. The forest approach of [Mi et al.2008] allows all translation productions consistent with any tree in the forest over the source sentence.

Instead of constraining the supervisor to allow only predictor inferences that are consistent with a particular multitree, we can remove the multitree inference filter from the supervisor's grammar hierarchy and let the supervisor allow any predictor inference consistent with the sentence pair and PseudoITG constraints. Due to the design of the parse coordinator, we need not explicitly compute a multiforest over the input sentence – we merely need to determine whether each inference made by the predictor is approved by the supervisor. The only time we

[7]Where there may be other paths for the same sentences that are more consistent with the model's biases.

need to build a multiforest is when we infer an initial inference lexicon.

In our pilot experiment for forest strict labeling updates, we applied an example bias to each example generated by the parse coordinator, by weighting it in inverse proportion to the number of other inferences yielding a consequent item covering the same source span of the input sentence. The intuition behind doing so was that when there are not many inferences that can yield a consequent for that span, we wished for those to be more highly weighted positive examples than those inferences with many possible alternatives, because we considered the heavier examples more reliable for learning. This approach, however, led to a decline in translation accuracy compared to assigning bias for each inference. We are not certain why this was the case.

Whereas the first step in the straw-man and single-tree update experiments was to use a hierarchical aligner to infer a multitreebank over the bitext, we skipped that step in the forest experiments. Instead, we used the supervisor multiparser unconstrained by a multitree inference filter to generate an inference lexicon.

The only difference between the forest strict labeling update and the forest permissive labeling update is the labeling strategy. The same inference lexicons were used for both experiments.

5.4 Dependent variables

Our primary evaluation measure for English-French translation consisted of manually evaluating a sample of the test output of our five systems. We selected a random sample of 100 sentences from the test corpus, and collected the translations output by each of the five systems for each English sentence. For each English sentence, we randomly permuted the five translations. We then partitioned the 100 sentences and their five translations among three French speakers[8] to evaluate. We asked them to rank the systems' translations on two criteria: semantic fidelity to the source sentence, that is, how well they could reconstruct the meaning of the English sentence from the French translation, and the grammaticality or fluency of the translation. No ties were allowed in the rankings. If an evaluator considered two translations to be equally good or bad, they were asked to tiebreak according to their personal preference. If two translations of an English sentence were identical, we did not include them in the comparisons when we compared the systems pairwise. The exact instructions were as follows:

> Thank you very much for helping me with my dissertation. I've sent you 33 English sentences with five French translations each. Most translations have both grammatical errors and semantic errors. Your task is to rank the five French translations for each English sentence

[8] One of whom was the author.

in two ways – in terms of their fidelity to the meaning of the English without regards to their grammaticality, and in terms of the number and severity of grammatical mistakes. For example, take the following English sentence and French translations:

```
Mr. Anastassopoulos , that is the price of democracy .

___  ___ M. Anastassopoulos , c' le prix est des garantir .
___  ___ M. Anastassopoulos , c' est le prix des democratie .
___  ___ M. Anastassopoulos , c' est une prix de la democratie .
___  ___ M. Anastassopoulos , c' est le prix de la democratie .
___  ___ M. Anastassopoulos , le est le monnaie des democratie .
```

I would rank them as follows, where the first column is the semantic fidelity and the second is the grammaticality

```
_5_  _4_ M. Anastassopoulos , c' le prix est des garantir .
_2_  _3_ M. Anastassopoulos , c' est le prix des democratie .
_3_  _2_ M. Anastassopoulos , c' est une prix de la democratie .
*_1_  _1_ M. Anastassopoulos , c' est le prix de la democratie .
_4_  _5_ M. Anastassopoulos , le est le monnaie des democratie .
```

When determining the ranking based on grammaticality, please rank them based on the number and severity of grammatical mistakes, including poor word ordering, inserted words, incorrect gender/number/person agreement, etc. When determining the ranking based on semantic fidelity, please rank them according to how well you can reconstruct the original meaning from the French translation. Ties are not allowed! If you think that two translations are equally good/bad, please break the tie according to your personal preference.

When you find translations that might be considered correct, even if they're not very literal, please mark them with an asterisk.

Finally, I would welcome any comments or questions you have on what I've asked you to do, or on your experience doing it.

Using the rankings, we counted how many translations output by each system were better than the translations output by each of the other systems. As stated earlier, if two translations were the same, we did not include them in the count.

We also used an automatic evaluation measure for the quality of translations. We utilized the Precision/Recall/F-measure measures discussed in Section 4.8.2.

Table 5.1: Results of a pairwise comparison of the accuracy of the translation systems based on manual evaluations. Each entry contains the number of translations proposed by the system in the row that were better than the corresponding translation output by the system in the column. Italicized numbers are statistically significant at p=0.05, and bold numbers are statistically significant at p=0.01. Statistical significance was computed by the sign test. ST refers to the baseline experiment, PT refers to single-tree update with permissive labeling, SF refers to forest update with strict labeling, and PF refers to forest update with permissive labeling.

Semantic Fidelity					
	Straw-man	**ST**	**PT**	**SF**	**PF**
Straw-man	-				
ST	53	-			
PT	**71**	**76**	-		
SF	**65**	**69**	52	-	
PF	**69**	**72**	*61*	**64**	-
Grammaticality					
	Straw-man	**ST**	**PT**	**SF**	**PF**
Straw-man	-				
ST	56	-			
PT	**68**	**71**	-		
SF	**66**	**67**	50	-	
PF	**67**	**72**	*62*	*61*	-

5.5 Results

The results can be found in Table 5.1. For all the hypothesis tests, both for the manual evaluations and the automatic evaluations, the null hypothesis was that there was no difference between the quality of the translations output by the different systems. The manual evaluations confirm our hypothesis that updating towards multiforests is better than updating towards a single multitree, and that using permissive labeling is better than using strict labeling.

We were unable to find evaluators to evaluate English-Hindi translation. We therefore had to rely on automatic evaluations for English-Hindi. To validate the automatic evaluation measures, we first computed evaluation measures for English-French translation to show that the automatic measures correlate well with human judgements for our experiments. The correlation between the automatic evaluation measures and the manual evaluations validate the automatic evaluation of Hindi translations as well. The automatic evaluations of the English-French experiments are presented in Table 5.2 and the evaluations of the English-Hindi

experiments are in Table 5.3. We utilized three automatic evaluation measures: precision/recall/f-measure [Melamed et al.2003], BLEU [Papineni et al.2002] and NIST [Doddington2002]. Statistical significance on precision/recall/f-measure were computed using the Wilcoxon signed-ranks test, and statistical significance on BLEU and NIST scores was computed using bootstrap resampling [Koehn2004]. The matrices of statistical significance on precision/recall/f-measure are in Tables 5.5 and 5.6. To situate the accuracy of systems in the research literature, we also trained finite-state phrase models for English-French and English-Hindi translation using the moses SMT system [Koehn et al.2007]. We ran the training procedure using the default options, except that we limited maximum phrase length to three, mirroring our constraint on fuse lengths. Statistical significance was computed with respect to the translation accuracies of our best systems[9].

Automatic evaluation of SMT is a difficult problem in and of itself. Automatic evaluation is usually done by counting the number and length of word sequences in the intersection between an output translation and a set of reference translations (e.g. [Papineni et al.2002, Melamed et al.2003, Lavie and Agarwal2007]). However, given the ambiguity of natural language and given that there may be many valid translations for a single source sentence, comparing a translation against a finite (usually small) number of references may lead to some perfectly valid translations receiving a low evaluation score due to lack of overlap between the output and reference. Furthermore, some studies have shown that there may be cases where the automatic evaluation measure does not correlate well with human judgements (e.g. [Callison-Burch et al.2006]). However, automatic evaluation is inexpensive in terms of time compared to manual evaluations, and so is widely utilized in the research community. Also, automatic evaluations are needed to tune our hyperparameters. The optimal regularization penalties for each experiment are presented in Table 5.4.

The straw-man configuration, which utilizes the training approach of [Wellington et al.2006a], has the advantage of being comparatively fast to train. On a cluster of approximately 100 CPU cores, both English-French and English-Hindi straw-man configurations took a few hours to train. The single tree updated models took several days, with no discernible difference in training times between strict and permissive labeling strategies. The forest-updated models each took approximately two weeks to train. One of the primary disadvantages of our approach compared to other systems in the literature is the time it takes to train the systems. We translate the entire bitext multiple times, unlike the translator used by MERT [Och2003] and MIRA [Chiang et al.2008] that iterate on roughly one thousand sentences. The only other fully discriminative approaches we found that translate the entire bitext are [Liang et al.2006], who limit the sentences they train on to have length

[9]For both English-French and English-Hindi, the most accurate translations were produced by the system trained using permissive labeling with the supervisor constrained by a multiforest.

Table 5.2: Automatic evaluation measures of our five English-French translation systems. We computed Precision (**Prec**), Recall (**Rec**), and F-measure (**FMS**) with exponents 1 (first row of each system) and 2 (second row of each system). For each system, we also computed the BLEU score over the test set. Numbers in bold indicate statistical significance at $p = 0.01$, and numbers in italic indicate statistical significance at $p = 0.05$. For the single tree strict labeling systems, statistical significance was computed with respect to the straw-man configuration. For the forest strict labeling systems, statistical significance was computed with respect to the single tree strict labeling systems. For the single tree and forest permissive labeling systems, statistical significance was calculated with respect to the single tree strict labeling and forest strict labeling systems, respectively.

English-French Translation									
	Expo-	Development			Test				
	nent	**Prec**	**Rec**	**FMS**	**Prec**	**Rec**	**FMS**	**BLEU**	**NIST**
Straw-man	1	0.4985	0.4643	0.4808	0.5017	0.4715	0.4861	0.1299	4.6556
	2	**0.2177**	0.2028	0.2100	0.2181	0.2050	0.2113		
Single tree strict labeling	1	**0.5039**	**0.4859**	**0.4948**	0.5018	**0.4855**	**0.4935**	**0.1635**	**5.2674**
	2	**0.2237**	**0.2157**	**0.2197**	**0.2259**	**0.2186**	**0.2222**		
Forest strict labeling	1	0.5055	**0.4986**	*0.5020*	**0.5103**	**0.4949**	**0.5025**	**0.1778**	**5.4180**
	2	**0.2295**	**0.2264**	**0.2279**	**0.2323**	**0.2253**	**0.2287**		
Single tree permissive labeling	1	*0.5110*	0.4818	0.4960	**0.5140**	0.4869	**0.5001**	0.1628	**5.3625**
	2	**0.2320**	0.2188	**0.2252**	**0.2316**	0.2194	*0.2253*		
Forest permissive labeling	1	**0.5178**	0.4957	0.5065	**0.5197**	**0.5015**	**0.5105**	*0.1819*	**5.5516**
	2	**0.2374**	0.2273	*0.2322*	**0.2384**	**0.2300**	**0.2341**		
Moses	1	**0.5325**	**0.5310**	**0.5337**	**0.5325**	**0.5345**	**0.5335**	**0.2333**	**5.9237**
	2	**0.2585**	**0.2578**	**0.2591**	**0.2585**	**0.2594**	**0.2590**		

Table 5.3: Automatic evaluation measures of our five English-Hindi translation systems. We computed Precision (**Prec**), Recall (**Rec**), and F-measure (**FMS**) with exponents 1 (first row of each system) and 2 (second row of each system). For each system, we also computed the BLEU score over the test set. Numbers in bold indicate statistical significance at $p = 0.01$, and numbers in italic indicate statistical significance at $p = 0.05$. For the single tree strict labeling systems, statistical significance was computed with respect to the straw-man configuration. For the forest strict labeling systems, statistical significance was computed with respect to the single tree strict labeling systems. For the single tree and forest permissive labeling systems, statistical significance was calculated with respect to the single tree strict labeling and forest strict labeling systems, respectively.

English-Hindi Translation								
	Expo-	Development			Test			
	nent	**Prec**	**Rec**	**FMS**	**Prec**	**Rec**	**FMS**	**BLEU**	**NIST**
Straw-man	1	0.5271	0.4975	0.5119	0.5226	0.5002	0.5112	0.1281	5.1594
	2	0.2014	0.1901	0.1956	0.2010	0.1923	0.1966		
Single tree strict labeling	1	0.5303	0.5020	0.5157	**0.5338**	**0.5093**	**0.5213**	*0.1349*	**5.2399**
	2	0.1993	0.1886	0.1938	0.2052	0.1958	0.2004		
Forest strict labeling	1	**0.5566**	**0.5097**	**0.5321**	**0.5546**	0.5161	**0.5347**	**0.1503**	**5.4627**
	2	**0.2158**	**0.1976**	**0.2063**	**0.2188**	**0.2036**	**0.2109**		
Single tree permissive labeling	1	**0.5394**	**0.5124**	**0.5256**	**0.5466**	**0.5254**	**0.5358**	**0.1561**	**5.5114**
	2	**0.2069**	**0.1966**	**0.2016**	**0.2156**	**0.2072**	**0.2113**		
Forest permissive labeling	1	0.5504	**0.5207**	0.5352	0.5535	**0.5322**	**0.5427**	*0.1574*	**5.5559**
	2	0.2169	**0.2052**	**0.2109**	0.2174	**0.2090**	*0.2131*		
Moses	1	0.5383	0.5263	0.5322	0.5454	**0.5449**	0.5451	**0.2065**	5.6021
	2	0.2220	0.2171	0.2195	**0.2346**	**0.2344**	**0.2345**		

Table 5.4: The optimal regularization penalties λ for each experiment

English-French	
Straw-man	0.00585937500
Single tree strict labeling	0.00585937500
Forest strict labeling	0.02343750000
Single tree permissive labeling	0.00585937500
Forest permissive labeling	0.00585937500
English-Hindi	
Straw-man	0.00585937500
Single tree strict labeling	0.00146484375
Forest strict labeling	0.00585937500
Single tree permissive labeling	0.00292968750
Forest permissive labeling	0.00585937500

Figure 5.6: Single tree strict labeling English-French translation accuracy on development set across iterations. The top graph shows precision/recall/f-measure with exponent 1; the bottom graph shows precision/recall/f-measure with exponent 2.

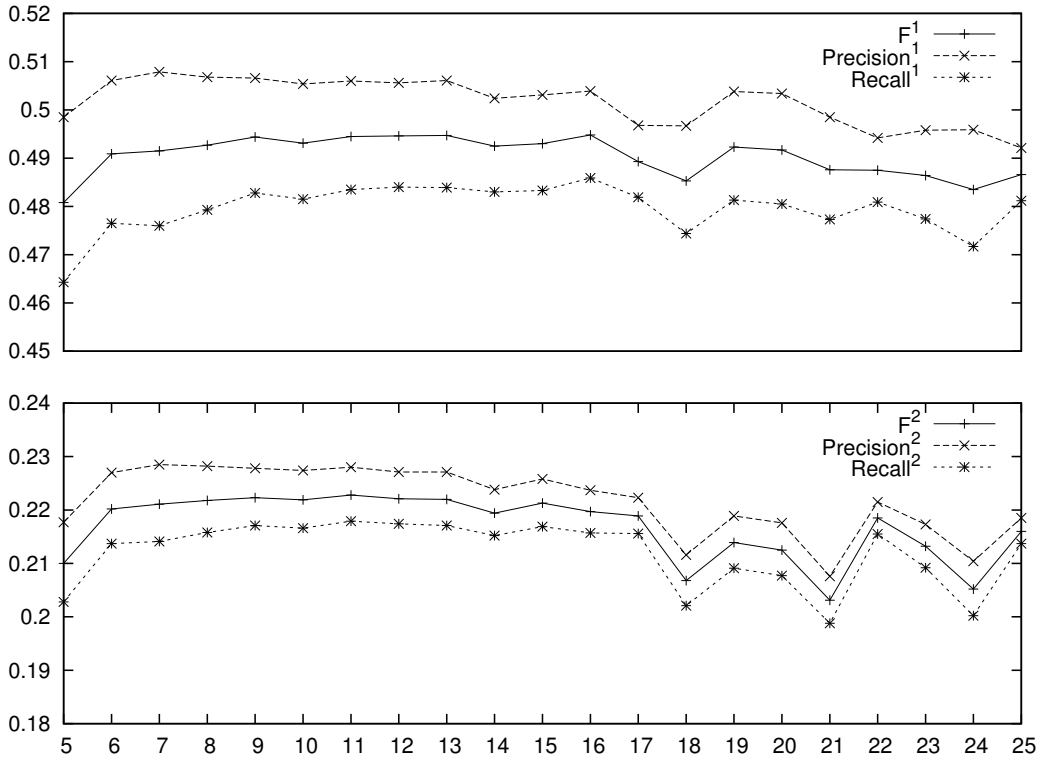


Table 5.5: p-values for English-French test set, comparing the minimum statistical significance levels at which the relevant null hypothesis was rejected for the system in the row and the system in the column. There are no entries in the table for which the system in the row is worse than the system in the column.

		Straw-man		
System	**Exp**	**Prec:**	**Rec:**	**FMS:**
Single tree strict	1	0.7838	$1.72 \times 10^{-}10$	0.00062
	2	0.00027	$1.64 \times 10^{-}15$	$2.04 \times 10^{-}9$
Forest strict	1	0.0056	1.20×10^{-15}	4.48×10^{-9}
	2	6.41×10^{-13}	3.71×10^{-26}	7.56×10^{-20}
Single tree permissive	1	3.292×10^{-8}	1.197×10^{-13}	4.29×10^{-11}
	2	1.25×10^{-18}	1.09×10^{-22}	1.17×10^{-20}
Forest permissive	1	5.56×10^{-10}	9.92×10^{-24}	8.69×10^{-18}
	2	5.11×10^{-24}	7.36×10^{-38}	4.54×10^{-32}
		Single tree strict		
	Exp	**Prec:**	**Rec:**	**FMS:**
Forest strict	1	0.0033	0.0050	0.0022
	2	$1.43 \times 10^{-}4$	1.842×10^{-5}	2.61×10^{-5}
Single tree permissive	1	2.58×10^{-5}		0.0099
	2	$1.16 \times 10_{-4}$		0.0126
Forest permissive	1	1.27×10^{-9}	$4.42 \times 1-^{-8}$	8.9×10^{-10}
	2	1.49×10^{-11}	1.71×10^{-11}	2.03×10^{-12}
		Forest strict		
	Exp	**Prec:**	**Rec:**	**FMS:**
Single tree permissive	1			
	2			
Forest permissive	1	4.68×10^{-4}	5.38×10^{-4}	3.62×10^{-4}
	2	4.54×10^{-4}	6.96×10^{-4}	6.63×10^{-4}
		Single tree permissive		
	Exp	**Prec:**	**Rec:**	**FMS:**
Forest permissive	1	0.054	1.11×10^{-6}	4.13×10^{-4}
	2	9.41×10^{-5}	1.19×10^{-9}	2.48×10^{-7}

Table 5.6: p-values for the English-Hindi test set, comparing the minimum statistical significance levels at which the relevant null hypothesis was rejected for the system in the row and the system in the column. There are no entries in the table for which the system in the row is worse than the system in the column.

		Straw-man		
	Exp	**Prec:**	**Rec:**	**FMS:**
Single tree strict	1	0.0014	0.0046	0.00062
	2	0.393	0.622	0.50
Forest strict	1	1.46×10^{-15}	2.82×10^{-4}	3.93×10^{-9}
	2	1.19×10^{-9}	3.38×10^{-4}	2.07×10^{-6}
Single tree permissive	1	1.01×10^{-8}	1.00×10^{-9}	4.29×10^{-11}
	2	4.24×10^{-8}	6.93×10^{-10}	3.79×10^{-9}
Forest permissive	1	3.25×10^{-13}	3.75×10^{-13}	3.08×10^{-12}
	2	1.69×10^{-10}	2.22×10^{-12}	7.52×10^{-12}
		Single tree strict		
	Exp	**Prec:**	**Rec:**	**FMS:**
Forest strict	1	3.47×10^{-8}	0.066	2.90×10^{-4}
	2	$4.65 \times 10^{-}8$	2.31×10^{-5}	
Single tree permissive	1	0.0017	5.94×10^{-5}	2.15×10^{-4}
	2	1.4×10^{-4}	2.58×10^{-6}	2.34×10^{-5}
Forest permissive	1	6.42×10^{-6}	9.05×10^{-9}	2.87×10^{-8}
	2	4.77×10^{-7}	1.41×10^{-9}	8.39×10^{-10}
		Forest strict		
	Exp	**Prec:**	**Rec:**	**FMS:**
Single tree permissive	1		0.033	0.817
	2		0.108	0.768
Forest permissive	1	0.33	5.13×10^{-7}	0.0092
	2	0.55	6.88×10^{-5}	0.047
		Single tree permissive		
	Exp	**Prec:**	**Rec:**	**FMS:**
Forest permissive	1	0.0098	0.0174	0.012
	2	0.096	0.0014	0.0038

Figure 5.7: Forest strict labeling English-French translation accuracy on development set across iterations. The top graph shows precision/recall/f-measure with exponent 1, the bottom graph shows precision/recall/f-measure with exponent 2.

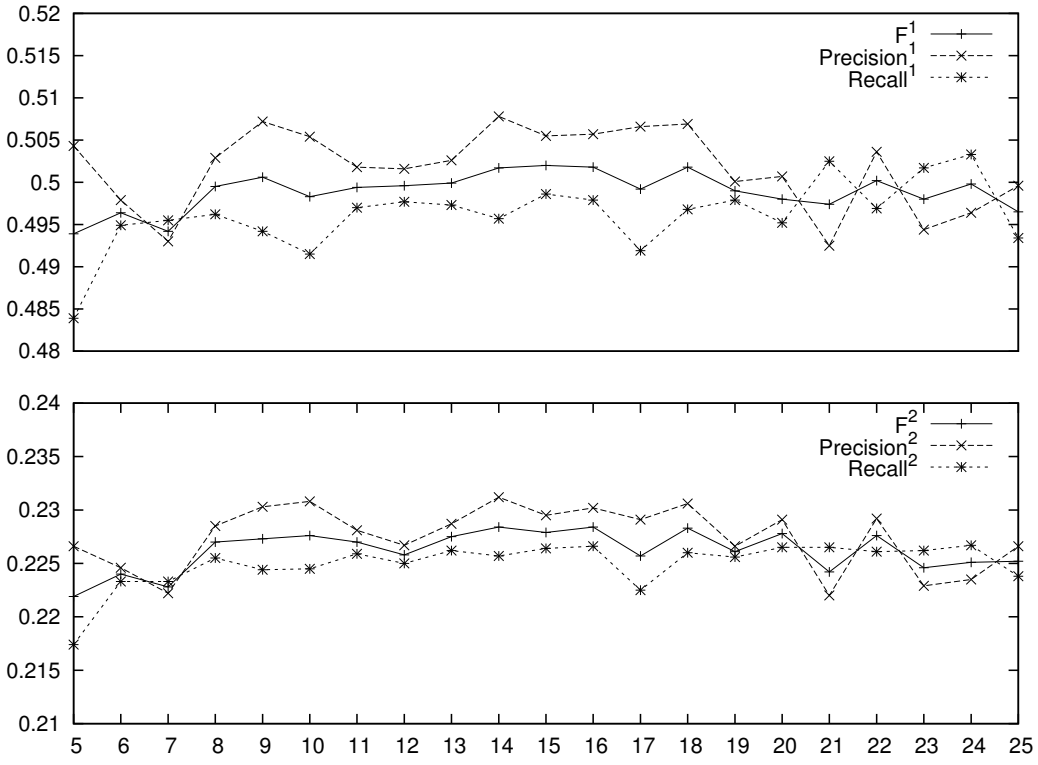


Figure 5.8: Single tree permissive labeling English-French translation accuracy on development set across iterations. The top graph shows precision/recall/f-measure with exponent 1, the bottom graph shows precision/recall/f-measure with exponent 2.

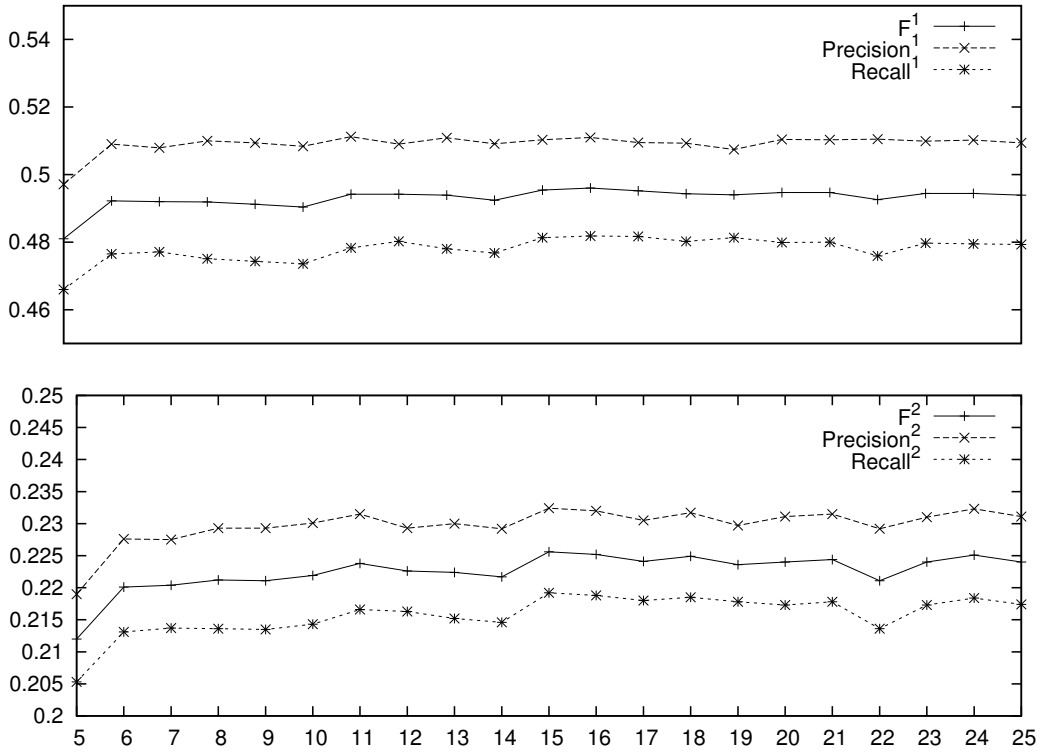


Figure 5.9: Forest permissive labeling English-French translation accuracy on development set across iterations. The top graph shows precision/recall/f-measure with exponent 1, the bottom graph shows precision/recall/f-measure with exponent 2.

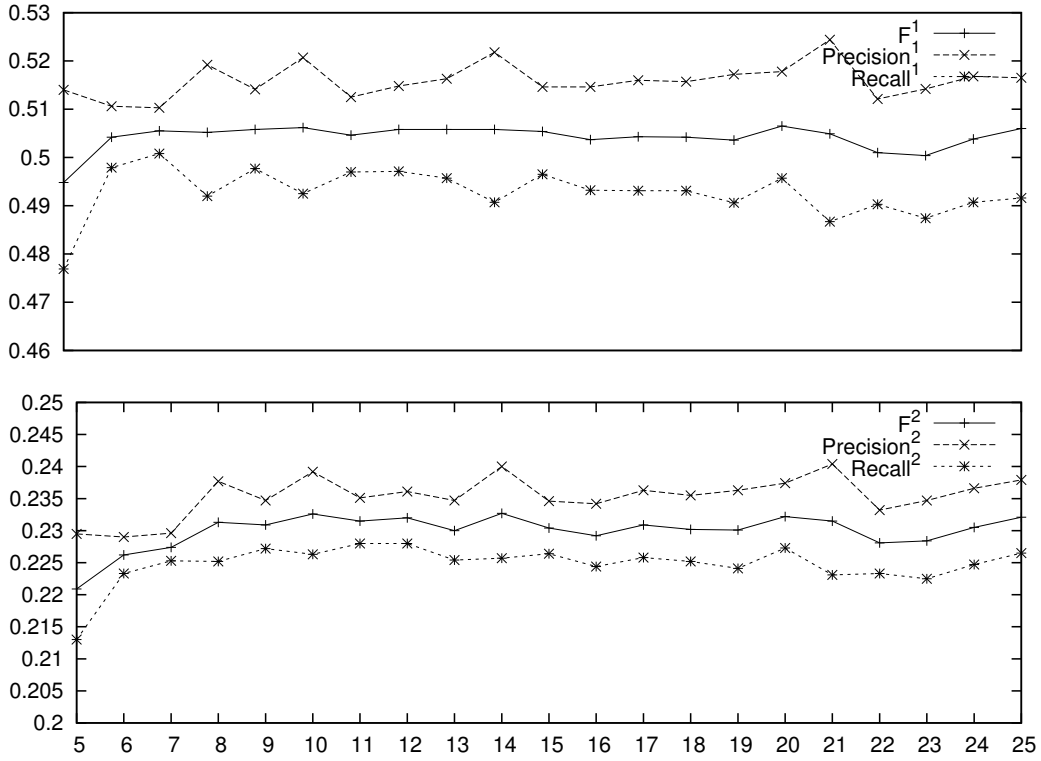


Figure 5.10: Single tree strict labeling English-Hindi translation accuracy on development set across iterations. The top graph shows precision/recall/f-measure with exponent 1, the bottom graph shows precision/recall/f-measure with exponent 2.

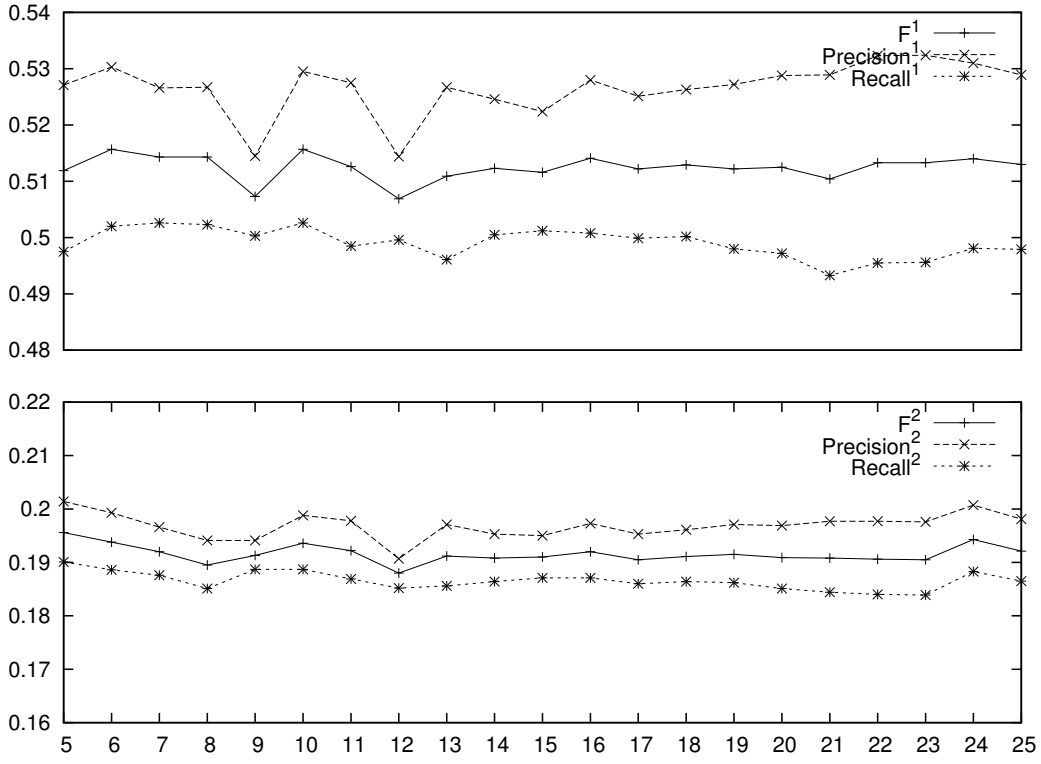


Figure 5.11: Forest strict labeling English-Hindi translation accuracy on development set across iterations. The top graph shows precision/recall/f-measure with exponent 1, the bottom graph shows precision/recall/f-measure with exponent 2.

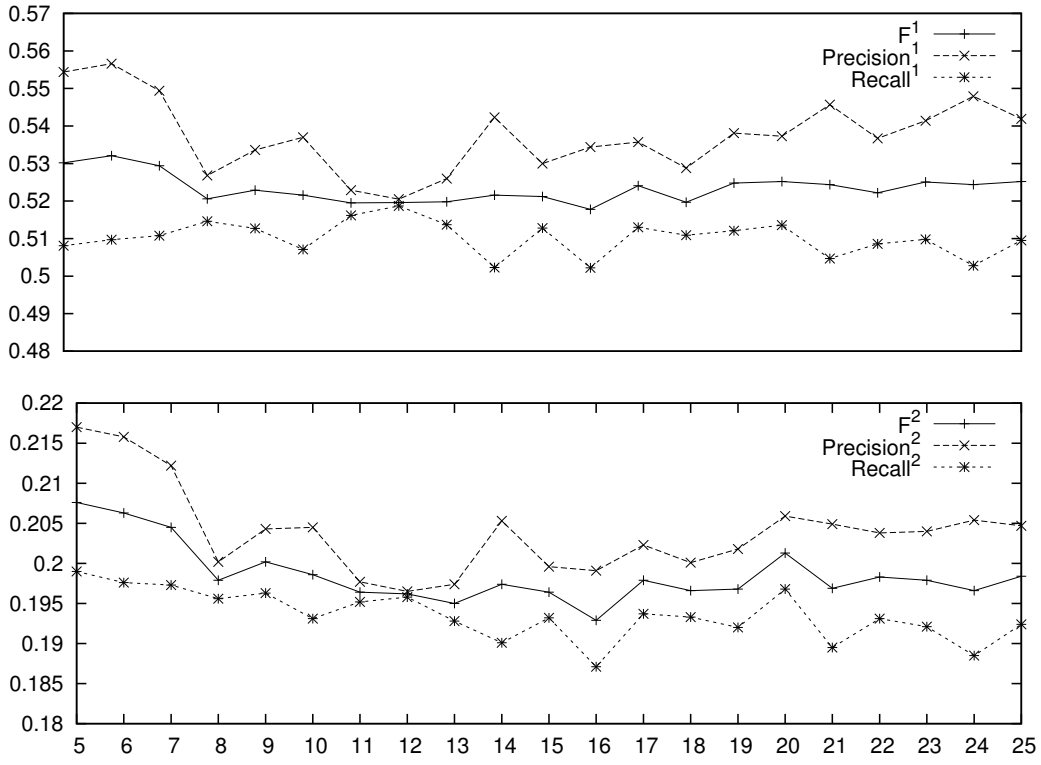


Figure 5.12: Single tree permissive labeling English-Hindi translation accuracy on development set across iterations. The top graph shows precision/recall/f-measure with exponent 1, the bottom graph shows precision/recall/f-measure with exponent 2.

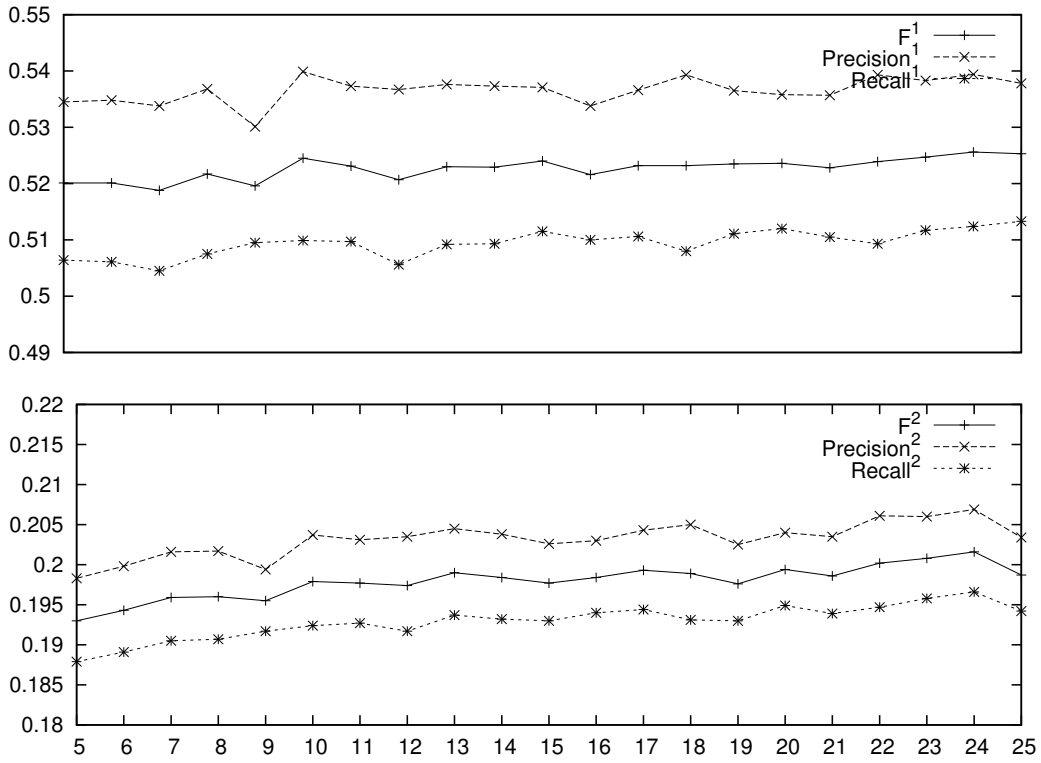
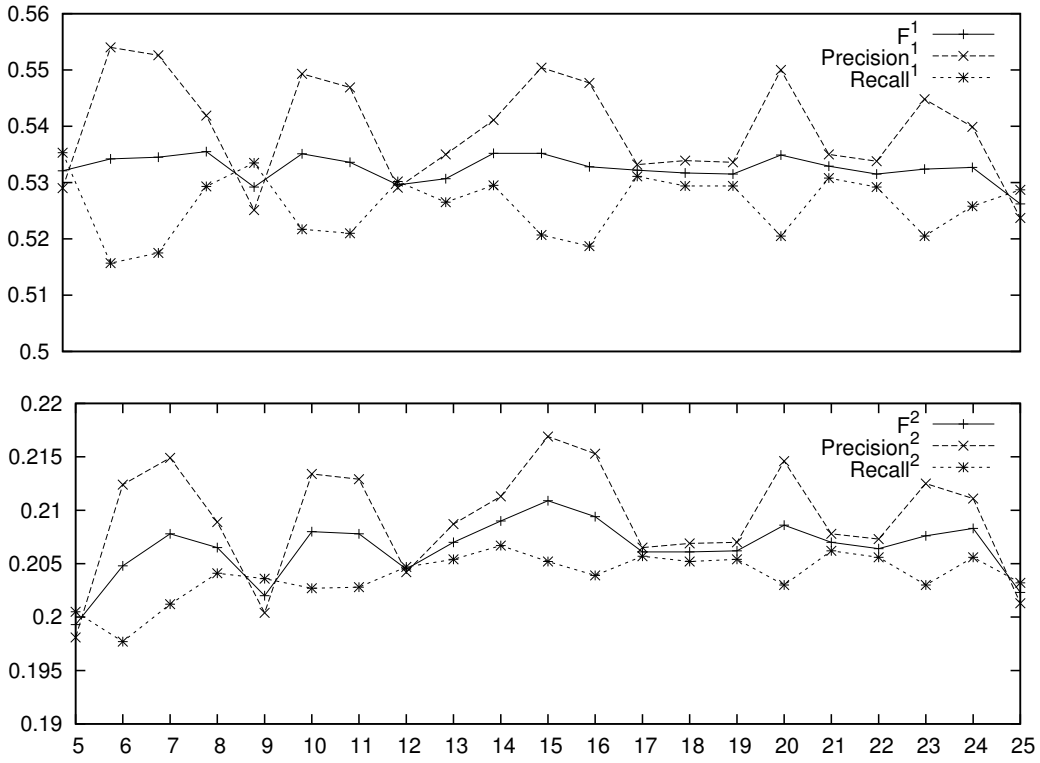


Figure 5.13: Forest permissive labeling English-Hindi translation accuracy on development set across iterations of the forest update target. The top graph shows precision/recall/f-measure with exponent 1, the bottom graph shows precision/recall/f-measure with exponent 2.



at most 15, and [Wellington et al.2006a] who require only a single pass over the bitext.

We also tracked the accuracy of the systems on the development set as training progressed. The graphs are presented in Figures 5.6-5.13. They track the translation accuracy on the development set from iteration 5 through 25. The first model was trained on iteration 5, using exploration threshold 0. Iterations 6-10 were trained with exploration threshold 1, iterations 11-15 with exploration threshold 2, etc. for five epochs in total. An interesting point stands out in comparing the translation quality of the English-French and English-Hindi translation systems as training proceeds. Increasing the exploration threshold improved the translation accuracies of the models for English-French translation using strict labeling, but it tended to degrade the translation accuracy for English-Hindi translation using strict labeling. We traced the accuracy of the model on the development set across iterations (refer to Figures 5.6, 5.10, 5.7, and 5.11) and for the English-French systems, we achieved significant gains after a single iteration where the exploration threshold was raised to 2. On the English-Hindi systems F^2, P^2, and R^2 more or less steadily decreased as training progressed and as the exploration threshold was raised. The decline in accuracy suggests that as training progressed, the models for 2x2 composition inferences, which model the relative order of how partial translations combine, got progressively worse. We conjecture that the quality of the bilingual composition inferences did not degrade because in practice all the relevant examples are generated by the coordinator at low exploration radii. We conjecture that the rarity of complete multitrees in the English-Hindi dataset may be responsible for this phenomenon. When we do not have a complete multitree for a bitext sentence pair, it is often because the supervisor is not able to infer some of the higher-level 2x2 composition inferences that are required to infer a complete multitree. Consequently, under the strict labeling strategy the training set might not contain many positive examples for higher-level 2x2 composition inferences, thereby reducing the predictive capabilities of the trained model. When translating, the translator will always be able to infer a complete multitree, since it is unconstrained by word alignments. Thus, the distribution of higher-level 2x2 composition inferences during test time will be different than during training. The English-Hindi systems trained using permissive labeling probably do not suffer from this degradation of translation quality as much due to the fact that they can add high-level 2x2 composition inferences as positive examples even if they are not part of the multitree or multiforest inferable by the supervisor[10].

[10]Since they will be labeled as positive examples if they have WIQ > 0.

5.6 Analysis

In this section we analyze how the experimental conditions of our five systems impact the translations output by the system. We select representative examples of sentences in our development set to determine how the experimental conditions affect the inference process for the sentences. We focus on the output of the English-French systems[11].

5.6.1 No exploration vs. exploration

To illustrate how allowing exploration affects the behavior of the translator, we examine how the straw-man configuration and the baseline configuration each translate the following sentence from the English/French development set:

Input: *This aspect has been neglected in the motion for the resolution .*
Reference: *Cet aspect a été négligé dans la proposition de résolution .*
Straw-man: *Ce aspect a été négligées dans la proposition pour la résolution .*
Baseline: *Cet aspect a été négligées dans la proposition de la résolution .*

The straw-man output has three mistakes relative to the reference. First, the determiner "This" should be translated as "Cet" rather than "Ce", since the subsequent word begins with a vowel; second, the translation of "neglected" should be singular masculine rather than feminine plural; and third, "for the" should be translated as "de" rather than "pour la." Two of the translation errors are caused by the same phenomenon. Consider the translation of the noun phrase "This aspect". Among the items the translator considers, it infers ones with the following source/target yields and costs: $a_1 = [\text{This} \mid \text{Ce}; 1.360]$, $a_2 = [\text{This} \mid \text{Cet}; 3.336]$, and $b = [\text{aspect} \mid \text{aspect}; 2.598]$. The two pairs of antecedent items can be used in monotonic 2x2 composition inferences. Both the inference taking a_1 and b as antecedents and the inference taking a_2 and b as antecedents get a low cost of 0.484 under the straw-man configuration. The low cost is to be expected since in French determiners tend to precede nouns. This means that, due to the MinMerge-derivation semiring, the cost of the 2x2 composition inference was unlikely to have an impact on the cost of the final multitree output. The same problem affects the translation of "for" as "pour" rather than "de". The relevant classifiers give the translation of "for" to "pour" slightly lower cost (1.361) than the translation of "for" to "de" (1.588), but the compositions yielding both "pour la résolution" and

[11] Because the author speaks and understands French more insight was gleaned from examining the French translations.

"de la résolution" have very low cost (0.0299) since prepositions[12] always precede noun phrases[13] in French prepositional phrases[14].

We could potentially multiply the costs of all 2x2 compositions by some value to elevate their costs so that the cost is higher than the costs of bilingual composition inferences. However, even if the inference were to have the highest cost in the multitree it might still not help us decide between translating "This aspect" as "Ce aspect" or "Cet aspect." The composition classifiers are trained only to determine whether the target yields of the antecedents should be in monotonic or inverted order with respect to the source. The assumption of the straw-man configuration is that the classifiers for bilingual composition inferences are responsible for selecting the correct translationally-equivalent fuses, and the 2x2 composition inference classifiers only need to determine the relative ordering of the antecedents. The 2x2 composition classifiers are not trained to disambiguate between different sets of antecedents. Doing so would be fine if we can always rely on the classifiers of the bilingual composition inferences to determine which one to make. In this case, we would need to know the translation of "aspect" to determine whether to translate "This" as "Ce" or "Cet," since the determiner in French must agree with the noun in terms of gender and in terms of whether the noun begins with a vowel. The lowest inference in the multitree to which this information is available is the composition inference. In order for the translator to select between such sets of antecedent items, the classifier must learn from training examples for inferences whose antecedents are incorrect.

Under the single tree strict labeling training configuration, "This" is now correctly translated as "Cet" and "for" is correctly translated as "de". To see why, we examine the costs of the items involved in the inferences. Just as under the straw-man training configuration, the inference translating "This" as "Ce" has cost 1.361 and the inference translating "This" as "Cet" has cost 3.575. However, now the cost of composing "Ce" with "aspect" is much higher than the cost of composing "Cet" with aspect – the composition inference yielding "Ce aspect" has cost 3.976, whereas the composition inference yielding "Cet aspect" has cost 2.927. This is to be expected because during training, the classifier that assigns cost to the 2x2 composition will learn from inferences with antecedent items yielding "Ce" and "Cet". Thus, in addition to learning the relative order of the antecedents, the classifier is now trained to disambiguate between combinations of antecedent items. Since the dominant cost in the cost sequence of the item with yield "Cet aspect" is 3.575 and the dominant cost of the item with yield "Ce aspect" is 3.976, "This aspect" is correctly translated as "Cet aspect." Similarly, for translating "for" as "de" rather than "pour", the cost of "for" to "pour" is 1.029 and the cost of "for" to "de" is

[12]Constituents with nonterminal label `IN`.
[13]Constituents with nonterminal label `NP`.
[14]Constituents with nonterminal label `PP`.

1.587, but the cost of the inference yielding "pour la résolution" is 2.479 whereas the cost of the inference yielding "de la résolution" is 2.2247.

5.6.2 Density of the search space

To illustrate the difference in system behavior when we vary the density of the search space by configuring the supervisor to infer a single tree or forest, we consider the following example from the English-French development set:

> **Input:** *There are other countries , apart from ACP countries , which are dependent on exports of bananas .*
> **Reference:** *Il existe d' autres pays qui dépendent également de l' exportation des bananes .*
> **Tree update:** *Il sont d' autres pays , Hormis les pays ACP , qui sont dépendent sur les exportations de bananas .*
> **Forest update:** *Il existe d' autres pays , provenant pays ACP , qui dépendent des exportations de bananas .*

Although the reference is not a literal translation of the English source sentence, there are still portions of the output that we can improve. For example, the baseline tree-update system translates "are" as "sont" rather than "existe", (and "are dependent" as "sont dépendent" rather than the verb "dépendent".) We could conceivably learn to translate "There are" as "Il existe" because the training bitext contains the sentence pair

> **Input:** *There are five points I would like to highlight because I believe that sooner or later they will have to be reviewed :*
> **Reference:** *Il existe cinq points que je souhaite souligner car je pense que , tôt ou tard , ils devront être reconsidérés :*

The word alignment for this sentence pair does not align "existe" to any source word and does not align "are" to any target word. The multitree contains a constituent covering "There / Il existe" and a constituent covering "are five points / cinq points". Because we commit to a single multitree, we cannot learn a bilingual composition inference composing antecedents with yields "There are" and "Il existe". In fact, in the 6 sentence pairs in the training corpus containing the subsequence pair "There are / Il existe", four of the multitrees have a constituent covering "There / Il existe" with a different constituent covering "are", one of the multitrees has a constituent covering "There are / Il existe déjà", and one has a constituent translating "There" to "existe" and "are" to "Il". This means that we will not get any positive examples for inferences translating "There" as "Il" or "are" as "existe", and we will not observe any positive examples for inferences composing the source fuse "There are" to target fuse "Il existe." Therefore, these

bilingual compositions will not be in the inference lexicon.

Under the forest-update, the inference lexicon contains a term allowing a bilingual composition of a monolingual item with source yield "There are" and a monolingual item with target yield "Il existe", which was not available in the inference lexicon for the single tree update experiments. The inference lexicon also allows for a bilingual composition of "are dependent" with "dépendent", thereby correctly deleting "are". This advantage can be seen in the rise in translation accuracy, from F^1-measure of 0.4935 to 0.5025, and from F^2-measure of 0.2222 to 0.2287 in Table 5.2.

5.6.3 Strict vs. permissive labeling

A problem we observed with strict labeling was that when we trained the classifiers all incorrect inferences were treated as equally bad. The training scheme had no way to determine which incorrect inferences led to better translations. As soon as an incorrect inference is made, all subsequent inferences building upon the incorrect one will also be incorrect when using strict labeling. The translator is not taught to select between them.

This issue can be seen in the translation of the following sentence from the development set by the baseline single-tree strict labeling configuration:

Input: *Paragraph 54 of my report sets out the conditions for Parliament to agree the transfer of funds from the 1998 budget reserve .*
Reference: *Le paragraphe 54 de mon rapport fixe les conditions pour que le Parlement autorise le transfert de fonds de la réserve du budget 1998 .*
Single tree update with strict labeling: *Le paragraphe 54 de mon rapport contingents applicables jusque les les conditions pour le Parlement à accord le transfert des fonds partir de l' 1998 réserve budgétaire .*
Single tree update with permissive labeling: *Le paragraphe 54 de mon rapport les fixe les conditions pour le Parlement de accord le transfert des fonds de réserve budgétaire 1998 la .*

There are many mistakes in the output of the systems, but we focus on the translation of the substring "sets out the conditions". The baseline translator translates it as "contigent applicables jusque les les conditions", whereas the reference translates it as "fixe les conditions." When we look at the items output by both systems, however, the least cost translation for "sets out" is "les fixe", and the least cost translation for "the conditions" is the correct translation "les conditions." The inference lexicon does not contain a fuse pair translating "sets out" to "fixe". However, both translations "contigent applicables jusque les les conditions" and "les fixe les conditions" are incorrect with respect to the reference. Both compositions get very high cost under the model trained with strict labeling, because the

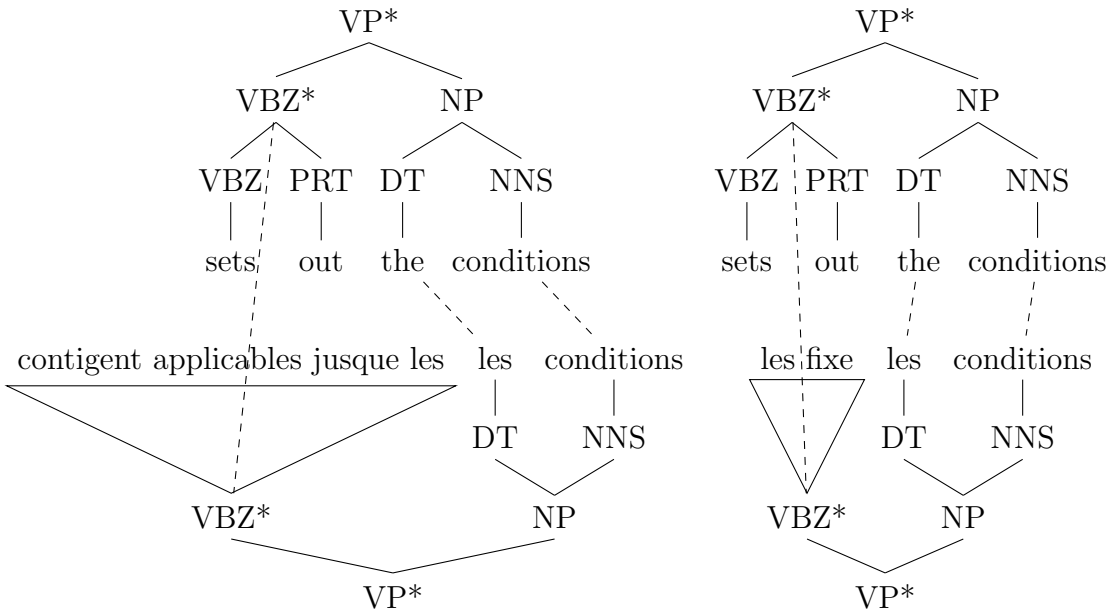
most highly-weighted feature is the frequency of the least frequent trigram. The least-frequent trigram in both translation outputs has frequency 0 in the language model. The classifiers, therefore, will give both translations negative confidences, since both are incorrect. There are no signals to the classifiers that a translation containing "les fixe les conditions" should be preferred over a translation containing "contigent applicables jusque les les conditions." This problem is due to the strict labeling strategy, since it cannot determine that "les fixe les conditions" is better than the alternative translation that was selected.

As we can see, under the permissive labeling strategy "sets out the conditions" is translated as "les fixe les conditions", which, though not completely correct, is a much better translation than "contigent applicables jusque les les conditions." To see why the alternative is selected, consider the sub-multitrees for the two outputs in Figure 5.14. Both multitrees are inferred by the two models with different costs. Under the strict labeling strategy, the highest cost inference for the tree on the left is the one composing "contigent applicables jusque les" with "les conditions" which has cost 4.2853. The highest cost inference for the tree on the right is the one composing "les fixe" with "les conditions", which has cost 5.3702. Under the permissive labeling strategy, the highest-cost inference for the tree on the left is the one composing "contigent applicables jusque les" with "les conditions" with cost 4.1833, and the highest-cost inference for the tree on the right is the one composing "les fixe" and "les conditions" with cost 2.9031. Under the strict labeling strategy, both compositions would be labeled as negative examples during training, since neither is part of the reference translation. However, utilizing WIQ1 in the permissive labeling strategy allows us to determine that the composition of "les fixe" and "les conditions" improves the translation quality of the consequent relative to the antecedents, whereas the composition of "contigent applicables jusque les" and "les conditions" does not. The e measure of "les fixe les conditions" is $\sqrt{3^2} - \frac{\sqrt{1^2}+0}{2} - 0 = 2.5$, whereas maximum e measure of the antecedents is 2 (for "les conditions" which matches the reference). On the other hand, the e measure of "contigent applicables jusque les les conditions" is $\sqrt{2^2} - \frac{\sqrt{4^2}+0}{2} - 0 = 0$, whereas the maximum e measure of the antecedents is 2. This is a prime example of how the permissive labeling strategy, along with WIQ1, gives the model a preference among inferences that do not yield the reference translation.

5.6.4 Feature analysis

To get a sense of how the global training objective affected the induced classifiers, we examined the model weights of the classifiers induced under the five training objectives. We wish to gain some insight as to how the different global objectives affect the predictive power of the atomic features. Recall that GBDT training algorithm computes weights for *compound features* (CFs) – that is, the

Figure 5.14: Two multitrees covering "sets out the conditions". The one on the left is the sub-multitree output by the model trained on single tree updates with strict labeling, and the one on the right is the sub-multitree output by the model trained on single tree updates with permissive labeling, but both trees are inferable by both.



conjunction of features along the path from the root to each node in each decision tree. For each CF $a \wedge b \wedge c$, we can compute the provisional confidence of CF $a \wedge b$ if it were a leaf, and take the difference between the parameter weights of $a \wedge b \wedge c$ and the parameter weights of $a \wedge b$ to get an estimate of the utility of splitting on atomic feature c. We assign each CF into a class **c** based on the type of feature c that it last splits on. We then accumulate the magnitude of the difference in confidences of each CF in **c** and normalize by the sum of the magnitude of the difference in confidences to give us a distribution of the positive and negative confidence masses over feature types. To perform the analysis, we randomly selected 2000 classifiers for bilingual composition inferences and 2000 classifiers for 2x2 composition inferences from the English-French experiments. For each classifier, we selected the decision trees induced under the optimal regularizer for each translation system. The results are presented in Tables 5.7 – 5.11.

The first thing to notice is the difference between feature classes that are most heavily weighted by bilingual composition classifiers versus the classes that are most heavily weighted by classifiers for 2x2 compositions. Classifiers for bilingual compositions rely more on the predictive capability of lexical and POS features such as window features (e.g. "the source word 2 positions left of the consequent is X") or dependency features (e.g. "the head of the sibling of the maximal projection is Y"). 2x2 composition classifiers seem to rely mostly on language model information, such as the minimum or average uni/bi/trigram count and minimum uni/bi/trigram informativeness. The second aspect to notice is that by-and-large, the same atomic feature types are used for prediction regardless of system. Interestingly, there don't seem to be significant differences between the distributions of feature types parameterized by the different systems.

We also compared the average magnitudes of confidences of feature types between the systems. The magnitude of confidences given to features in bilingual composition inference classifiers tended to be similar for the five systems. The average magnitude of confidences for feature types weighted by 2x2 composition inference classifiers had a marked difference however, especially between the systems trained using strict labeling and those trained using permissive labeling. In systems trained using permissive labeling, the highest magnitude confidence weights, which were on language model features, tended to be lower than the highest magnitude confidence weights of the models trained using permissive labeling. The average magnitude confidence for the 2x2 composition classifiers trained on single tree updates with strict labeling was 0.36 for features with positive confidences and 0.389 for features with negative confidences, whereas the average magnitude confidence for the 2x2 composition classifiers trained on single tree updates with permissive labeling was 0.208 for features with positive confidences and 0.224 for features with negative confidences. The average magnitude confidence for the 2x2 composition classifiers trained on forest updates with strict labeling was 0.313 for features with

Table 5.7: Most frequently used feature types in straw-man English-French experiment

Bilingual Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Label of source window word =	0.2713	Label of source window word ≠	0.2980
Source window word =	0.2511	Source window word ≠	0.2350
Source dependent word =	0.1662	Source dependent word ≠	0.1628
Source dependent NT label =	0.1608	Source dependent NT label ≠	0.1609
Source dependent POS tag =	0.0816	Source dependent POS tag ≠	0.0823
Number of siblings of maximum projection >	0.0356	Number of siblings of maximum projection ≤	0.0307
Number of children of maximum projection >	0.0224	Number of children of maximum projection ≤	0.0197
Source last token =	0.0052	Source last token ≠	0.0054
Position of dependent =	0.0032	Position of dependent ≠	0.0025
Source last POS tag =	0.0004	Source last POS tag ≠	0.0004
2x2 Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Language model average frequency >	0.2303	Language model average frequency ≤	0.1867
Language model informativeness >	0.1125	Label of source window word ≠	0.1191
Label of source window word =	0.1036	Source window word ≠	0.1058
Source dependent NT label =	0.0943	Language model informativeness ≤	0.1021
Language model minimum frequency >	0.0902	Source dependent NT label ≠	0.1003
Source window word =	0.0870	Source dependent word ≠	0.0974
Source dependent word =	0.0800	Language model minimum frequency ≤	0.0807
Yield length ratio >	0.0421	Target window word ≠	0.0481
Target window word =	0.0388	Source dependent POS tag ≠	0.0405
Source dependent POS tag =	0.0357	Yield length ratio ≤	0.0289
Target dependent word =	0.0185	Target dependent word ≠	0.0247
Production =	0.0181	Production ≠	0.0194
Length of sequence of same production >	0.0173	Length of sequence of same production ≤	0.0170
Number of children of maximum projection >	0.0111	Number of children of maximum projection ≤	0.0107
Number of siblings of maximum projection >	0.0108	Number of siblings of maximum projection ≤	0.0104
Target dependent label =	0.0054	Target dependent label ≠	0.0041
Dependent position =	0.0017	Dependent position ≠	0.0016
Source last token =	0.0015	Source last token ≠	0.0013
Source last POS tag =	0.0003	Source last POS tag ≠	0.0001

Table 5.8: Most frequently used feature types in the English-French single-tree update with strict labeling experiment

Bilingual Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Label of source window word =	0.2775	Label of source window word ≠	0.3102
Source window word =	0.2531	Source window word ≠	0.2290
Source dependent word =	0.1750	Source dependent label ≠	0.1601
Source dependent label =	0.1515	Source dependent word ≠	0.1569
Source dependent POS tag =	0.0794	Source dependent POS tag ≠	0.0809
Number of siblings of maximum projection >	0.0360	Number of siblings of maximum projection ≤	0.0346
Number of children of maximum projection >	0.0181	Number of children of maximum projection ≤	0.0185
Source last token =	0.0038	Source last token ≠	0.0047
Dependent position =	0.0027	Dependent position ≠	0.0024
Source last POS tag =	0.0008	Source last POS tag ≠	0.0010
2x2 Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Language model average frequency >	0.2622	Language model average frequency ≤	0.2331
Source window word =	0.0994	Target window word ≠	0.1116
Language model informativeness >	0.0987	Source window word ≠	0.1073
Target window word =	0.0942	Language model informativeness ≤	0.099
Source dependent word =	0.0798	Source dependent word ≠	0.0858
Label of source window word =	0.0761	Label of source window word ≠	0.0720
Language model minimum frequency >	0.0692	Language model minimum frequency ≤	0.0702
Source dependent label =	0.0635	Source dependent label ≠	0.0613
Target dependent word =	0.0412	Target dependent word ≠	0.0522
Yield length ratio >	0.0322	Source dependent POS tag ≠	0.0261
Source dependent POS tag =	0.0271	Production ≠	0.0248
Production =	0.0223	Yield length ratio ≤	0.0244
Production >	0.0146	Production ≤	0.0139
Number of siblings of maximum projection >	0.0066	Number of siblings of maximum projection ≤	0.0062
Number of children of maximum projection >	0.0060	Number of children of maximum projection ≤	0.0057
Target dependent label =	0.0043	Target dependent label ≠	0.0037
Source last token =	0.0011	Source last token ≠	0.0010
Dependent position =	0.0010	Dependent position ≠	0.0009
Source last POS tag =	0.0002	Source last POS tag ≠	0.0002

Table 5.9: Most frequently used feature types in the English-French forest update with strict labeling experiment

Bilingual Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Label of source window word =	0.2878	Label of source window word ≠	0.3001
Source window word =	0.2394	Source window word ≠	0.2502
Source dependent label =	0.1750	Source dependent label ≠	0.1615
Source dependent word =	0.1458	Source dependent word ≠	0.1574
Source dependent POS tag =	0.0735	Source dependent POS tag ≠	0.0720
Number of siblings of maximum projection >	0.0426	Number of siblings of maximum projection ≤	0.0318
Number of children of maximum projection >	0.0280	Number of children of maximum projection ≤	0.0209
Source last token =	0.0051	Source last token ≠	0.0039
Dependent position =	0.0013	Dependent position ≠	0.0011
Source last POS tag =	0.0005	Source last POS tag ≠	0.0002
2x2 Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Language model average frequency >	0.2441	Language model average frequency ≤	0.2170
Target window word =	0.1006	Target window word ≠	0.1128
Source window word =	0.0874	Source window word ≠	0.0911
Source dependent word =	0.0827	Source dependent word ≠	0.0844
Source dependent label =	0.0813	Language model informativeness ≤	0.0832
Label of source window word =	0.0797	Target dependent word ≠	0.0800
Language model informativeness >	0.0788	Source dependent label ≠	0.0798
Target dependent word =	0.0649	Label of source window word ≠	0.0765
Language model minimum frequency >	0.0583	Language model minimum frequency ≤	0.0612
Yield length ratio >	0.0475	Yield length ratio ≤	0.0400
Source dependent POS tag =	0.0228	Source dependent POS tag ≠	0.0228
Production >	0.0180	Production ≠	0.0193
Production =	0.0175	Production ≤	0.0163
Number of siblings of maximum projection >	0.0054	Number of siblings of maximum projection ≤	0.0058
Number of children of maximum projection >	0.0050	Number of children of maximum projection ≤	0.0048
Target dependent label =	0.0037	Target dependent label ≠	0.0027
Source last token =	0.0009	Source last token ≠	0.0009
Dependent position =	0.0009	Dependent position ≠	0.0008
Source last POS tag =	0.0001	Source last POS tag ≠	0.0001

Table 5.10: Most frequently used feature types in the English-French single tree update with permissive labeling experiment

Bilingual Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Label of source window word =	0.2915	Label of source window word ≠	0.3026
Source window word =	0.2320	Source window word ≠	0.2435
Source dependent label =	0.1775	Source dependent label ≠	0.1658
Source dependent word =	0.1455	Source dependent word ≠	0.1484
Source dependent POS tag =	0.0853	Source dependent POS tag ≠	0.0801
Number of siblings of maximum projection >	0.0322	Number of siblings of maximum projection ≤	0.0285
Number of children of maximum projection >	0.0238	Number of children of maximum projection ≤	0.0216
Source last token =	0.0061	Source last token ≠	0.0042
Dependent position =	0.0031	Dependent position ≠	0.0026
Source last POS tag =	0.0007	Source last POS tag ≠	0.0004
2x2 Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Language model average frequency >	0.2720	Language model average frequency ≤	0.2134
Target window word =	0.1157	Target window word ≠	0.1367
Source window word =	0.0864	Source window word ≠	0.1043
Language model informativeness >	0.0863	Language model informativeness ≤	0.0851
Label of source window word =	0.0734	Source dependent word ≠	0.0846
Source dependent word =	0.0724	Label of source window word ≠	0.0792
Source dependent label =	0.0666	Source dependent label ≠	0.0686
Language model minimum frequency >	0.0648	Target dependent word ≠	0.0653
Target dependent word =	0.0538	Language model minimum frequency ≤	0.0628
Yield length ratio >	0.0284	Source dependent POS tag ≠	0.0268
Production =	0.0252	Production ≠	0.0258
Source dependent POS tag =	0.0245	Yield length ratio ≤	0.0202
Production >	0.0128	Production ≤	0.0111
Number of siblings of maximum projection >	0.0064	Number of siblings of maximum projection ≤	0.0063
Number of children of maximum projection >	0.0054	Number of children of maximum projection ≤	0.0050
Target dependent label =	0.0034	Target dependent label ≠	0.0026
Dependent position =	0.0011	Dependent position ≠	0.0008
Source last token =	0.0011	Source last token ≠	0.0007
Source last POS tag =	0.0002	Source last POS tag ≠	0.0001

Table 5.11: Most frequently used feature types in the English-French forest update with permissive labeling experiment

Bilingual Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Label of source window word =	0.3077	Label of source window word ≠	0.3264
Source window word =	0.2348	Source window word ≠	0.2322
Source dependent label =	0.1582	Source dependent label ≠	0.1542
Source dependent word =	0.1499	Source dependent word ≠	0.1463
Source dependent POS tag =	0.0775	Source dependent POS tag ≠	0.0738
Number of siblings of maximum projection >	0.0311	Number of siblings of maximum projection ≤	0.0322
Number of children of maximum projection >	0.0291	Number of children of maximum projection ≤	0.0254
Dependent position =	0.0057	Dependent position ≠	0.0034
Source last token =	0.0030	Dependent position ≠	0.0029
Source last POS tag =	0.0005	Source last POS tag ≠	0.0005
2x2 Compositions			
Fraction of positive confidence mass		Fraction of negative confidence mass	
Language model average frequency >	0.2743	Language model average frequency ≤	0.2565
Target window word =	0.1126	Target window word ≠	0.1162
Target dependent word =	0.0875	Target dependent word ≠	0.0960
Source window word =	0.0777	Source window word ≠	0.0828
Source dependent word =	0.0757	Source dependent word ≠	0.0763
Source dependent label =	0.0741	Source dependent label ≠	0.0727
Label of source window word =	0.0724	Language model informativeness ≤	0.0702
Language model informativeness >	0.0654	Source window label ≠	0.0689
Language model minimum frequency >	0.0509	Language model minimum frequency ≤	0.0547
Yield length ratio >	0.0430	Yield length ratio ≤	0.0387
Source dependent POS tag =	0.0191	Source dependent POS tag ≠	0.0193
Production >	0.0187	Production ≤	0.0176
Production =	0.0159	Production ≠	0.0174
Number of children of maximum projection >	0.0041	Number of siblings of maximum projection ≤	0.0044
Number of siblings of maximum projection >	0.0040	Number of children of maximum projection ≤	0.0041
Target dependent label =	0.0027	Target dependent label ≠	0.0022
Dependent position =	0.0008	Source last token ≠	0.0008
Source last token =	0.0007	Dependent position ≠	0.0007
Source last POS tag =	0.0002	Source last POS tag ≠	0.0002

positive confidences and 0.271 for features with negative confidences, whereas the average magnitude confidence for the 2x2 composition classifiers trained on single tree updates with permissive labeling was 0.127 for features with positive confidences and 0.178 for features with negative confidences. The features that were most affected by this difference were the language model features, which tended to be the features which had the most impact on the 2x2 composition inferences that became part of the multitree. Using permissive labeling seems to soften the predictions made by the classifiers, and also potentially reduces the impact of 2x2 compositions on the score of the final multitree.

5.7 Lesioning Experiment: MinMerge vs. Viterbi Semiring

In addition to evaluating the effect of the objective functions, amount of exploration, and search space density on translation accuracy, we also ran a lesioning experiment to measure the effect of the choice of semiring on translation accuracy. We selected the experimental system that gave us the greatest change over the straw-man experiment, namely the English-French system trained on a forest with permissive labeling. We trained two systems according to this objective. One of the systems' supervisor, predictor, and test translator were configured to use the Viterbi-derivation semiring. The other's supervisor, predictor, and test translator were configured to use the MinMerge-derivation semiring. The accuracies of the two configurations are shown in Table 5.12. The unigram precision of the system trained under the Viterbi-derivation semiring is slightly higher than the unigram precision of the MinMerge-derivation system, but the two recall measures and two F-measures of the MinMerge-derivation system are higher than than those of the Viterbi-derivation. The optimal regularizer λ for the Viterbi-derivation system was 0.01171875, whereas the optimal λ for the MinMerge-derivation system was 0.0234375.

The average length of translations output by the Viterbi-derivation system was 20.52 words per sentence, and the average length of translations output by the MinMerge-derivation system was 21.47 per sentence, potentially explaining the slightly higher unigram precision and the lower unigram recall under the Viterbi-derivation semiring. We conjecture that another factor in the increase in accuracy between the Viterbi-derivation and MinMerge-derivation semirings might be that the objective defined by the MinMerge-derivation semiring used during training[15] is better correlated with the evaluation measure than the objective under the Viterbi-derivation semiring, which minimizes the sum of inference costs.

[15]The objective that maximizes the minimum margin.

Table 5.12: Translation accuracies of the English-French forest permissive-labeling system using the Viterbi-derivation semiring and the MinMerge-derivation semiring. Numbers in bold are statistically significant over the Viterbi-derivation experiment at p=0.01 using the Wilcoxon signed rank test

English-French Translation				
			Test	
	Exponent	**Prec**	**Rec**	**FMS**
Viterbi-derivation	1	0.5204	0.4830	0.5010
semiring	2	0.2285	0.2120	0.2200
MinMerge-derivation	1	0.5197	**0.5015**	**0.5105**
semiring	2	**0.2384**	**0.2300**	**0.2341**

Chapter 6

Conclusions

6.1 Summary

In this dissertation, we have presented a novel fully discriminative approach for training machine translation systems. We approached the problem by having the translators learn a search process by which they can infer high quality translations. Developing this approach required many innovations. Our three primary contributions have been as follows:

- We have described the architecture and algorithms of an example generation framework that gives us a method to both select an approved set of search paths through the hypergraph and to learn from the set of paths. Our parsing coordinator permits us to explicitly control the amount of exploration performed by the predictor, thereby allowing us to select which inferences to learn from. Our architecture also allows us to control the density of the space of possible translations by selecting the kind and number of derivation path(s) we wish for the model to learn from. The density is controlled by specifying the set of multitrees the supervisor is allowed to infer. The parsing coordinator allows the supervisor and predictor to lazily generate only the inferences needed to train the model. Due to the way that the supervisor guides the predictor, we can take advantage of early update [Collins and Roark2004, Daumé and Marcu2005] and its generalizations while still allowing us to explore the space of incorrect translations.

- We have presented a new semiring allowing us to compare translations in a manner that, we conjecture, is closer to how a human compares translations, based on the most egregious error in each translation. The MinMerge-derivation semiring allows us to do more specific blame assignment over the Viterbi-derivation semiring since we can determine the worst inference involved in inferring a translation by looking at the first element in the cost se-

quence. Finally, the MinMerge-derivation semiring gives us an unbiased way to compare posets of inferences since, unlike the Viterbi-derivation semiring, it is not biased towards shorter translation derivations.

- We presented a new whole inference quality class of objective functions that measures the gain in translation accuracy achieved by firing an inference. Optimizing the translation model on a WIQ objective allows the translator to learn the merit of parser states relative to the merit of their predecessor states, thereby allowing the translator to learn which incorrect inferences are better than other incorrect inferences and recover from mistakes.

We have empirically demonstrated that our new objective function, WIQ1, brings about an improvement over a tree-structured fully-discriminative baseline using strict labeling, and have demonstrated how the parsing coordinator architecture facilitates controlled experimentation with the objectives and search space densities. While we focused on tree-structured SMT, we believe our approaches can be applied to finite-state SMT by configuring the supervisor and predictor to utilize finite state grammars rather than context free grammars. Furthermore, we conjecture that our approach can be used for other structured prediction problems as well.

6.2 Future Work

The major disadvantage of our approach so far is the difficulty in scaling up to train on corpora with millions of sentence pairs. This disadvantage is due to translating the entire training set on each epoch. One way to overcome this disadvantage is to train the bilingual composition inferences on a bitext of millions of sentence pairs, which does not require search over the entire search space. We can then select a smaller subcorpus to train the 2x2 composition inferences in the manner we described in this work. This approach is in line with the mildly discriminative approaches of [Och2003, Chiang et al.2008] where they learn most of their parameters on a large corpus but tune the parameters using the minimum error rate or max-margin objective on a smaller tuning corpus. In our case, however, the approach would still be fully discriminative.

The other major modification we would make would be to train the inference lexicon on the same objective on which the classifiers are trained. Currently, we precompute the inference lexicon based on the word alignment probabilities and the fertility probabilities. We described how the ideal training procedure would reestimate the inference lexicon as well as the classifiers, but noted the computational cost of doing so. Addressing this issue would allow all the parameters of our system to be optimized for the same objective.

6.3 Other applications

In Chapter 1 we conjectured that the methods presented in this dissertation could be applied to a wide variety of structured prediction problems. Many problems in computer science can be formulated as structured classification problems, including natural language parsing, named entity recognition, handwriting recognition, image segmentation, and robot navigation. In this section, we present a high-level overview of how the ideas behind the parsing coordinator and the objective functions presented in this work could be applied to these problems.

Parsing In supervised constituency and dependency parsing, the task is to infer a tree structure over a single input sentence. In constituency parsing, each parse tree is a possible label $\mathcal{Y}$ consisting of bracket/non-terminal label pairs $\mathcal{Y}_1 \times \ldots \times \mathcal{Y}_k$ with the constraint that no two brackets may partially overlap without one containing the other, that there exist a bracket for each word in the input sentence, and that there exist a bracket covering the entire sentence. In dependency parsing, each dependency tree is a possible label $\mathcal{Y}$ consisting of head/dependent word pairs $\mathcal{Y}_1 \times \ldots \times \mathcal{Y}_k$, with the constraint that all but one word is the dependent of another word. The one word that is not a dependent of any other words is the head of the entire sentence. Many approaches to parsing infer $\mathcal{Y}$ by firing inferences to determine the value of each $\mathcal{Y}_i$ that minimizes some cost function. Typically, the value for each bracket/non-terminal label or head/dependent pair $\mathcal{Y}_i$ is inferred given a subset of random variables corresponding to the labels on antecedent items (e.g. [Taskar et al.2004b,Bikel2004,Turian and Melamed2006,McDonald et al.2005]). In supervised parsing the training set consists of a treebank containing a parse tree $\mathcal{Y} = \hat{\mathbf{y}}$ for each sentence in the corpus. The task is to learn a process by which the correct "gold-standard" parse tree $\hat{\mathbf{y}}$ can be inferred for a sentence.

Given that the SMT method presented in this work is an instance of generalized parsing, it is possible to train a constituency parser in the same manner as the translator in this work. Additionally, we conjecture that we could train a dependency parser in the exact same manner with an appropriate specification of item and inference signatures. We could utilize the parsing coordinator with a supervisor constrained to infer the reference parse tree and the predictor constrained to infer any tree over the input sentence. One could then utilize both strict and permissive labeling strategies to determine whether the inferences inferring each bracket/non-terminal label or head/dependent pair y_k should be labeled as positive or negative examples. The strict labeling strategy would work just as described in this dissertation, by checking whether an inference infers an item $y_k \in \hat{\mathbf{y}}$. To use the permissive labeling strategy, we would need to define an appropriate WIQ measure. For constituency parsing, WIQ could perhaps compare the number of crossed or matching brackets in the subtree rooted at a consequent item

relative to the number of crossed or matching brackets of the antecedent subtrees. For dependency parsing, WIQ could compare the number of correct and incorrect dependency relations in the subtree rooted at a consequent item relative to the number of correct and incorrect dependency relations in its antecedents. Given the examples generated by the parsing coordinator and the labels assigned by the labeling strategy, we could then train binary classifiers to discriminate between inferences that are useful in inferring a good parse tree and those that are not [Turian and Melamed2006].

Named entity recognition Another potential application of our techniques is in named entity recognition. In NER, the task is to find all named entities in an input document and to label the entities with their type. For example, given the sentence "The Bank of New York Mellon is located in New York City", "Bank of New York Mellon" should be labeled as an organization, and "New York City" should be labeled as a location. The set of possible label sequences for an input sentence x is $\mathcal{Y}$, and each $\mathcal{Y}_i \in \mathcal{Y}$ can take entity type label $y_{i,1}, \ldots, y_{i,\ell}$. Many approaches to NER have been studied (refer to [Nadeau and Sekine2007] for a survey). A common approach is to use graphical models, such as sequential Markov Networks or Conditional Random Fields to encode dependencies between the labels $\mathcal{Y}$, and then use the Viterbi algorithm to find the most-likely or least-cost sequence of labels $\hat{\mathbf{y}}$. With an appropriate definition of items and inferences, our approach for SMT could be utilized to train labelers. For NER, we could define two types of items: items representing a word x_i and items representing a label y_i for word x_i. The specification of inferences would depend on the type of model. If we make a Markov assumption, an inference would take word x_i and label y_{i-1} as antecedents and infer label y_i as a consequent[1]. We could then utilize the parsing coordinator architecture and the objective functions to train binary classifiers to make each inference. The supervisor could be constrained to predict the set of reference labels for each sentence in the corpus. The predictor would be allowed to label words according to its model's biases. Under the strict labeling strategy, any inference made that infers a correct label sequence would be labeled as a positive example, and all other inferences proposed by the predictor would be labeled as negative. We could also apply a permissive labeling strategy, using a WIQ measure with a task appropriate definition of e-score, for instance the difference between the number of correct and incorrect labels in a label sequence.

Handwriting recognition Like NER, the problem of handwriting recognition can be formulated as a sequence labeling problem, where for a segmented, handwritten text, the goal is to infer the true character each handwritten character

[1]Note that we could probably use this same specification of inferences without making Markov assumptions, with appropriately defined feature functions.

represents. The difference lies in the complexity of the input space. Many approaches have been applied to this problem, including Max-Margin Markov Networks [Taskar et al.2004a], Conditional Random Fields, and Hidden Markov Models (e.g. [Feng et al.2006]). Just like in NER, we could train the model to predict the label of each handwritten character by training a set of binary classifiers to assign a cost to each step in the process of inferring a label sequence. To infer training examples we could use the configuration of the parsing coordinator described for NER, with appropriate feature functions over sets of pixels and possible labels rather than text. We conjecture that we could utilize the same definition of WIQ that we proposed for NER.

Image segmentation The problem of image segmentation is also a structured prediction problem. The input x consists of a set of pixels. The set of possible outputs is task dependent, but it typically consists of a set of labels $\mathcal{Y}$, with each random variable in $\mathcal{Y}$ a label on each pixel. The label, for instance, could be whether the pixel is part of the foreground or background of an image, whether the pixel is part of an edge, or the type of object the pixel is a part of. A common approach is to first group pixels together into *segments*, and then label the pixels comprising entire segments. As in sequence labeling, a popular method is to model the dependencies between segments using graphical models, such as Conditional Random Fields or Hidden Markov Models (e.g. [He et al.2004]). Unlike sequence labeling, however, the topology of the graph is not necessarily sequential. In sequence labeling, each label $\mathcal{Y}_i$ typically depends on previous label $\mathcal{Y}_{i-1}$ or labels $\mathcal{Y}_1, \ldots, \mathcal{Y}_{i-1}$. In image segmentation, this assumption might be suboptimal. A common approach is to use Markov networks (e.g. [Koller and Friedman2009]) to encode the dependencies. Unfortunately, exact inference is intractable over Markov networks in general. Instead, it is common to use approximate inference methods, such as Markov Chain Monte-Carlo or Belief Propagation [Pearl1982] over the network.

While we have only experimented with exact inference methods for structured classification in this work, we hypothesize that the parsing coordinator architecture could be used with approximate methods too. Belief propagation iteratively updates *messages* between labels $\mathcal{Y}_c = \{\mathcal{Y}_{c,1}, \ldots, \mathcal{Y}_{c,\ell}\}$ participating in a clique of variables c. The messages encode the marginal probabilities of the variables in the clique. Belief propagation can be used to find the marginal probability of each value that each $\{\mathcal{Y}_{c,1}, \ldots, \mathcal{Y}_{c,\ell}\}$ can take, which we can use to find the label values that maximize the marginal probabilities. In the case of acyclic networks, belief propagation is equivalent to the Viterbi algorithm. In the case of both cyclic and acyclic networks, we believe that we can apply the coordinator architecture to generate training examples for discriminative classifiers to model the marginal probabilities. Since the supervisor is constrained to infer the reference value for each $\mathcal{Y}_{c,i}$, the

supervisor's inference procedure would not need to use belief propagation, whereas the predictor would. Apart from the inference procedure, however, the supervisor and predictor, and their interaction, would be the same as for sequence labeling. Strict and permissive labeling could be applied just as for sequence labeling.

Robot navigation In robot navigation, the task is to learn how to make sequences of actions in an environment to satisfy a given set of goals. Each action y undertaken by the robot results in a new state s, where s represents a snapshot of the robot's knowledge of its relationship to its environment. The task can be conceptualized as a structured prediction problem, where we wish to learn how to make an optimal sequence of decisions $\mathcal{Y}$ in an environment to achieve a goal. Each $\mathcal{Y}_i \in \mathcal{Y}$ would be a set of possible actions $\{y_{i,1}, \ldots, y_{i,\ell}\}$ that the robot can given its current state s. A common approach to solving this problem is reinforcement learning (e.g. [Gullapalli1992, Mahadevan and Connell1990]).

We conjecture that our coordinator architecture could be adapted to train a robot to navigate an environment as well. The predictor would be an instance of the robot exploring its environment. Given the robot's current state and steps potentially resulting from taking a particular action $y_{i,j}$ at each point in its action sequence, it can decide whether or not to execute $y_{i,j}$. This decision would be analogous to an inference made by the translator. How to configure the supervisor is less clear. One way to configure it would be to simulate a random sequence of actions that the robot can take from a set of states to arrive at a goal state. If the robot used as the predictor can make all action sequences simulated by the supervisor, we can use both strict and permissive labeling strategies. If the coordinator uses the strict labeling strategy, it would only need to check whether an action taken by the predictor is one taken by the supervisor. If it uses the permissive labeling strategy, we would need to define a measure estimating a state's distance from the goal. WIQ would be the difference between the distance from the robot's current state to the goal and the resulting state's distance to the goal upon taking an action. How we define distance would depend on the robot's task and its environment.

Chapter 7

Bibliography

[Ahrenberg2007] L. Ahrenberg. 2007. Lines: An english-swedish parallel treebank. In *Proceedings of the 16th Conference of Computational Linguistics (NOLADIA '07)*.

[Alshawi et al.1998] Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 1998. Automatic acquisition of hierarchical transduction models for machine translation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 41–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Alshawi et al.2000] Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite-state head transducers. *Comput. Linguist.*, 26(1):45 – 60.

[Arun and Koehn2007] Abhishek Arun and Philipp Koehn. 2007. Online learning methods for discriminative training of phrase based statistical machine translation. In *MT Summit XI*.

[Baker1979] James Baker. 1979. Trainable grammars for speech recognition. In *Speech communication papers presented at the 97th meeting of the Acoustical Society of America*, pages 547–550, Cambridge, MA, June. MIT.

[Bengio et al.2009] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum learning. In *International Conference on Machine Learning*.

[Berger et al.1996] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), March.

[Bikel2004] Daniel M. Bikel. 2004. Intricacies of collins' parsing model. *Comput. Linguist.*, 30(4):479–511, December.

[Blunsom et al.2008] Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio, June. Association for Computational Linguistics.

[Brown et al.1988] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. 1988. A statistical approach to language translation. In *Proceedings of the 12th conference on Computational linguistics - Volume 1*, COLING '88, pages 71–76, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Brown et al.1993] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19:263–311, June.

[Burbank et al.2005] A. Burbank, M. Carpuat, S. Clark, M. Dreyer, P. Fox, D. Groves, K. Hall, I. D. Melamed, Y. Shen, A. Way, B. Wellington, and D. Wu. 2005. Final report of the 2005 language engineering workshop on statistical machine translation by parsing. Technical report, Johns Hopkins University 2005 Summer Workshop.

[Callison-Burch et al.2006] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation of the role of bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

[Caraballo and Charniak1998] S. Caraballo and E. Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Comput. Linguist.*, 24(2).

[Charniak and Johnson2005] Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Charniak1997] Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI*, pages 598–603.

[Cherry and Lin2007] Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical*

Translation, pages 17–24, Rochester, New York, April. Association for Computational Linguistics.

[Chiang et al.2008] David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, Hawaii, October. Association for Computational Linguistics.

[Chiang et al.2009] David Chiang, Kevin Knight, and Wei Wang. 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*, pages 218–226, Boulder, Colorado, June. Association for Computational Linguistics.

[Chiang2005] David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Chiang2007] David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2), June.

[Chiang2010] David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July. Association for Computational Linguistics.

[Church1988] K. W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143. ACL.

[Collins and Roark2004] Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

[Collins et al.2002] Michael Collins, Robert E. Schapire, and Yoram Singer. 2002. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1–3):253–285.

[Collins1999] Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

[Collins2002] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

[Cowan et al.2006] Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 232–241, Sydney, Australia, July. Association for Computational Linguistics.

[Daumé and Marcu2005] Hal Daumé, III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 169–176, New York, NY, USA. ACM.

[Daumé III et al.2009] Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Mach. Learn.*, 75:297–325, June.

[Dempster et al.1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. In *Journal of the Royal Statistical Society*, volume 39(B), pages 1–38.

[Ding and Palmer2005] Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 541–548, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Doddington2002] G Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *HLT 2002, Human Language Technology conference*, San Diego, California.

[Earley1970] Jay Earley. 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February.

[Eisner2003] Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, Sapporo, Japan, July. Association for Computational Linguistics.

[Feng et al.2006] Shaolei Feng, R. Manmatha, and Andrew Mccallum. 2006. Exploring the use of conditional random field models and hmms for historical handwritten document recognition. In *the Proceedings of the 2nd IEEE International Conference on Document Image Analysis for Libraries (DIAL*, pages 30–37.

[Galley et al.2004] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

[Galley et al.2006] Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia, July. Association for Computational Linguistics.

[Gentile2002] C. Gentile. 2002. A new approximate maximal margin classification algorithm. *JMLR*, 2.

[Goodman1999] J. Goodman. 1999. Semiring parsing. *Comput. Linguist.*, 25(4), December.

[Graehl and Knight2004] Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 105–112, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

[Gullapalli1992] V. Gullapalli. 1992. *Reinforcement Learning and its application to control*. Ph.D. thesis, University of Massachusetts.

[Haghighi et al.2009] Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. 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 923–931, Suntec, Singapore, August. Association for Computational Linguistics.

[He et al.2004] Xuming He, R.S. Zemel, and M.A. Carreira-Perpinan. 2004. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–695 – II–702 Vol.2, june-2 july.

[Hearne and Way2003] Mary Hearne and Andy Way. 2003. Seeing the wood for the trees: data-oriented translation. In *MT Summit IX*, pages 165–172.

[Huang et al.2006] Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*.

[Huang et al.2011] Shujian Huang, Stephan Vogel, and Jiajun Chen. 2011. Dealing with spurious ambiguity in learning itg-based word alignment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 379–383, Portland, Oregon, USA, June. Association for Computational Linguistics.

[Huang2008] Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

[Kasami1965] T. Kasami. 1965. An efficient recognition and syntax algorithm for context-free languages. Technical Report Technical Report AF-CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

[Klein and Manning2001] D. Klein and C. D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing*, Beijing, China, October.

[Klein and Manning2003] D. Klein and C. D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, Edmonton, AB.

[Koehn and Hoang2007] Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June. Association for Computational Linguistics.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pages 48–54, Edmonton, May-June.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

[Koehn2004] Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP*

2004, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

[Koehn2005] Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit X*.

[Koller and Friedman2009] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models*. The MIT Press.

[Lavie and Agarwal2007] Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic, June. Association for Computational Linguistics.

[Liang et al.2006] Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July. Association for Computational Linguistics.

[Liu et al.2006] Yang (1) Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July. Association for Computational Linguistics.

[Liu et al.2009] Yang Liu, Yajuan Lü, and Qun Liu. 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*, pages 558–566, Suntec, Singapore, August. Association for Computational Linguistics.

[Liu et al.2010] Shujie Liu, Chi-Ho Li, and Ming Zhou. 2010. Discriminative pruning for discriminative itg alignment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 316–324, Uppsala, Sweden, July. Association for Computational Linguistics.

[Mahadevan and Connell1990] S. Mahadevan and J. Connell. 1990. Automatic programming of behavior-based robots using reinforcement learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Boston, Massachusetts.

[Marcu and Wong2002] Daniel Marcu and Daniel Wong. 2002. A phrase-based,joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 133–139. Association for Computational Linguistics, July.

[Marton and Resnik2008] Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011, Columbus, Ohio, June. Association for Computational Linguistics.

[McDonald et al.2005] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

[Megyesi et al.2008] B. Megyesi, B. Dahlqvist, E. Pettersson, and J. Nivre. 2008. Swedish-turkish parallel treebank. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-2008)*.

[Melamed and Wang2004] I. D. Melamed and W. Wang. 2004. Generalized parsers for machine translation. Technical report, New York University.

[Melamed et al.2003] I. Dan Melamed, Ryan Green, and Joseph Turian. 2003. Precision and recall of machine translation. In *Proceedings of HLT-NAACL 2003*.

[Melamed et al.2004] I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 661–668, Barcelona, Spain, July.

[Melamed2004] I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 653–660, Barcelona, Spain, July.

[Mi et al.2008] Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199, Columbus, Ohio, June. Association for Computational Linguistics.

[Mohri2002] Mehryar Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages, and Combinatorics*, 7(3):321–350.

[Nadeau and Sekine2007] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January. Publisher: John Benjamins Publishing Company.

[Neu and Szepesvari2009] Gergely Neu and Csaba Szepesvari. 2009. Training parsers by inverse reinforcement learning. *Machine Learning*, 77:303–337.

[Ng2004] Andrew Y. Ng. 2004. Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 78–, New York, NY, USA. ACM.

[Och and Ney2003] Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1), March.

[Och and Ney2004] Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4), December.

[Och and Weber1998] Franz Josef Och and Hans Weber. 1998. Improving statistical natural language translation with categories and rules. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 985–989, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Och et al.1999] Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

[Park and Hastie2007] Mee Young Park and Trevor Hastie. 2007. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677.

[Pauls et al.2010] Adam Pauls, Dan Klein, David Chiang, and Kevin Knight. 2010. Unsupervised syntactic alignment with inversion transduction grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 118–126, Los Angeles, California, June. Association for Computational Linguistics.

[Pearl1982] Judea Pearl. 1982. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second National Conference on Artificial Intelligence, AAAI-82*, pages 133 – 136.

[Petrov and Klein2007] Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

[Poutsma2000] Arjen Poutsma. 2000. Data-oriented translation. In *COLING-2000*, pages 635–641.

[Quirk et al.2005] Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Ratnaparkhi1999] A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34.

[Roark2001] B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Comput. Linguist.*, 27(2), June.

[Rosenblatt1958] F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychology Review*, 65.

[Russell and Norvig1995] S. J. Russell and P. Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

[Schapire and Singer1999] Robert E. Schapire and Yoram Singer. 1999. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

[Sogaard and Kuhn2009] Anders Sogaard and Jonas Kuhn. 2009. Empirical lower bounds on aligment error rates in syntax-based machine translation. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 19–27, Boulder, Colorado, June. Association for Computational Linguistics.

[Taskar et al.2004a] Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004a. Max-margin markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

[Taskar et al.2004b] Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. 2004b. Max-margin parsing. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 1–8, Barcelona, Spain, July. Association for Computational Linguistics.

[Tillmann and Zhang2005] Christoph Tillmann and Tong Zhang. 2005. A localized prediction model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 557–564, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Tillmann and Zhang2006] Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical mt. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 721–728, Sydney, Australia, July. Association for Computational Linguistics.

[Tillmann et al.1997] Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. 1997. A dp-based search using monotone alignments in statistical translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 289–296, Madrid, Spain, July. Association for Computational Linguistics.

[Tillmann2003] Christoph Tillmann. 2003. A projection extension algorithm for statistical machine translation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Turian and Melamed2006] Joseph Turian and I. Dan Melamed. 2006. Advances in discriminative parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 873–880, Sydney, Australia, July. Association for Computational Linguistics.

[Turian et al.2006] Joseph Turian, Benjamin Wellington, and I. Dan Melamed. 2006. Scalable discriminative learning for natural language parsing and translation. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS)*, Vancouver, BC.

[Vapnik1995] Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer New York.

[Venkatapathy and Bangalore2009] Sriram Venkatapathy and Srinivas Bangalore. 2009. Discriminative machine translation using global lexical selection. 8(2):8:1–8:23, May.

[Vogel et al.1996] Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm based word alignment in statistical translation. In *Proceedings of the 34th Annual Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark, August.

[Wang et al.2007] Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754, Prague, Czech Republic, June. Association for Computational Linguistics.

[Watanabe et al.2007] Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic, June. Association for Computational Linguistics.

[Wellington et al.2006a] Benjamin Wellington, Joseph Turian, Chris Pike, and I. Dan Melamed. 2006a. Scalable purely-discriminative training for word and tree transducers. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 251–260. Cambridge, MA, August.

[Wellington et al.2006b] Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006b. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 977–984, Sydney, Australia, July. Association for Computational Linguistics.

[Wellington2007] Benjamin Wellington. 2007. *Tree-Structured Models of Multitext: Theory, Design and Experiments*. Ph.D. thesis, New York University, New York, NY, USA.

[Wu and Wong1998] Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the 17th international*

conference on Computational linguistics - Volume 2, COLING '98, pages 1408–1415, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Wu1997] D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguist.*, 23(3), September.

[Xiao et al.2009] Tong Xiao, Mu Li, Dongdong Zhang, Jingbo Zhu, and Ming Zhou. 2009. Better synchronous binarization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 362–370, Singapore, August. Association for Computational Linguistics.

[Xiao et al.2011] Xinyan Xiao, Yang Liu, Qun Liu, and Shouxun Lin. 2011. Fast generation of translation forest for large-scale smt discriminative training. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 880–888, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

[Yamada and Knight2001] Kenji Yamada and Kevin Knight. 2001. A syntax based statistical translation model. In *Proceedings of ACL-01.*

[Yamada and Knight2002] Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *Proceedings of ACL-02.*

[Younger1967] D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control.*

[Zhang and Gildea2005] Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 475–482, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Zhang et al.2006] Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263, New York City, USA, June. Association for Computational Linguistics.

[Zhang et al.2008] Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567, Columbus, Ohio, June. Association for Computational Linguistics.