University of Alberta

SYNTACTIC MOVEMENT IN STATISTICAL TRANSLATION

by

Colin Andrew Cherry

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2007

Your file Votre référence
ISBN: 978-0-494-32938-2
Our file Notre référence
ISBN: 978-0-494-32938-2

Abstract

Statistical machine translation (SMT) provides a framework to learn automatic translation systems from parallel texts. The statistical approach recasts translation and its subproblems as optimization problems, where one must find the best output as scored by an empirically-derived model. One complication shared by most translation subproblems is **movement**. While sentence order usually remains the same during translation, the order of the concepts within those sentences can vary drastically from language to language. If one assumes that concepts can move with complete freedom during translation, then the set of possible outputs can become very large. One can reduce this complexity by assuming that sentences have a context-free syntactic tree-structure, which explains all movement. This decomposes a sentence into subtrees, which define syntactic phrases that must exist in both the source and its translation. This thesis employs two syntactic constraints: an inversion transduction grammar (ITG) constraint that considers all possible binary trees, and a cohesion constraint that considers only a single tree, which is provided for one of the two languages.

We develop three distinct methods that use syntactic movement constraints to improve either the efficiency or accuracy of existing, non-syntactic solutions to SMT subproblems, such as alignment and decoding. The first is a phrasal ITG, which introduces an ITG constraint in order to gain polynomial-time algorithms for phrasal translation modeling. The resulting syntactic system improves performance over a comparable, flat-string model. The second project compares and combines ITG and cohesion constraints, as they are applied to bilingual word alignment. We present two combined alignment spaces, and show that a combination of ITG and cohesion constraints improves upon a comparable, bipartite-matching aligner. We also present a method to discriminatively train bitext parsers, allowing us to incorporate a powerful soft cohesion constraint into discriminative word-alignment.

Our third project defines cohesion on the translations output by a phrase-based decoder, given a source-side dependency tree. The resulting cohesive, phrase-based decoder is shown to produce translations that are preferred over non-cohesive output by both human evaluators and automatic metrics.

Acknowledgements

I would like to thank my supervisor Dekang Lin for his insights and support. I would also like to thank my network of sounding boards and proof readers for their advice, including Shane Bergsma, Chris Pinchak, Qin Wang, Chris Westbury, Dale Schuurmans and Greg Kondrak. I should also acknowledge my patient friends who were occasionally sucked into my research, especially Daniel Neilson for bug-hunting, Jess Enright for white-boarding, and Dan Lizotte for always listening. I have been extremely lucky to receive financial support from NSERC, iCORE, and Alberta Ingenuity.

I would like to thank Kevin Knight for his comments and suggestions as my external examiner. I am also grateful to Alexandra Birch for supplying the code for her joint phrasal translation model. This research would have been much more difficult without the efforts of the members of the GIZA++ and Moses projects to make state-of-the-art translation tools available to the research community. The human evaluation found in this thesis was made possible by my dutiful bilingual annotators, Dan Lizotte and Elaine Parker.

Finally, putting together a document like this would not have been possible, or even worthwhile, without the love and support of my wife, Becky.

Table of Contents

1 Introduction **1**
- 1.1 Task Descriptions . . . 2
  - 1.1.1 Translation . . . 2
  - 1.1.2 Word Alignment . . . 2
- 1.2 Motivation . . . 4
- 1.3 Approach . . . 5
  - 1.3.1 Syntactic Structures . . . 5
  - 1.3.2 Syntactic Constraints . . . 7
- 1.4 Outline . . . 8

2 Literature Review **9**
- 2.1 Basic Word Alignment . . . 10
  - 2.1.1 IBM Translation Models . . . 10
  - 2.1.2 Competitive Linking . . . 14
- 2.2 Inversion Transduction Grammar . . . 18
  - 2.2.1 Formalism and Grammars . . . 18
  - 2.2.2 Probability Model and Learning . . . 21
  - 2.2.3 Algorithms . . . 22
  - 2.2.4 Alignment Space . . . 29
  - 2.2.5 Discussion . . . 30
- 2.3 Other Syntactic Solutions . . . 31
  - 2.3.1 Lexicalized ITG . . . 31
  - 2.3.2 Generalized Multitext Parsing . . . 32
  - 2.3.3 Tree-to-string Methods . . . 34
  - 2.3.4 Tree-to-tree Methods . . . 37
- 2.4 Phrasal Bitext Analysis . . . 38
  - 2.4.1 Heuristic Combination . . . 39
  - 2.4.2 Matrix Factorization . . . 41
  - 2.4.3 Recursive Alignment . . . 42
  - 2.4.4 Information Theoretic Model Re-estimation . . . 43
  - 2.4.5 Joint Phrasal Translation Model . . . 45
  - 2.4.6 Phrasal Alignment Review . . . 46
- 2.5 Discriminative Alignment . . . 47
  - 2.5.1 Perceptron Alignment . . . 48
  - 2.5.2 Discriminative Matching . . . 49
  - 2.5.3 Other Discriminative Solutions . . . 51
  - 2.5.4 SVM for Structured Output . . . 52
- 2.6 Decoding Algorithms . . . 54
  - 2.6.1 Phrase-based Statistical Machine Translation . . . 54
  - 2.6.2 Syntactic Preprocessing . . . 58
  - 2.6.3 Dependency-based Statistical Machine Translation . . . 59
  - 2.6.4 Tree Transducers . . . 60

3 Evaluation Metrics **62**
- 3.1 Alignment . . . 62
- 3.2 Translation: BLEU . . . 63

4 Phrasal ITG **65**
4.1 Adding Phrases . . . 66
4.2 Comparison to Joint Phrasal Translation Model . . . 67
4.2.1 Improving Efficiency . . . 68
4.2.2 Handling the ITG Constraint . . . 71
4.3 Applying the Model . . . 71
4.3.1 Translation Modeling . . . 71
4.3.2 Phrasal Word Alignment . . . 72
4.4 Experiments . . . 73
4.4.1 Experimental Details . . . 74
4.4.2 Pruning Speed Experiments . . . 76
4.4.3 Alignment Experiments . . . 77
4.4.4 Translation Experiments . . . 78
4.5 Summary . . . 80

5 Syntactic Constraints for Word Alignment **81**
5.1 Syntactic Cohesion . . . 82
5.1.1 Qualitative Alignment Space Comparison . . . 83
5.1.2 Chart Modification . . . 84
5.1.3 Custom Grammar . . . 86
5.2 Head-aware Dependency Match . . . 89
5.2.1 Chart Modification . . . 90
5.2.2 Custom Grammar . . . 90
5.2.3 Alignment Space . . . 91
5.3 Alignment Space Experiments . . . 91
5.3.1 Experimental Details . . . 92
5.3.2 Guidance Test . . . 94
5.3.3 Expressiveness Test . . . 95
5.4 Discriminative ITG with Soft Cohesion Constraint . . . 97
5.4.1 SVM Training . . . 98
5.4.2 Features . . . 100
5.4.3 Gold Standard Design . . . 103
5.4.4 Re-implementation of Discriminative Matching . . . 104
5.5 Discriminative Experiments . . . 104
5.5.1 Experimental Details . . . 105
5.5.2 Alignment Evaluation . . . 105
5.6 Summary . . . 106

6 Cohesive Phrasal Decoding **107**
6.1 Cohesive Phrasal Output . . . 108
6.2 Cohesion Constraint for Phrasal Decoding . . . 111
6.2.1 Algorithm . . . 112
6.2.2 Soft Constraint . . . 114
6.2.3 Implementation . . . 115
6.3 Experiments . . . 115
6.3.1 Experimental Details . . . 115
6.3.2 K-best list filtering . . . 116
6.3.3 Constrained Decoding . . . 118
6.3.4 Error Analysis . . . 123
6.3.5 Related Results . . . 126
6.4 Summary . . . 127

7 Conclusion **128**

Bibliography **133**

A Glossary **141**

B Proof: Interruption $\Longleftrightarrow$ Cohesion Violation **143**

List of Tables

2.1 A co-occurrence contingency table for word pair (e, f) 16
2.2 A qualitative comparison of several phrasal word alignment systems. 47

4.1 Inside-outside run-time comparison of pruning methods. 77
4.2 Comparison of the phrasal ITG aligner to GIZA++ combination. 78
4.3 Translation comparison of various phrase tables. 79

5.1 Alignment space comparison with an unsupervised, learned link score. . . 94
5.2 Alignment space comparison with an oracle link score. 96
5.3 Performance of SVM aligners with various degrees of cohesion constraint. . 105

6.1 Spans of the local trees rooted at *begins* shown in Figures 6.2 (a, b) 110
6.2 BLEU score comparison of K-best list cohesion filter. 116
6.3 A comparison of baseline and cohesion-filtered English-French translations. 117
6.4 BLEU scores for English to French with integrated cohesion constraint. . . 118
6.5 BLEU scores for English to French with lexical re-ordering. 119
6.6 BLEU scores for English to German. 120
6.7 Counts of preferences from the human evaluation. 121
6.8 Confusion matrix from human evaluation. 122
6.9 Piece-wise BLEU evaluations for development and test sets. 122
6.10 Weights resulting from minimum error rate training. 123
6.11 Analysis of soft constraint. 124

List of Figures

1.1 A bipartite graph alignment visualization. 3
1.2 A matrix alignment visualization. 4
1.3 A constituency parse tree. 6
1.4 A dependency parse tree. 6
1.5 The projects described in this thesis, and the constraints they employ. 8

2.1 An example of an ITG parse tree. 20
2.2 Three structures that result in the same alignment. 20
2.3 Constructing a bitext constituent from smaller constituents. 24
2.4 Outside recurrence – four ways to produce X from a larger Z. 27
2.5 Cells considered when assessing (e_s^t, f_u^v) for pruning. 28
2.6 The two forbidden structures disallowed by ITG alignment. 29
2.7 The forbidden structures in Figure 2.6, drawn as alignment matrices. 30
2.8 A violation of phrasal cohesion in a dependency tree. 36
2.9 Examples of alignment combination. 40
2.10 Many-to-many alignment with cepts 41
2.11 A number of phrases that could be extracted from a fixed alignment. 55
2.12 Three possible block orientations. 57

4.1 Three ways in which a phrasal ITG can analyze a multi-word span. 66
4.2 Example of a phrasal ITG decomposition. 67
4.3 Three alignments illustrating the non-compositional constraint. 73

5.1 Maximally (a) and minimally (b) permissive dependency trees. 83
5.2 Example valid and invalid spans. 85
5.3 The invalid spans induced by a dependency tree. 85
5.4 A recursive ITG. 87
5.5 A counter-intuitive ITG structure. 89
5.6 Structures allowed by the head constraint. 90

6.1 Example English source tree with translation French output. 109
6.2 Two candidate translations for the same parsed source. 110
6.3 Narrowing down the source subtrees to be checked for completeness. 113
6.4 Three false violations. 125

B.1 A hypothesis extension $\bar{f}_1^h + \bar{f}_{h+1}$ that interrupts $T(r)$. 144

Chapter 1

Introduction

Machine translation, or MT, is the study of how computers can assist humans in the endeavor of translating natural language. The ultimate objective of this field is to create a system that can translate one human language into another, accurately and automatically. Traditionally, this task has been handled by applying a series of manually-designed rules to the source text, which transforms it into the target language. However, in the past two decades, the availability of large-scale examples of translation has allowed the creation of effective statistical methods, where a translator is learned from data. Statistical machine translation, or SMT, has several advantages over hand-engineered systems, including ease of extension to new vocabulary, and increased speed of deployment to a previously unconsidered language pair or domain. It also represents a significant shift in overall translation strategy. Data is used to construct a model that scores translations; so the actual translation process becomes a search for the output that receives the best score according to the model. Similarly, model construction becomes a search for a model that optimizes some quality measure over the data. In effect, translation and its subproblems have been recast as optimization problems.

The translation models required for SMT could not be built without large-scale collections of translation examples. We refer to a collection of documents available in multiple languages as a **parallel text** or **parallel corpus**. Most frequently, the number of languages is two; in which case, we call the collection a **bitext**. These texts are generally produced by human translators for some unrelated purpose. The most plentiful and common examples of parallel corpora are the proceedings of multilingual political entities, such as Canada, Hong Kong, and the European Union. Another common source is the technical manuals from multinational corporations, such as Microsoft and IBM. There is a wealth of translation information available in these parallel corpora, but it is not clear how to best put it to use. Af-

ter all, we have examples of entire documents being translated, not instructions for general translation. Fortunately, the sentence alignment task, which determines which sentences correspond to one another in a bitext, is considered by most to be a solved problem (Gale and Church, 1991b). So, we effectively have access to examples of sentence-by-sentence translation. The remaining challenges are to extract a model of translation from these **sentence pairs**, and to then use that model to actually translate text. This thesis will contribute toward the solution of both problems; therefore, we describe them in further detail before we motivate and introduce our contributions.

1.1 Task Descriptions

1.1.1 Translation

Since it is a familiar task, often carried out by humans, the goal of translation is easy to state, but very difficult to solve. Simply put, the task is to translate an input sentence from one human language into another. We call the input language the **source**, and the output language the **target**. Assume that we have constructed a translation model from the data in a parallel corpus; given a source sentence and this translation model, the process of finding the best target translation according to the model is referred to as **decoding**. The term is borrowed from SMT's roots in information theory.

Originally, model design and training were somewhat coupled to the decoding task. The statistical translation models used by the decoder were inferred directly from a sentence-aligned bitext. More recent approaches train their models on a word-alignment of the bitext, to reduce the complexity of model construction. This alignment step provides an additional layer of bitext modeling, one that is not necessarily aware of the decoder. The task of creating these word alignments automatically is also explored in this thesis.

1.1.2 Word Alignment

The goal of word alignment is to find word-to-word correspondences in bilingual sentence pairs. Two words correspond if they are translations of one another, or if they play the same role in their respective sentences. Since this task is considerably less familiar than translation, it is described here in considerably greater detail.

The notions of source and target are not always appropriate in alignment, where both languages are provided as input. Therefore, we will often refer to the two languages to be aligned as **English** and **French**. The techniques described, however, should be understood

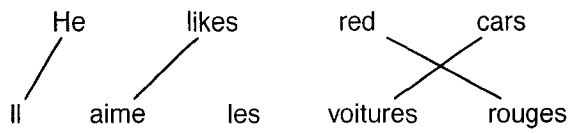

Figure 1.1: A bipartite graph alignment visualization.

to be language independent unless indicated otherwise. We refer to a specific occurrence of a word in text as a **token**, while we refer to the concept of that word as a **word** or **type**. As we align a sentence pair, we call the entire structure explaining its bilingual connections an **alignment**, while we call the individual word-to-word connections **links**. Formally:

Given an English-French sentence pair $(\vec{e}, \vec{f})$, with respective sentence lengths m and n, an alignment is a set A of links (e_i, f_j) between tokens e_i and f_j, where $1 \leq i \leq m$ and $1 \leq j \leq n$, such that $(e_i, f_j) \in A$ indicates that e_i and f_j correspond to one another.

This is the broadest possible definition of word alignment. Certain techniques use alternate notations that suit their limitations or models. For example, an alignment can also be viewed as an m-by-n binary matrix A, where $A_{i,j} = 1$ indicates a link (e_i, f_j), with no loss in expressiveness. If we only want to explain every French token with its single most appropriate English token, then the alignment can be represented as an n-dimensional vector $\vec{a}$, where $a_j = i$ indicates a link (e_i, f_j).

Visualization

A word alignment is easiest to interpret when visualized. We present examples of the two most common visualizations here: bipartite graphs and alignment matrices. Suppose we have the English-French sentence pair, ("He likes red cars", "Il aime les voitures rouges"). The correct alignment for this pair would be:

$$A = \{(\text{He}_1, \text{Il}_1), (\text{likes}_2, \text{aime}_2), (\text{red}_3, \text{rouges}_4), (\text{cars}_4, \text{voitures}_3)\}$$

Figure 1.1 illustrates this alignment as a bipartite graph, where nodes are tokens, and an edge indicates a pair of linked tokens. Figure 1.2 shows the same alignment in matrix or grid format. In this visualization, a black dot denotes a link between the tokens corresponding to its row and column labels.

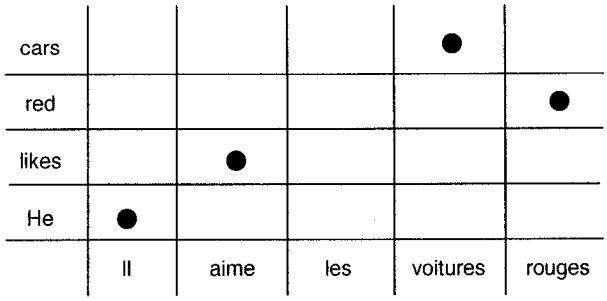

Figure 1.2: A matrix alignment visualization.

1.2 Motivation

The framework of statistical machine translation has recast translation as a series of optimization problems, which include word alignment and decoding. In both of these problems, there is a model that scores outputs, as well as a set, or **space**, of possible outputs that must be considered in the search for the highest score. Machine translation is a challenging problem because it requires structured outputs that are naturally complex and flexible, making these output spaces very large. The size of these spaces can lead to two types of problems:

1. Large output spaces place a heavy burden on the model; it must be sufficiently complex to differentiate between the many possible outputs.
2. In order to efficiently enumerate a large structured output space, one may need to place assumptions on the model that weaken its modeling power.

Note that these two problems place conflicting pressures on the system designer: the model must be complex enough to identify the correct answer, but simple enough to actually find it. In some cases, the best solution is to shrink the output space by making assumptions that enable efficient enumeration while simultaneously eliminating undesirable outputs.

Take, for example, the alignment task. Assume we are given a sentence pair where both sentences have n tokens. The unconstrained **alignment space** has 2^{n^2} possible alignments, because each of the n^2 potential links can be either on or off. In this space, links do not naturally interact; a weak link is never eliminated because a strong link is present, we can always include both. This means that we require a complex model that accounts for interactions between linking decisions, and because our model ties all such decisions together, we will likely need to perform alignment using an approximate beam search with clever heuristics. However, as Melamed (2000) observed, if we enforce a **one-to-one** constraint on alignment space, allowing each token to participate in at most one link, then links

naturally rule each other out, reducing the need to model their interactions; furthermore, we can search this reduced alignment space efficiently using the bipartite matching algorithm (West, 2001). Unfortunately, such a constraint also limits expressiveness, preventing us from recovering phrase-to-phrase relationships in our alignments. However, the solution remains a strong example of the benefits of shrinking an output space.

The one-to-one constraint reduces the complexity of the alignment problem by simplifying one of the complexities of translation: it assumes translation happens word-by-word, when in reality, this is not always the case. Another major source of complexity in translation results from differences in word order between languages. From our simple example of "red cars" becoming "voitures rouges" in French, to long-distance movement of verbs when translating from English to German or Japanese, it can appear as if words are free to undergo any permutation during translation. If one allows unconstrained concept movement, then the movement component alone is sufficient to show translation with a language model to be NP-complete (Knight, 1999). On the other hand, if we assume concepts always remain in the same order, translation becomes much easier (Zens and Ney, 2004), but the resulting models are unable to express many common phenomena. Fortunately, an interesting middle-ground does exist. An appeal to the syntactic structure of language can limit concept movement while still allowing many reasonable permutations. If the assumptions behind doing so are sound, one can hope to increase both efficiency and performance at the cost of some, hopefully unnecessary, expressiveness.

1.3 Approach

This thesis investigates the benefits of syntactic constraints on translation. In particular, we examine the effect of assuming that language has a context-free structure, and that this structure is sufficient to describe the concept movement that occurs during translation. To understand what this means, it is important to first specify our structures.

1.3.1 Syntactic Structures

This thesis deals with two main types of tree structures that can be used to represent the syntactic relationships that exist within a sentence: constituency trees and dependency trees. We briefly describe these formalisms and their visualizations below.

Constituency trees reflect the derivation of a sentence according to a context-free grammar, or CFG (Hopcroft and Ullman, 1979). Leaf nodes of the tree correspond to words in the sentence, which are also terminal symbols in the grammar. Internal nodes correspond

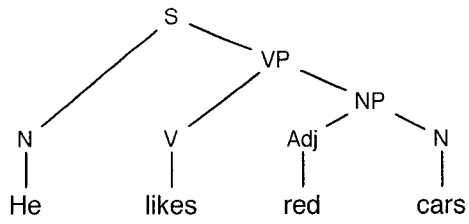

Figure 1.3: A constituency parse tree.

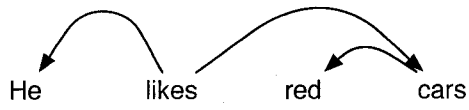

Figure 1.4: A dependency parse tree.

to non-terminal symbols. In any tree, we refer to a tree fragment consisting of only a parent and its immediate children as a **local tree**. Each CFG production in the derivation is represented by a local tree. A simple constituency tree for the English half of our running example is shown in Figure 1.3.

Dependency trees reflect the connections between words more directly. The tree has one node for each word in the sentence. A directed link that connects $e_i \rightarrow e_j$ indicates that e_j modifies the meaning of e_i. A dependency tree for our example sentence is shown in Figure 1.4. In this case "red" modifies "cars", in that it provides a color description for the noun "cars", while "cars" itself provides an object modifier for the verb "likes". A dependency tree can be created from a constituency tree by defining heads in CFG productions, and then pulling those heads up the tree structure, replacing internal nodes. In Figure 1.3, "cars" is the head of the NP, while "likes" is the head of both the VP and the S.

We refer to any tree structure built over language as a **parse tree**, because it can be found using a natural language parser. Regardless of the formalism, we assume that our parse trees are **projective**. This means that if e_k is a descendant of e_i, then $\forall j$ between i and k, e_j is also a descendant of e_i. Visually, this means that none of the arcs in the tree ever cross each other. Projectivity is a side-effect of assuming a context-free formalism. Because of projectivity, we can guarantee that any observable tokens found under a subtree of a parse will form a contiguous block of text, which corresponds to a **syntactic phrase**. For example, in Figure 1.3, we have the syntactic phrases "red cars" rooted at NP, and "likes red cars" rooted at VP, while the dependency tree in Figure 1.4 contains the phrase "red cars" rooted at "cars". Since phrase is an overloaded term in SMT, we often refer to

these syntactic phrases as subtrees.

1.3.2 Syntactic Constraints

As we discuss in our literature review (§ 2), syntax can be used in a number of ways to improve bitext analysis or translation, and we are by no means the first to explore the advantages of doing so. What we present here, though, is a series of projects that limit how words can be permuted according to how they are syntactically related. Specifically, we benefit from the assumption that *syntactic phrases move together during translation.* If we assume that our syntactic phrases, like "red cars", are meaningful units of thought, then it makes sense that we would be able to find the same units from the source in the translation. We only ever consider the tree structure of the sentence, and never the part-of-speech tags, non-terminal labels, or dependency labels that may exist in a parse tree. In this manner, our methods take minimal advantage of syntax, isolating the movement constraints that are usually implicit in other syntactic approaches. This places us in the unique position of asking to what degree these constraints can help or harm a system, if employed on their own. Whenever possible, we also investigate soft constraints, to benefit from these general rules as much as possible while still being able to handle exceptions.

The assumption that syntactic phrases move together has two major benefits. First and foremost, it reduces the space of possible outputs for both translation and word alignment, by reducing the amount of concept movement that can occur during translation. This can improve efficiency, but we will generally be more interested in the potential improvements to accuracy, which can occur when we eliminate candidates that were confusing our statistical models. Also, with the correct algorithms and assumptions, breaking a sentence or sentence pair into a hierarchy of subtrees can improve efficiency by enabling dynamic programming. We investigate this second benefit in particular in Chapter 4.

We explore two major types of constraints, which are summarized here and described in greater detail in the literature review. The first, called an ITG[1] constraint (§ 2.2), assumes that there is some binary constituency structure that explains any observed movement, but the exact structure is not specified in advance. The second, called a cohesion constraint (§ 2.3.3), uses a monolingual parser to determine a fixed dependency structure before processing begins. Each of our projects works with one or both of these constraints, as shown in Figure 1.5. We explore these constraints in the context of three SMT subtasks, which are also shown in Figure 1.5, with arrows indicating where the task appears

[1]The acronym ITG stands for Inversion Transduction Grammar, which is described in depth in § 2.2.

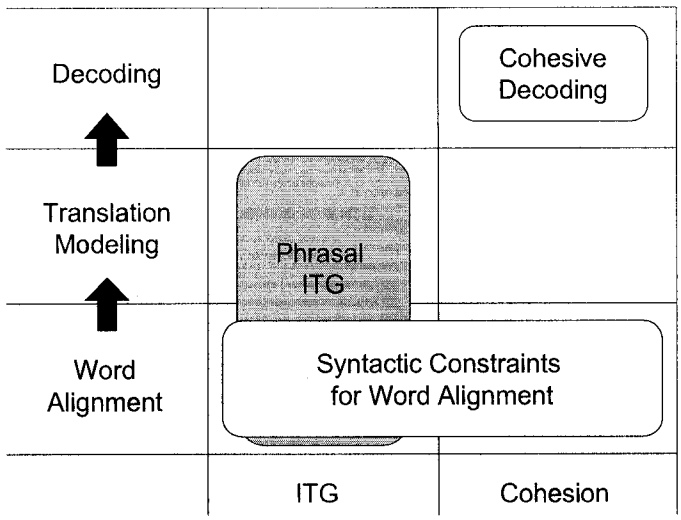

Figure 1.5: The projects described in this thesis, and the constraints they employ.

in the SMT pipeline: word alignment enables translation modeling, which in turn guides decoding.

1.4 Outline

This thesis describes three novel projects, which are united by a theme of syntax-limited concept movement. We begin in Chapter 2, with a comprehensive review of the literature, covering previous word alignment and decoding methods, focusing on syntactic and phrasal approaches. We then briefly describe the evaluation metrics for both translation and word alignment in Chapter 3. Next, we present each project in its own chapter, complete with a description of its methods and experiments. Chapter 4 describes our phrasal ITG, an ITG-constrained system that uses bitext parsing to create an efficient and effective joint phrasal translation model. It is evaluated using both the translation and word alignment tasks. Chapter 5 compares and combines ITG and cohesion constraints in the context of one-to-one word alignment, isolating the effects of these movement constraints. It also presents a method to discriminatively train a bitext parser, which we use to create a soft cohesion constraint on word alignment. Chapter 6 presents cohesive phrasal decoding, where we provide a definition of cohesion that is valid even when a decoder employs non-syntactic phrases. We then modify a popular left-to-right phrasal decoder to eliminate or avoid cohesion violations, according to a source-side dependency tree. Chapter 7 summarizes and discusses our contributions, and presents some ideas for future work.

Chapter 2

Literature Review

Statistical machine translation is a young and rapidly growing field. Each year, the number of papers published on this topic increases, creating a large number of competitors for any proposed SMT technique. This review attempts to summarize previous work that is most relevant to this thesis. The task of conducting a concise literature review is further complicated by the fact that this thesis is centered around an idea, rather than a single solution or technique. We wish to take advantage of syntactic limits on concept movement. In doing so, we touch on a number of modeling, learning, and search techniques.

Two of the three projects presented here are concerned with word alignment and bitext analysis; they aid in the extraction of translation models from bitext. As such, our review will focus mostly on word alignment and related techniques. However, since the beginning of SMT, the line between model construction and translation has been a blurry one; many of the translation models we discuss in a word-alignment context are also used in translation. Furthermore, both tasks are concerned with how words are translated, and how they can move during translation, so lessons from one can easily influence the other.

We begin with a discussion of basic, flat-string word alignment. This introduces several concepts, such as expectation maximization and statistical correlation measures, which are used heavily in our own methods. In § 2.2 and 2.3 we move on to syntactic approaches to bitext analysis, examining first Inversion Transduction Grammar, which is the foundation for two of our main contributions, and then covering other syntactic methods. This is followed in § 2.4 and 2.5 by discussions of phrasal and then discriminative bitext analysis, paving the way for our contributions in each of these areas. We end with a relatively brief review of complete translation systems and decoders, focusing on those most relevant to our own cohesive decoder.

2.1 Basic Word Alignment

This section will review two basic approaches for word alignment: the IBM models and Melamed's competitive linking algorithm. Together, these methods lay the foundation for most other word alignment methods. They are united in that they track only word-based statistics, and they treat sentences as flat strings. These reviews are intentionally brief, as both topics are reviewed in detail in (Cherry, 2003).

2.1.1 IBM Translation Models

In (Brown et al., 1993) a team of IBM researchers proposed a series of translation models that could be learned from sentence-aligned bitext. These models were intended for use in a decoder that would search for an English translation $\vec{e}$ of a French source sentence $\vec{f}$, so that $\vec{e}$ maximizes $\Pr(\vec{e}|\vec{f})$, where:

$$\Pr(\vec{e}|\vec{f}) \propto \Pr(\vec{f}|\vec{e})\Pr(\vec{e}) \tag{2.1}$$

Here, $\Pr(\vec{f}|\vec{e})$ is one the IBM translation models, and $\Pr(\vec{e})$ is a monolingual language model (see Appendix A) designed to identify good English. This decomposition is often referred to as the fundamental equation of statistical machine translation. Recently, the word-based IBM translation models have been abandoned for decoding purposes, in favor of phrasal models (§ 2.6). However, due to their practicality, effectiveness, and the availability of the free implementation, GIZA++ (Och and Ney, 2003), they remain the standard method for creating word alignments in bitext.

The IBM models are a classic example of a purely statistical alignment technique; everything needed to solve the problem, such as the alignment space and alignment features, is encapsulated in the translation model. This approach would go on to inspire competing techniques such as (Yamada and Knight, 2001) reviewed in § 2.3.3, and (Marcu and Wong, 2002) reviewed in § 2.4.5. In this section, we review the three IBM models that still see heavy use: IBM-1, HMM, and IBM-4.

IBM Model 1

Introduced in (Brown et al., 1993), IBM Model 1, or IBM-1, is the simplest and most elegant of the IBM translation models. It is defined in terms of a generative story of how a French sentence $\vec{f}$ is created from an English sentence $\vec{e}$ with length m. In this story, first a number of French words n is selected from a uniform distribution. Then, each French position j selects an English generating position i, again according to a uniform distribution.

The selected generating positions define a word alignment $\vec{a}$, such that $a_j = i$. Finally, each French position j selects a French word f_j according to a distribution defined by its English generator: $\Pr(f_j|e_{a_j})$. A French sentence's probability is found by summing over all possible alignments that would generate the sentence:

$$\Pr(\vec{f}|\vec{e}) = \sum_{\vec{a}} \Pr(\vec{f}, \vec{a}|\vec{e}) \tag{2.2}$$

where:

$$\Pr(\vec{f}, \vec{a}|\vec{e}) = \frac{\epsilon}{(m+1)^n} \prod_{j=1}^{n} \Pr(f_j|e_{a_j}) \tag{2.3}$$

and where ϵ is the probability of the French length n. An artificial null word e_0 is added to each sentence pair, to generate any French token that has no clear connection to any other word in $\vec{e}$, hence each of the n French tokens has $m+1$ generators to choose from.

Under these generative assumptions, there are $(m+1)^n$ possible alignments for any $(\vec{e}, \vec{f})$ pair, making the sum over all possible alignments quite intimidating. Fortunately, since the choice of a_j has no effect on the probability of any $a_{j'} : j' \neq j$, Brown et al. (1993) show us that (2.2, 2.3) can be factored and re-written as the far more manageable:

$$\Pr(\vec{f}|\vec{e}) = \frac{\epsilon}{(m+1)^n} \prod_{j=1}^{n} \sum_{i=0}^{m} \Pr(f_j|e_i) \tag{2.4}$$

A similar transformation allows efficient link counting. That is, if a word pair (e, f) is linked $link(e, f; \vec{e}, \vec{f}, \vec{a})$ times by alignment $\vec{a}$, then $\sum_{\vec{a}} \left[\Pr(\vec{a}|\vec{e}, \vec{f}) \cdot link(e, f; \vec{e}, \vec{f}, \vec{a})\right]$ can be computed in polynomial time. These counts allow us to conduct the E-step required to train IBM-1 using Expectation Maximization or EM (Dempster et al., 1977).

The EM algorithm begins by collecting link counts over all sentence pairs $(\vec{e}, \vec{f})$ in the training corpus (the E-step), according to a uniform alignment distribution $\Pr(\vec{a}|\vec{e}, \vec{f})$. Then the $\Pr(f|e)$ tables are re-estimated according to how often e was seen generating each possible French word f (the M-step). These new tables are used to re-estimate $\Pr(\vec{a}|\vec{e}, \vec{f})$, and links are counted again. This training process iterates until a local maximum is reached, or until some early stopping criteria is met. This sort of iterative process lies at the heart of most unsupervised word alignment algorithms. EM is guaranteed to converge to a local maximum in the probability of the training data: $\prod_{(\vec{e},\vec{f})} \Pr(\vec{f}|\vec{e})$. Furthermore, IBM-1 is provably convex (Brown et al., 1993), which means that any local maximum found is also a global maximum. For this reason, IBM-1 parameters are often used to boot-strap more complicated, non-convex models.

Above, we discuss training a translation model; however, we are often interested in word alignment. Fortunately, aligning a sentence pair $(\vec{e}, \vec{f})$ with IBM-1 is simply a matter of finding $\text{argmax}_{\vec{a}} \Pr(\vec{a}|\vec{e}, \vec{f})$. Due to the structure of IBM-1, this can be done easily by having each French token f_j select the English token e_i that maximizes $\Pr(f_j|e_i)$, breaking ties arbitrarily. Recently, it has been shown that a slightly altered IBM-1 training process can be initialized heuristically to achieve more accurate alignments when training with early stopping (Moore, 2004a).

Alignment Hidden Markov Model

One weakness of IBM-1 is that its decisions are informed only by lexical information. No attempt is made to capture notions related to token position such as:

1. In language pairs with similar word order, token pairs with similar positions in their respective sentences are more likely to be linked; or,
2. Words tend to move together during translation, that is the translation of e_i will tend to appear near the translation of e_{i-1}.

One can incorporate either of these two ideas by replacing the uniform position distribution from IBM-1 with a conditional distribution. IBM-2 (Brown et al., 1993) captures the first notion, while the alignment Hidden Markov Model or HMM, introduced in (Vogel et al., 1996), captures the second. In the alignment HMM, the selection of the French position j's generator a_j is conditioned on the selected generator for the previous French token a_{j-1}. The probability of a particular translation and its alignment becomes:

$$\Pr(\vec{f}, \vec{a}|\vec{e}) = \epsilon \prod_{j=1}^{n} \left[\Pr(f_j|e_{a_j}) \Pr(a_j|a_{j-1}, m)\right] \tag{2.5}$$

As usual, translation probability can be computed by summing over all possible alignments as in (2.2). This sets up a problem similar to HMM part-of-speech tagging (see Appendix A), where instead of tagging tokens with their corresponding parts-of-speech, we are tagging French tokens with their generating English tokens.

The motivating intuition for the alignment HMM is that words tend to move together during translation. A table of probabilities for position pairs $\Pr(a_j|a_{j-1}, m)$ would fail to generalize as intended. Separate parameters would exist for $(14|13, 20)$ and $(4|3, 20)$, when intuitively, both situations are very similar: an English token's translation was placed directly after the previous English token's translation. Instead, parameters $s(a_j - a_{j-1})$ are

maintained for each "jump" in position, and the transition probability becomes:

$$\Pr(a_j|a_{j-1}, m) = \frac{s(a_j - a_{j-1})}{\sum_{i=1}^{m} s(i - a_{j-1})}$$

By conditioning on a_{j-1}, we no longer have the independence between links that allowed us to collect expectations and align quickly in IBM-1. Fortunately, a_j is only dependent on a_{j-1}, and this Markov-1 dependence allows the use of HMM algorithms to efficiently enumerate the space. The forward-backward algorithm (Manning and Schütze, 2001) can be used to implicitly sum over all possible alignments, and perform the E-step in EM. Meanwhile, the Viterbi algorithm (Manning and Schütze, 2001) can be used to find the single most likely alignment.

Och and Ney (Och and Ney, 2000a) present several extensions to the alignment HMM. They add the null word e_0 to the model, introduce forward-backward training, and add further conditioning on automatically-derived word clusters. They also implemented the HMM inside GIZA++ (Och and Ney, 2003), where it has come to replace IBM models 2 and 3, which will not be discussed in detail here. Many see the HMM as the simplest IBM model after IBM-1, so it has been used to experiment with novel alignment features and alternate models, such as in (Lopez and Resnik, 2005; Deng and Byrne, 2005).

IBM Model 4

IBM-1 and HMM alignment are one-to-many alignment methods: each English token can participate in many links, while each French token can participate in at most one link. However, they are not one-to-many models; during any given link decision, the model has no notion of how often a particular English token has been used as a generator, or which other French tokens it has generated. For Models 3 and above, Brown et al. (1993) tell a new generative story, focused on the English generators rather than the French tokens, so that these factors are taken into account. In this story, the French is generated from English by walking through the English sentence from left to right, with each English token generating some number of French tokens and placing them in the French sentence. Of the models that use this story, IBM-4 still sees heavy use, as part of the GIZA++ alignment package. We summarize the features and structure of IBM-4 here, but for the sake of brevity we avoid specifics and equations, as the notation gets quite complicated. For more details, the reader is referred to (Brown et al., 1993; Cherry, 2003).

IBM-4 adds two new features to IBM-1's $\Pr(\vec{f}, \vec{a}|\vec{e})$ equation shown in (2.3). The first is a notion of fertility, which also appears in IBM-3. Fertility is a probability distribution

that indicates how many French tokens an English type e tends to generate. This allows the model to understand that the English "the," which translates to "le", "la" or "les" usually generates at most one token, while "over," which translates to "au - dessus," generally generates three tokens. The second feature is a Markov-1 position selector. Suppose e_i generates three French tokens $f_{j_1 \ldots j_3}$, where the subscript indicates their left-to-right order in the French sentence, so f_{j_1} appears further to the left than the others. The position of f_{j_1} is drawn from a probability distribution according to the location of the average positions of the French tokens generated by e_{i-1}. This is similar to the HMM parameterization, but focused on the English rather than the French. The positions of the remaining tokens f_{j_2} and f_{j_3} are determined according to the previous token generated by e_i; that is, f_{j_2} depends on f_{j_1} and f_{j_3} depends on f_{j_2}. Brown et al. (1993) also condition on word cluster membership when selecting positions.

There are two main problems with IBM-4. First, it is well-known to be deficient; that is, probabilities are assigned to alignments that do not correspond to real sentences. For example, some probability will be reserved for cases where every French token is generated on the same position. This is fixed in (Brown et al., 1993) with IBM-5, which over-rides these cases with a zero probability. Och and Ney (2000b) show that a simpler solution to the problem is to make the model equally deficient in all its component distributions, allowing IBM-4 to out-perform IBM-5. Second, with the inclusion of the fertility parameter, links have been tied in such a way that alignments can no longer be enumerated efficiently. To solve the alignment problem, an imperfect hill climbing search is used to find the approximately most likely alignment. The E-step of EM sums over only a set of high-probability alignments found using hill-climbing and sampling, instead of all possible alignments. However, the modeling power of IBM-4 overcomes these defects, allowing it to improve upon HMM alignment (Och and Ney, 2003).

2.1.2 Competitive Linking

The IBM models are completely statistical translation models. Every aspect of the problem, from the training algorithm to the alignment space, is provided by applying standard methods to a well-defined generative model. This is both a strength and a weakness. Modifying these generative models can be quite intimidating. Furthermore, the purely generative framework complicates the inclusion of some features that can help alignment. Melamed's models of translational equivalence (2000), typified by the competitive linking algorithm, provide an effective alternative to the IBM framework.

In competitive linking, we no longer rely on a generative story where one sentence translates into another; we are just trying to find good alignments. As a result of this, tokens no longer have generators, and the alignment vector $\vec{a}$ is replaced by a set A of links (e, f). The basic structure of this approach is as follows:

1. Initialize a $score(e, f)$ on bilingual word pairs, based on their correlations in a bitext.
2. Find the highest scoring alignment A for each sentence pair in the corpus according to $\sum_{(e,f)\in A} score(e, f)$ and some set of constraints on the links allowed in A.
3. Create new $score(e, f)$ from link counts. Repeat from step 2.

The algorithm retains the iterative structure of EM, without the statistical baggage. Steps 1 and 3 correspond to M-steps, while step 2 is the E-step. Note that the notion of counting all possible alignments, or even sampling from a weighted set of alignments, has been dropped in order to allow scores with no clear probabilistic interpretation. We examine the alignment search first, and then the various correlation and link-based *score* functions.

Alignment search

Our alignment search now requires an explicit set of constraints. In the IBM models, there is implicit competition, English tokens compete to be selected as generators, and in models with fertility, French tokens compete to be generated. With alignment represented as a set, there is no implicit competition; in order to avoid blindly selecting all links with positive *scores*, we need to impose external constraints. Put another way, we need to explicitly define the alignment space we are searching (§ 1.2). This can be seen as an advantage, allowing us to vary the alignment space without necessarily changing the *score* function, as is done in (Moore, 2005a; Cherry and Lin, 2003).

Traditionally, a one-to-one constraint (Melamed, 2000) is applied to the alignment search: in both sentences, each token can participate in at most one link. This is often sufficient to express most alignments of close language pairs, such as English-French, though as we will argue later, it is generally too restrictive. However, it does have some nice properties. Both languages are treated symmetrically, allowing some interesting modeling options that we discuss later in this section. Furthermore, under this constraint, the alignment search reduces to weighted bipartite matching (West, 2001), which allows optimal alignments to be found quickly. A greedy approximation is often used for historical reasons.

	f		$\neg f$	
e	$cooc(e, f)$	(a)	$cooc(e, \neg f)$	(b)
$\neg e$	$cooc(\neg e, f)$	(c)	$cooc(\neg e, \neg f)$	(d)

Table 2.1: A co-occurrence contingency table for word pair (e, f)

Correlation-based *score* functions

The first step in the iterative learning framework involves inducing a $score(e, f)$ without link counts. The most common source of information in this case is sentence co-occurrence: how often two types occur in the same sentence pair. Melamed (2000) argues eloquently that IBM-1 induces its initial model only accounting for how often the two words co-occur, which corresponds to (a) in the contingency table shown in Table 2.1. He then argues in favor of correlation metrics that account for all four entries in Table 2.1, where $\neg x$ indicates that any type other than x occurred. Two such metrics are ϕ^2 (Gale and Church, 1991a) and Dunning's log likelihood ratio (1993).

Gale and Church (1991a) introduced the ϕ^2 metric for use in a word-alignment task. The actual value used is the unnormalized χ^2 statistic, usually used for correlation tests:

$$\phi^2(e, f) = \frac{(ad - bc)^2}{(a + b)(a + c)(b + d)(c + d)} \tag{2.6}$$

However, the ϕ^2 statistic has at its core some normality assumptions that may not be well-suited for working with text, where many words are rare. Dunning (1993) argues for an alternative log likelihood ratio built around a binomial assumption. Later, Moore (2004b) re-formulated the ratio, showing it can be viewed as weighted mutual information between the two words:

$$G(e, f) = 2N_c \left[\sum_{e? \in \{e, \neg e\}} \sum_{f? \in \{f, \neg f\}} \Pr(e?, f?) \log \frac{\Pr(e?, f?)}{\Pr(e?)\Pr(f?)} \right] \tag{2.7}$$

where N_c is the total number of observed co-occurrences, and $\Pr(x, y) = cooc(x, y)/N_c$, and $\Pr(x) = \left(cooc(x, y) + cooc(x, \neg y)\right)/N_c$.

Link-based *score* functions

A weakness of correlation-based *scores* is that they do not take into account the fact that some pairs (e, f), though highly correlated, are explained away by other pairs that take precedence. Scoring functions built on link counts do not suffer from this drawback. Melamed (2000) proposes two such functions.

The first is a simple, joint probability model. In this case, rather than generating one sentence from the other, both sentences are generated in tandem. The generative process first creates a bag of bilingual concepts (cepts), and then generates English and French words from those concepts according to a joint distribution. Word order is then drawn from a uniform distribution. A link in an observed alignment indicates one concept. Under this probability model, a word-pair's $score(e, f)$ becomes $\log \Pr(e, f)$, where:

$$\Pr(e, f) = \frac{link(e, f)}{\sum_{(e', f')} link(e', f')} \tag{2.8}$$

and $link(e, f)$ counts how many times a type pair is linked throughout an aligned corpus.

The joint probability model provides a principled method to rank word pairs according to their link counts. However, it only accounts for the absolute number of times a pair was linked. This biases the score toward frequently occurring pairs, failing to account for how many opportunities a pair was given to link. To counter this, Melamed (2000) formulates an explicit noise model. This calculates a binomial likelihood ratio, comparing the probability of a type pair (e, f) being a true translation pair versus it being a false one, given $link(e, f)$ and $cooc(e, f)$. Cherry and Lin (2003) later showed that the same factors could be taken into account by simply calculating:

$$\Pr(link|e, f) = \frac{link(e, f)}{cooc(e, f)} \tag{2.9}$$

This term generally needs to be smoothed with either additive smoothing (Cherry and Lin, 2003) or absolute discounting (Moore, 2005a). Our internal experiments suggest that this conditional link probability is as effective as the explicit noise model. Also, being a probability distribution, it has the nice property of being bounded between 0 and 1, where the explicit noise model can span the real numbers. Furthermore, it allows the inclusion of a naïve Bayes feature model (Cherry and Lin, 2003).

Extensions

The strength of the competitive linking framework lies in its simplicity, and its ease of extension. Several others have come behind Melamed to modify his framework. Cherry and Lin (2003) developed a conditional probability model based on (2.9) for use in a modified competitive linking algorithm. They replaced Melamed's greedy search with a beam search, and added a syntactic cohesion constraint (§ 2.3.3) to the one-to-one constraint. Since their $score(e, f)$ had a probabilistic interpretation, they were able to re-introduce sampling to the link counting step of the iterative framework. Moore (2005a) also made use of the competitive linking framework with conditional link probability and a beam search. He added a

monotonicity preference to the search, and extended it to allow one-to-many links. Some discriminative approaches to word alignment such as (Taskar et al., 2005; Moore, 2005b) can be seen as extensions to competitive linking, where $score(e, f)$ is a linear combination of features, whose weights are learned with a supervised, discriminative method.

2.2 Inversion Transduction Grammar

The alignment methods discussed in § 2.1 are often unsatisfying to the linguistically minded. Though sentences are written as flat strings, they are recognized to have syntactic structure, which can be discovered using a natural language parser, and represented using a dependency or constituency tree (§ 1.3.1). The IBM models and the Competitive Linking framework ignore this structure, treating sentences as sequences or bags of words. In this section and the one following, we review methods that go beyond flat-string representations, informing their translation models and search algorithms with syntactic notions.

There are three primary reasons to add syntax to word alignment. First, one can incorporate syntactic features, such as grammar productions, into the models that guide the alignment search. Second, movement can be modeled more naturally; when a three-word noun phrase moves during translation, it can be modeled as one movement operation instead of three. Finally, one can restrict the type of movement that is considered, shrinking the alignment space in a principled manner. Generally, only methods that rely on outside resources, such as monolingual parsers, can take advantage of grammatical features, as those sorts of features rely on linguistically motivated symbols. The two movement-related advantages can be leveraged in models that use syntax without necessarily using an informative grammar, as we will see if our first example of a syntactic tool for bitext analysis, the Inversion Transduction Grammar, or ITG (Wu, 1997). As it is the base technology for several of the techniques described in this thesis, we examine ITG in some detail in this section. We cover other syntactic methods in § 2.3.

2.2.1 Formalism and Grammars

One way to incorporate syntactic notions into bitext analysis is to assume that the bitext was produced by a transduction grammar (Wu, 1997). A transduction grammar writes its terminal symbols to two streams instead of one. We first consider a context-free grammar extended for transduction, where the two streams correspond to our two languages. For example, an English-French transduction grammar may have the following terminal pro-

duction to express the concept of the color blue in both streams:

$$\text{Adj} \rightarrow \textit{blue}/\textit{bleu}$$

Synchronously produced terminals are understood to be translations of one another. Otherwise identical to a normal CFG, a transduction grammar builds one tree, with two sets of labels for the leaf nodes.

Transduction grammars are unable to account for the sorts of movement that can occur during translation. For example, adjectives often appear before the noun in English, but after the noun in French. Because they share non-terminal productions between streams, one cannot write a transduction grammar that is correct for both cases. To handle this situation, Wu (1997) introduces the concept of an inverted production. An inverted production has its right-hand-side written from left to right in one stream, and from right to left in the other. This adds an extra layer of notation to the grammar, where inverted sequences are enclosed in angled brackets $\langle \ldots \rangle$, while straight sequences use square brackets $[\ldots]$. Our adjective situation can now be handled using the following production:

$$\text{NP} \rightarrow \langle \text{Adj NP} \rangle$$

The inversion transduction grammar or ITG formalism (Wu, 1997) provides a context-free formalism that uses these two concepts to model translation.

Binary bracketing grammar

Theoretically, one could design a linguistically motivated ITG for their language pair, with meaningful non-terminals such as the NP and Adj alluded to above. However, it is not clear how to construct such a grammar by hand without accounting for many special cases. In practice, a simple binary bracketing grammar is usually employed:

$$\text{A} \rightarrow [\text{AA}] \mid \langle \text{AA} \rangle \mid e/f \tag{2.10}$$

In this grammar, there is only one non-terminal A, which has no linguistic interpretation. It can produce straight or inverted non-terminal pairs, or a terminal in each stream. The synchronously produced terminals e and f are place-holders for words drawn from English and French respectively. The terminal alphabet for each stream also contains a special null token $\emptyset$ to allow for tokens without translation.

Given an ITG derivation, one can determine the corresponding word alignment according to which tokens were synchronously produced by terminal productions. An ITG parse

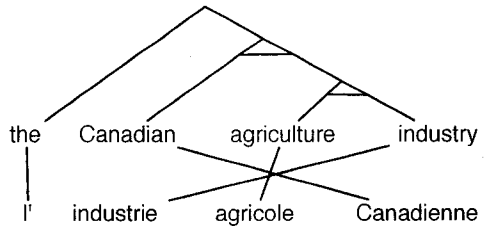

Figure 2.1: An example of an ITG parse tree.

tree of an English-French sentence fragment is shown in Figure 2.1, along with its corresponding word alignment. Inverted productions are indicated with a horizontal bar, and non-terminal labels are all understood to be A. Note that the ITG builds only one tree, but the order in which a node's children are written to a stream can vary depending on the stream.

Canonical Form Bracketing Grammar

An ITG may be able to build several trees that correspond to the same alignment. For example, Figure 2.2 shows three trees that each correspond to a translation with no inversions, and therefore no re-orderings. They all correspond to the same alignment. It is often desirable to eliminate redundant structures when working with ITGs. Having a single, canonical tree structure for each possible alignment can help when flattening binary trees, as the use of the default structure can indicate arbitrary binarization decisions (Wu, 1997). Canonical structures also eliminate double counting when performing tasks like taking inside-outside expectations for EM (§ 2.2.2). The nature of $\emptyset$ link handling in ITG parsing makes eliminating all redundancies difficult, but we can at least eliminate them in the absence of the $\emptyset$ symbol. The redundant structures produced by the binary bracketing grammar (2.10) can be eliminated by replacing it with the canonical form grammar (Wu, 1997), which has the following productions:

$$\begin{array}{ll} \mathrm{S} \rightarrow & \mathrm{A} \mid \mathrm{B} \mid \mathrm{C} \\ \mathrm{A} \rightarrow & [\mathrm{AB}] \mid [\mathrm{BB}] \mid [\mathrm{CB}] \mid [\mathrm{AC}] \mid [\mathrm{BC}] \mid [\mathrm{CC}] \\ \mathrm{B} \rightarrow & \langle \mathrm{AA} \rangle \mid \langle \mathrm{BA} \rangle \mid \langle \mathrm{CA} \rangle \mid \langle \mathrm{AC} \rangle \mid \langle \mathrm{BC} \rangle \mid \langle \mathrm{CC} \rangle \\ \mathrm{C} \rightarrow & e/f \end{array} \tag{2.11}$$

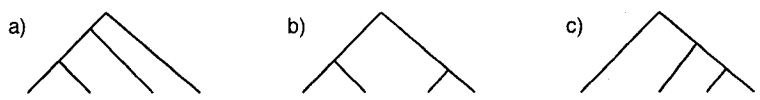

Figure 2.2: Three structures that result in the same alignment.

By design, this grammar allows only one structure per alignment. It works by restricting right-recursion to cases where the derivation alternates between inverted and straight productions. It will always construct Figure 2.2a, never considering the other two options. This grammar still covers the exact same alignment space as (2.10).

2.2.2 Probability Model and Learning

An ITG provides a syntactic method to generate bilingual sentence pairs. In the case of the canonical bracketing grammar, the process begins with an S symbol, and applies productions from (2.11) until all non-terminal symbols have been replaced by bilingual token pairs. The tree in Figure 2.1 shows one such derivation. By assigning probabilities to the grammar productions, one can create a generative model that assigns probability to sentence pairs. This called a stochastic ITG (Wu, 1997).

Stochastic ITGs employ a parameterization that is similar to monolingual probabilistic context-free-grammar. The probability of each production is conditioned on its left-hand non-terminal:

$$\Pr(\mathrm{X} \rightarrow \mathcal{L}) = \Pr(\mathcal{L}|\mathrm{X})$$

where $\mathcal{L}$ stands in for any sequence of symbols plus inversion information. For example, the $\mathrm{A} \rightarrow [\mathrm{AA}]$ production from (2.10) is parameterized with $\Pr([\mathrm{AA}]\,|\mathrm{A})$. This parameterization has the positive effect of factoring the trees created by the grammar. The probability of applying a particular production depends only on the current non-terminal and on the symbols it is generating, ignoring the rest of the tree.

Since ITG generates its two languages simultaneously, it produces a joint model of bitext, measuring $\Pr(\vec{e}, \vec{f})$ as opposed to the IBM models' conditional $\Pr(\vec{f}|\vec{e})$. Suppose we have a function $yield(D)$ that outputs the sentence pair produced by a complete ITG derivation tree D. The probability of a sentence pair $(\vec{e}, \vec{f})$ becomes the sum of the probabilities of its derivations:

$$\Pr(\vec{e}, \vec{f}) = \sum_{\{D|yield(D)=(\vec{e},\vec{f})\}} \Pr(D) \tag{2.12}$$

where a derivation is assigned probability according to its productions:

$$\Pr(D) = \prod_{(\mathrm{X}\rightarrow\mathcal{L})\in D} \Pr(\mathcal{L}|\mathrm{X}) \tag{2.13}$$

In the case of the canonical grammar in (2.11), and similarly for (2.10), the majority of the information in the stochastic grammar is stored in the many terminal probabilities $\Pr(e/f|\mathrm{C})$, as the variables e and f are instantiated over all English and French words.

This is similar to Melamed's joint word-to-word model (§ 2.1.2); in both cases probability mass is shared between all English-French word-pairs. Only the few non-terminal parameters, such as $\Pr([\mathrm{AB}]\,|\mathrm{A})$ deal with concept movement or distortion, and they serve mostly to memorize a context-free preference for or against inversion.

In early approaches, Wu (1997) employed an ad-hoc approach to parameter training, where a translation dictionary is constructed using IBM-1 and post-processing (Wu and Xia, 1995), and other parameters such as non-terminal probabilities and null probabilities are set by hand. This is unsatisfying from a machine learning stand-point. Fortunately, Zhang and Gildea (2004) have shown that the probabilities of all productions can be learned directly using EM. The E-step is performed by extending the inside-outside algorithm for CFG learning (Manning and Schütze, 2001) to the 2-dimensional bitext case, efficiently calculating expectations of productions over all derivations that yield the sentence pair. They initialize from a uniform probability, zero-knowledge bracketing grammar, and converge to a system which achieves word alignment accuracy that is better than IBM-2, but less accurate than an alignment HMM. We describe and analyze the algorithms required for doing alignment and training with a stochastic ITG in the following section.

2.2.3 Algorithms

There are three main dynamic programming algorithms employed to make use of ITGs:

1. **Parsing**: Determine the most likely derivation of a sentence pair.
2. **Inside**: Calculate the sum of all derivations of a sentence pair, as in (2.12).
3. **Outside**: Used together with inside to calculate expectations of production counts for a sentence pair.

We discuss each in turn, and call out similarities when they exist.

Parsing

ITG word alignment is performed by parsing. Given a stochastic ITG and a sentence pair $(\vec{e}, \vec{f})$, the goal in parsing is to find the most likely derivation that yields the sentence pair:

$$parse(\vec{e}, \vec{f}) = \mathrm{argmax}_{\{D|yield(D)=(\vec{e},\vec{f})\}} \left[\prod_{(\mathrm{X}\rightarrow\mathcal{L})\in D} \Pr(\mathcal{L}|\mathrm{X}) \right]$$

We will often refer to this most likely derivation as the Viterbi parse. We assume our ITG is in normal form (Wu, 1997), where all productions have the form: $\mathrm{X} \rightarrow [\mathrm{YZ}]$,

$X \rightarrow \langle YZ \rangle$, or $X \rightarrow e/f$. The binary bracketing grammar in (2.10) is already in normal form, while the canonical grammar in (2.11) can be put in normal form by expanding its unary S productions. Given these assumptions, ITG parsing can be accomplished with a 2-dimensional chart parser modeled after the CKY algorithm for monolingual parsing (Manning and Schütze, 2001). In describing this algorithm, we adopt a span notation, indicating substrings of sentences with start and end positions provided as subscripts and superscripts respectively. We index between words; for example, e_0^m would be the entire English sentence, while e_0^2 would include only the first two tokens.

The chart stores the score of the best analysis labeled with the non-terminal symbol X for the **bitext span** (e_s^t, f_u^v) in the entry $\beta(\mathrm{X}, e_s^t, f_u^v)$. The score for the best derivation of the sentence pair will eventually be found in $\beta(\mathrm{S}, e_0^m, f_0^n)$, where S is the start symbol of our grammar. Bitext spans covered by applicable terminal productions $X \rightarrow e/f$ are used to provide initial values for the chart:

$$\beta(\mathrm{X}, e_i^{i+1}, f_j^{j+1}) = \Pr(e_{i+1}/f_{j+1}|\mathrm{X})$$

The remaining entries can be built using a recurrence; the most likely labeled-constituent over a particular span will be constructed using a binary production to combine two other constituents, which are both optimal for their respective spans. This recurrence is implemented in the function max, shown in Algorithm 1. This algorithm iterates through bilingual split points (i, j), and tries all productions that combine the resulting constituents, as illustrated in Figure 2.3. The recurrence needs smaller spans to be available when

Algorithm 1 ITG parsing recurrence function $max(\mathrm{X}, e_s^t, f_u^v, \beta)$

```
ret ← 0
{Enumerating bilingual split points}
for i between s and t do
  for j between v and u do
    {Built by straight production}
    for (X → [YZ]) ∈ G do
      ret ← max [ret, β(Y, e_s^i, f_u^j) · β(Z, e_i^t, f_j^v) · Pr(X → [YZ])]
    end for
    {Built by inverted production}
    for (X → ⟨YZ⟩) ∈ G do
      ret ← max [ret, β(Y, e_s^i, f_j^v) · β(Z, e_i^t, f_u^j) · Pr(X → ⟨YZ⟩)]
    end for
  end for
end for
return ret
```

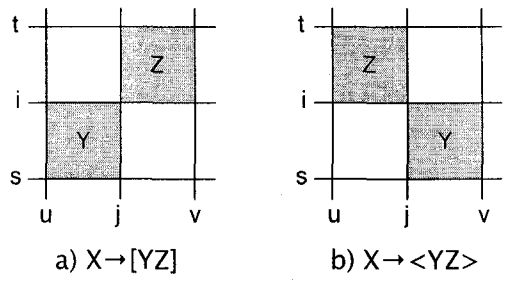

Figure 2.3: Constructing a bitext constituent from smaller constituents.

constructing the current span; therefore, spans are visited in order of increasing size. The CKY-inspired algorithm for visiting spans is shown in Algorithm 2.

Algorithm 2 CKY-style parsing for sentence pair (e_0^m, f_0^n) with ITG G

```
{Enumerating span sizes in both languages}
for l_E = 2 to m do
  for l_F = 2 to n do
    {Enumerating starting points for spans of length l}
    for s_E = 0 to m − l_E do
      for s_F = 0 to n − l_F do
        {Process spans with every possible non-terminal}
        for X ∈ G do
          β(X, e_{s_E}^{s_E+l_E}, f_{s_F}^{s_F+l_F}) ← max(X, e_{s_E}^{s_E+l_E}, f_{s_F}^{s_F+l_F}, β)
        end for
      end for
    end for
  end for
end for
```

From the four length-based loops in Algorithm 2, one can see that the algorithm visits $O(m^2n^2)$ spans. We will generally simplify this as $O(n^4)$, where n is understood to stand for the length of the longer sentence. In Algorithm 1 there are two loops used to enumerate bilingual split points, which are also bound by sentence length. This gives max a complexity of $O(n^2)$. Since it is nested inside Algorithm 2, the entire ITG parsing process has a time complexity of $O(n^6)$.

The parsing process presented here has been simplified to an extent. The algorithms shown above calculate only the probability of the most likely derivation, but the actual derivation can be recovered if each table entry maintains back-pointers specifying how it was constructed. Also, special processing required to handle the null symbol $\emptyset$ has been omitted from our explanation for the sake of clarity. The above algorithms can be extended to handle null links by using spans of length 0 to store cases where one language is unex-

plained by the other; for example, the entry $\beta(\mathrm{C}, e_1^1, f_1^2)$ is one of several entries that covers the case where the second French token is deleted during translation into English.

Parsing is well-defined for other ITG parameterizations. A binary ITG, where productions either do or do not exist, can be handled with the formalism above by assigning all existing productions probability 1. The algorithm will then find some valid derivation if one exists. A weighted ITG, where productions are given weights with no probabilistic interpretation, can be handled by replacing the multiplication of terms in Algorithm 1 with addition. Furthermore, CKY is just one of several possible parsing strategies. It suits our purposes, because its method for enumerating all spans also applies to our subsequent inside and outside algorithms. However, parsing does not necessarily need to enumerate all spans exhaustively, since it is taking a structured max. Recent work by Zhang and Gildea (2006a) has described an A* ITG Parser that increases efficiency while still guaranteeing an optimal parse. It works by placing potential constituents on a priority queue according to their probability plus an optimistic probability estimate for the remaining sentence pair. As constituents are removed from the queue they are guaranteed to be optimal, and can be added to the chart β to be used in the construction of new constituents.

Inside Algorithm

The inside algorithm calculates the total probability of a sentence pair (e_0^m, f_0^n):

$$\Pr(e_0^m, f_0^n) = \sum_{\{D|yield(D)=(e_0^m, f_0^n)\}} \left[\prod_{(\mathrm{X}\rightarrow\mathcal{L})\in D} \Pr(\mathcal{L}|\mathrm{X}) \right] \prod \Pr(\mathcal{L}|\mathrm{X}) \tag{2.14}$$

This probability is calculated using a dynamic programming process that is nearly identical to the parsing algorithms described above, placing the desired sum in $\beta(\mathrm{S}, e_0^m, f_0^n)$. Algorithm 2 replaces its call to *max* with a call to *sum*, which is simply Algorithm 1 modified so that its two max operations are replaced with sums. With this change, the table entries $\beta(\mathrm{X}, e_s^t, f_u^v)$ take on a different meaning; they now store the summed probability of all constituents rooted with an X that cover the span (e_s^t, f_u^v):

$$\beta(\mathrm{X}, e_s^t, f_u^v) = \Pr(e_s^t, f_u^v | \mathrm{X}_{s,u}^{t,v})$$

$\beta(\mathrm{X}, e_s^t, f_u^v)$ is the probability of producing (e_s^t, f_u^v) from X, which we refer to as the inside probability.[1] For the purposes of taking expectations in EM, these partial β terms are more important than the final sum shown in (2.14).

[1] $\beta(\mathrm{S}, e_0^m, f_0^n)$ provides the sum of all derivations for the entire sentence because the start symbol of our grammar is unambiguous: $\beta(\mathrm{S}, e_0^m, f_0^n) = \Pr(e_0^m, f_0^n | S_{0,0}^{m,n}) = \Pr(e_0^m, f_0^n)$

Outside Algorithm

The inside algorithm provides us with an efficient method to calculate $\Pr(e_s^t, f_u^v | \mathrm{X}_{s,u}^{t,v})$ for all $0 \leq s \leq t \leq m$ and $0 \leq u \leq v \leq n$. That is, for each constituent that is involved in the construction of (e_0^m, f_0^n), we can calculate its probability mass as if it were an independent tree with the start symbol X. However, EM's expectation task requires us to assess these partial structures in the context of the sentence pair that contains them. Put formally, we require $\Pr(e_0^m, f_0^n, \mathrm{X}_{s,u}^{t,v})$: the total probability of all complete derivations that also build an X over the indicated bitext span. A few transformations show how this can be calculated with the help of a second recurrence:

$$\begin{aligned} \Pr(e_0^m, f_0^n, \mathrm{X}_{s,u}^{t,v}) &= \Pr(e_0^s, e_s^t, e_t^m, f_0^u, f_u^v, f_v^n, \mathrm{X}_{s,u}^{t,v}) \\ &= \Pr(e_s^t, f_u^v | \mathrm{X}_{s,u}^{t,v}) \cdot \Pr(e_0^s, f_0^u, \mathrm{X}_{s,u}^{t,v}, e_t^m, f_v^n) \end{aligned} \tag{2.15}$$

The first term in (2.15) is inside probability, while the second term is outside probability, which measures the likelihood of producing what remains of the sentence pair, ignoring any portion of the derivation under X. Outside probability is stored in a second table:

$$\alpha(\mathrm{X}, e_s^t, f_u^v) = \Pr(e_0^s, f_0^u, \mathrm{X}_{s,u}^{t,v}, e_t^m, f_v^n)$$

Like inside probability, the table is constructed with a dynamic programming recurrence. However, there is only a single base case:

$$\alpha(\mathrm{S}, e_0^m, f_0^n) = \Pr(\mathrm{S}_{0,0}^{m,n}) = 1$$

The table is constructed in reverse order, from the largest spans to the smallest. The recurrence determines the probability of completing a bottom-up derivation from X by combining it with an adjacent constituent to create a larger constituent, which can then be assessed in terms of an already calculated outside probability. The process is illustrated in Figure 2.4. The recursion can be thought of as selecting a Z constituent that generates $\mathrm{X}_{s,u}^{t,v}$ and some Y. The Z symbol, along with the portion of the sentence pair surrounding Z, are scored with Z's outside probability, while the span under Y is scored with Y's inside probability. For example, an instance of the recurrence shown in Figure 2.4(a), with fixed values for Y, Z, i and j, calculates:

$$\begin{aligned} \Pr(e_0^s, f_0^u, \mathrm{X}_{s,u}^{t,v}, e_t^m, f_v^n, \mathrm{Y}_{t,j}^{i,u}, \mathrm{Z}_{s,j}^{i,v}) &= \Pr(e_0^s, f_0^j, \mathrm{Z}_{s,j}^{i,v}, e_i^m, f_v^n) \cdot \\ &\quad \Pr\left(\mathrm{Z}_{s,j}^{i,v} \rightarrow \left\langle \mathrm{X}_{s,u}^{t,v}, \mathrm{Y}_{t,j}^{i,u} \right\rangle\right) \cdot \\ &\quad \Pr(e_t^i, f_j^u | \mathrm{Y}_{t,j}^{i,u}) \\ &= \alpha(\mathrm{Z}, e_s^i, f_j^v) \cdot \Pr\left(\langle \mathrm{XY} \rangle | \mathrm{Z}\right) \cdot \beta(\mathrm{Y}, e_t^i, f_j^u) \end{aligned}$$

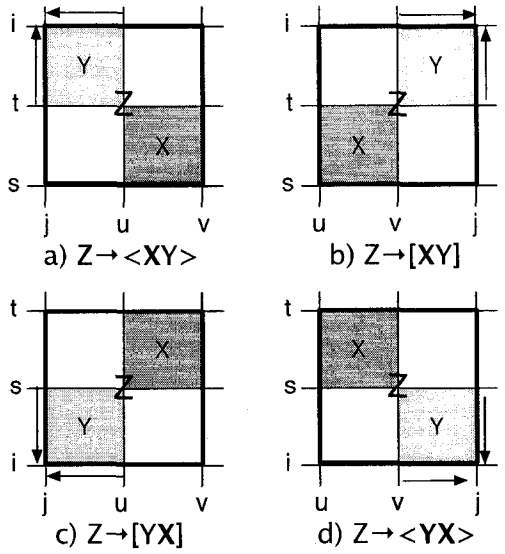

Figure 2.4: Outside recurrence – four ways to produce X from a larger Z.

Each pattern illustrated in Figure 2.4 covers a corner from which X can be extended. All four patterns, instantiated over all valid values of $\mathrm{Y}, \mathrm{Z}, i$, and j, are summed to calculate $\alpha(\mathrm{X}, e_s^t, f_u^v)$, effectively summing over every possible way to produce $\mathrm{X}_{s,u}^{t,v}$ from a larger Z. The actual pseudo-code is quite similar to the inside algorithm, but spans are visited in order of decreasing size, and we enumerate bilingual extension points out toward the edges of a sentence pair, rather than split points within a constituent. The inside algorithm is run first, so that β values will be available.

Given complete α and β tables, calculating expectations over productions is fairly straight-forward. Let us assume we are using the canonical grammar (2.11). For terminal productions, we can simply calculate the likelihood of building the pre-terminal C over a particular token pair with a direct application of (2.15):

$$\Pr(\mathrm{C} \rightarrow e_{i+1}/f_{j+1} | e_0^m, f_0^n) = \frac{\Pr(\mathrm{C}_{i,j}^{i+1,j+1}, e_0^m, f_0^n)}{\Pr(e_0^m, f_0^n)} = \frac{\alpha(\mathrm{C}, e_i^{i+1}, f_j^{j+1}) \cdot \beta(\mathrm{C}, e_i^{i+1}, f_j^{j+1})}{\beta(\mathrm{S}, e_0^m, f_0^n)}$$

Expectations for non-terminal productions can be calculated with a formulation that accounts for all derivations that use a specific production; for example:

$$\Pr\left(\mathrm{A}_{s,u}^{t,v} \rightarrow \left[\mathrm{B}_{s,u}^{i,j}, \mathrm{C}_{i,j}^{t,v}\right] | e_0^m, f_0^n\right) = \frac{\alpha(\mathrm{A}, e_s^t, f_u^v) \cdot \Pr([\mathrm{BC}] | \mathrm{A}) \cdot \beta(\mathrm{B}, e_s^i, f_u^j) \cdot \beta(\mathrm{C}, e_i^t, f_j^v)}{\beta(\mathrm{S}, e_0^m, f_0^n)}$$

The inside, outside, and expectation calculation algorithms all have a time complexity of $O(n^6)$, making each expectation step of EM also $O(n^6)$. This makes unsupervised ITG training expensive, but like the alignment HMM (§ 2.1.1), training does benefit from the exact expectations made available by inside-outside.

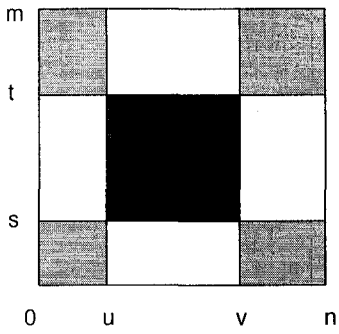

Figure 2.5: Cells considered when assessing (e_s^t, f_u^v) for pruning.

Scaling

ITG parsing and expectation algorithms have polynomial time complexity, but at $O(n^6)$, the polynomial is very large. Motivated by an even more expensive bitext model (§ 2.3.1), Zhang and Gildea (2005) developed a pruning method that helps speed up all forms of ITG processing. ITG algorithms work by analyzing $O(n^4)$ bitext spans, and each analysis takes $O(n^2)$ time. An effective approach to speed up these algorithms is to eliminate unlikely spans as a preprocessing step, assigning them 0 probability and saving the time spent processing them. This alters the alignment space searched by the ITG, but if the eliminated spans are not used in the correct derivation, the alteration is harmless.

Tic-tac-toe pruning (Zhang and Gildea, 2005) judges bilingual spans according to the IBM Model 1 probability inside and outside of the span. That is, the merit of (e_s^t, f_u^v) can be approximated with:

$$\text{FOM}(e_s^t, f_u^v) = \Pr_{\text{IBM1}}(f_u^v | e_s^t) \cdot \Pr_{\text{IBM1}}(f_0^u, f_v^n | e_0^s, e_t^m) \tag{2.16}$$

The portions of bitext that are considered by (2.16) are illustrated in Figure 2.5. The first term of covers the black area, while the second term covers the grey area. A $O(n^4)$ dynamic programming algorithm exists to calculate the above figure of merit (FOM) for all spans efficiently. Bitext spans can then be pruned according to a beam, where all spans (e_s^t, f_u^v) whose FOM is not within a certain ratio of $\max_{f_i^j} \text{FOM}(e_s^t, f_i^j)$ are pruned. Zhang and Gildea (2005) safely prune 70% of their spans using these probabilities. The idea of pruning spans will be a central notion in two of the methods proposed in this thesis. However, we will sometimes do so to provide the ITG with extra information, with the increase in efficiency being a pleasant side-effect.

2.2.4 Alignment Space

ITG parsing with either (2.10) or (2.11) is a one-to-one alignment method, because each terminal production creates only a single link. See § 2.4.3 for many-to-many ITG systems. ITG is also the first method we have reviewed that considers fewer than all possible concept re-orderings during translation. We will refer to the space that allows all possible re-orderings as **permutation space** and one that allows only ITG's restricted re-orderings as **ITG space**. When a binary bracketing ITG is used to model translation, only re-orderings that can be explained using inversions in some binary constituency tree will be explored. ITG space is identical to permutation space when either sentence has fewer than 4 words, but it is smaller than permutation space at any point after that. By the time both sentences have 7 words, the ITG is exploring only 75% of the alignments in permutation space; as sentence length reaches 15, ITG explores only 2% of possible permutations (Wu, 1997). We refer to these implicit constraints on re-ordering as **ITG constraints**.

The alignments disallowed by ITGs can be characterized by two forbidden structures shown in Figure 2.6. These are called "inside-out" alignments by Wu (1997) for their shape. These alignments cannot be formed by a binary ITG because no sequence of binary inversions can re-order "1 2 3 4" as indicated by either alignment, to either "2 4 1 3" or "3 1 4 2". Another way to see why the alignments cannot be built is to look at the links as fixed terminals, and to try to build a binary tree over them. Figure 2.7 depicts the two forbidden structures as alignment matrices. Note that no two dots can be enclosed in a rectangle without excluding a third dot, allowing no binary constituents to form in bitext without accounting for discontinuous constituents (Melamed, 2003). At the core of the issue is the fact that no two tokens that are adjacent in the top stream remain adjacent when projected onto the bottom stream.

ITG re-orderings are syntactically motivated, so there is a strong intuition that they should rarely be violated in natural translations (Wu, 1997). Some empirical evidence exists to back up this claim (Zens and Ney, 2003) for English-French, though compact examples of ITG-incompatible alignments do exist for other language pairs (Melamed, 2003).

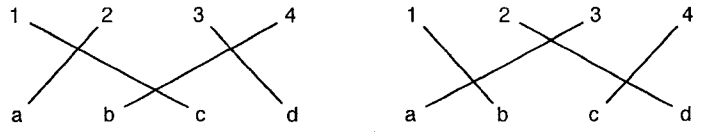

Figure 2.6: The two forbidden structures disallowed by ITG alignment.

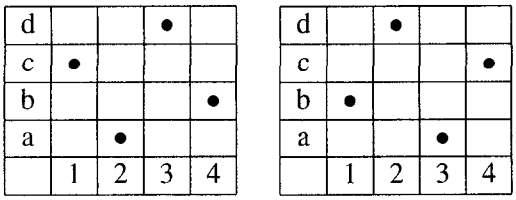

Figure 2.7: The forbidden structures in Figure 2.6, drawn as alignment matrices.

A recent empirical study over five language pairs (Wellington et al., 2006b) showed that in its weakest language pair, English-Chinese, an ITG is unable to reproduce 5% of the alignments found in a hand-aligned bitext. In the other language pairs the ITG lost only 2% or fewer alignments. We independently and simultaneously conducted a similar study for English-French, described in § 5.3. Because of their syntactic motivation and the existence of efficient algorithms, the ITG re-orderings have been studied carefully. Zens and Ney (2003) explore the re-orderings allowed by ITG parsing, and provide a formulation for the number of re-orderings that can be produced for a sentence with m words. Zhang et al. (2006) provide a linear-time shift-reduce parser to determine if a given permutation can be generated by a binary ITG.

2.2.5 Discussion

As a model of bitext, ITG has certain strengths and weaknesses, which we review briefly here. Its strengths include:

- Polynomial-time algorithms exist for alignment and EM training.
- Both syntactic and alignment structures are explicitly represented in the formalism.
- The alignment space is syntactically-motivated.
- The grammar and formalism are clean and easy to work with.

We strive to take advantage of ITG's representational power in our methods, and we attempt to measure the contributions of ITG's inherent strength in our experiments. However, the benefits of ITG do come at some cost. Weaknesses of ITG include:

- The algorithms for training and alignment are slow, even with tic-tac-toe pruning.
- The generative nature of EM training means that new features must be added carefully, and it is unclear how to incorporate any available labeled data.
- The same ITG constraint that enables polynomial time algorithms prevents the system from modeling all human translations.

Some of these weaknesses have been addressed by other syntactic methods for bitext analysis that will be reviewed in the following section. We address the issues of speed in Chapter 4, and we examine feature representation and training in Chapter 5.

2.3 Other Syntactic Solutions

In this section, we review alternate approaches for using syntax in bitext analysis. First we describe two extensions and generalizations of ITG: namely lexicalized ITG and MTG, which is followed by a discussion of fixed-tree approaches for incorporating syntax.

2.3.1 Lexicalized ITG

Inspired by the use of lexicalization in monolingual parsing (Manning and Schütze, 2001), where non-terminals are annotated with a head-word selected from among their descendants, Zhang and Gildea (2005; 2006b) introduced methods to lexicalize the non-terminal symbols of an ITG. The richer non-terminal set provided by lexicalization could help inform inversion decisions higher in the tree, creating a more intelligent distortion model. They explored two approaches, bilingual lexicalization and monolingual bilexicalization.

In a bilingually lexicalized ITG (Zhang and Gildea, 2005), or BL-ITG, each non-terminal is annotated with a bilingual head-word pair (e, f), selected from among the heads of its children. In standard ITG there are two types of binary productions, straight and inverted. The additional head selection decision creates four production types in BL-ITG:

$$\mathrm{X}(e/f) \rightarrow [\mathrm{Y}(e/f), \mathrm{Z}] \mid [\mathrm{Y}, \mathrm{Z}(e/f)] \mid \langle \mathrm{Y}(e/f), \mathrm{Z} \rangle \mid \langle \mathrm{Y}, \mathrm{Z}(e/f) \rangle$$

Note that the head of the unselected child is ignored in each production. The grammar is given the ability to select a head from between either of its two children, and it is left up to EM to learn which nodes are heads. The added task of selecting a head word makes BL-ITG algorithms more expensive, with time complexities of $O(n^8)$. From their experiments, lexicalization does not appear to have a dramatic effect on alignment quality.

In monolingually bilexicalized ITG, or MB-ITG, each non-terminal is annotated with a monolingual head word, selected from the heads of its children. Since the system is bilexicalized, the heads of both children are considered in productions. MB-ITG also has four types of binary productions:

$$\mathrm{X}(e) \rightarrow [\mathrm{Y}(e), \mathrm{Z}(e')] \mid [\mathrm{Y}(e'), \mathrm{Z}(e)] \mid \langle \mathrm{Y}(e), \mathrm{Z}(e') \rangle \mid \langle \mathrm{Y}(e'), \mathrm{Z}(e) \rangle$$

This extension is motivated by the hope that it will create more meaningful parameters for non-terminal productions, improving the distortion component of their model, but also improving the dependencies produced by the head-annotated tree built over the English sentence. A naïve dynamic programming implementation would have $O(n^8)$ time-complexity, as each split has any extra $O(n^2)$ term to consider as potential head words are enumerated for the two right-hand-side constituents. However, they show that the hook trick for monolingual lexicalized parsing (Eisner and Satta, 1999) applies, reducing time complexity to $O(n^7)$. Their experiments indicate that the addition of bilexicalization to the canonical bracketing ITG has no positive effect on alignment quality, but it does improve the quality of the resulting dependencies.

2.3.2 Generalized Multitext Parsing

ITG is an interesting and powerful formalism, and it has received a lot of attention in this review, as it is our preferred method for bitext analysis. However, it is not the only way to approach the use of syntax in translation. For example, an alternative method is to treat the translation process as a collection of finite-state head transducers (Alshawi et al., 2000) that are applied hierarchically to a sentence. This method is naturally lexical, as it draws parallels to dependency parsing where ITG draws parallels to CFG. These two approaches are different, but clearly related in some way. Multitext grammars or MTGs (Melamed, 2003), and Generalized Multitext Grammars or GMTGs (Melamed et al., 2004) are attempts to generalize these various syntactic models of translation, in order to expose similarities, extensions, and connections to monolingual parsing. They do so by completely generalizing CFGs to a multitext setting (Melamed and Wang, 2006), instead of simply adding transduction and inversion operators to their productions.

Completely explaining GMTG can be a lengthy undertaking. For the purposes of this document, it should be sufficient to provide some intuitions through a series of examples. A possible GMTG production is shown below:

$$\begin{bmatrix} (\mathrm{X}) \\ (\mathrm{Y}) \\ (\mathrm{X}) \end{bmatrix} \rightarrow \begin{bmatrix} (\mathrm{A}^1\mathrm{C}^3) \\ (\mathrm{A}^2\mathrm{D}^1) \\ (\mathrm{B}^4\mathrm{A}^1\mathrm{C}^2) \end{bmatrix} \tag{2.17}$$

Each row corresponds to a stream, so we can immediately see a generalization, the notation allows more than two streams. The left-hand-side indicates that this production expands Xs in streams 1 and 3 and a Y in stream 2, where all three non-terminals are understood to be linked. This highlights another extension, linked non-terminals no longer need to share

names. This does not add any representational power, but it does allow eloquent handling of cases where a concept is expressed as a verb in one language, but as a noun in the other. Links on the right-hand-side are indicated by non-terminals that share superscripts. For example, Link 1 produces the concept $[(\mathrm{A}), (\mathrm{D}), (\mathrm{A})]$ across the three streams. Right-hand-side terms are always understood to be written in left to right order. The inversion operator from ITG has been generalized by this superscript link notation, allowing arbitrary re-ordering of concepts to be specified in the grammar. Since productions need not be binary, this allows the creation of rules that produce non-ITG orderings. Finally, note that links 3 and 4 are each present in only one stream. These monolingual non-terminals would be then be re-written with monolingual productions, such as:

$$\begin{bmatrix} (\mathrm{C}) \\ () \\ () \end{bmatrix} \rightarrow \begin{bmatrix} (\mathrm{Adj\ NP}) \\ () \\ () \end{bmatrix} \tag{2.18}$$

that are free to extend into an arbitrarily large monolingual grammar. Having the synchronous grammar diverge at some point in this manner would be how phrasal translation and alignment are handled (Melamed et al., 2004). This also means that non-terminals can be deleted in one language high in the tree, instead of persisting until a leaf node containing the empty word is produced. This null handling can help eliminate redundancies in parsing.

Both MTGs and GMTGs have normal forms that are analogous to Chomsky normal form (Hopcroft and Ullman, 1979) and admit faster parsing algorithms. However, this involves binarization, which will prevent a traditional multitext parser from forming certain re-orderings. Any binary grammar will run into the same limits we encountered in the ITG case in § 2.2.4. Where we tend to view these limits as a positive aspect of ITG, MTG researchers see them as limitations that need to be overcome. Literature exists on parsing with discontinuous constituents (Johnson, 1985), which can be extended to the bitext case, as is done in (Melamed, 2003; Melamed, 2004). This allows non-ITG re-orderings to be constructed with a binary grammar by tracking multiple span boundaries for designated discontinuous constituents. This comes at an expense, though, significantly increasing the complexity of parsing.

GMTG is a highly expressive formalism; it is much more attractive to try to hand-craft a grammar in this setting than it is in ITG. However, that would still be a huge undertaking, especially for situations with more than 2 streams. A practical method for learning a GMTG from data is presented in (Melamed, 2004; Melamed and Wang, 2006). The approach requires the following:

- A bitext for the language pair (L1, L2).
- A monolingual, stochastic CFG for L1, learned from a monolingual treebank.
- A stochastic word-to-word translation model relating L1 and L2, learned from bitext using some sort of alignment method.

Given these resources, a specialized GMTG and multitext parser can be used to simultaneously discover structure in L1 and align L1 to L2, projecting the discovered structure onto L2. A GMTG can then be learned from the production counts observed in this bitext. The learned GMTG can then be used to actually guide a complete SMT system, also specified as a generalization of GMTG parsing.

The work on GMTG is still highly theoretical. Though an implementation of a GMTG parser does exist, few empirical results have been reported to the research community. The results that have been reported do not begin to test the full flexibility of this formalism; work presented in (Wellington et al., 2006a) is very similar in terms of expressiveness to (Yamada and Knight, 2002), and is actually more restrictive in its re-ordering than a binary ITG. As of now, it appears that GMTG is actually too expressive for the sorts of bitext analysis and alignment tasks where ITGs are currently employed. For example, it is not clear how one would formulate something roughly equivalent to a language-independent bracketing grammar in this formalism, while still leveraging its advantages over ITG.

2.3.3 Tree-to-string Methods

Syntax can also be added to translation and alignment without directly assuming a synchronous grammar generated strings in both languages. The growing accuracy of monolingual parsing technology allows us to inject syntax by simply parsing one of the two languages (usually English), allowing us to model translation as if it were the English tree being translated into French. Most prevalent among the methods that do so is the tree-to-string translation model of Yamada and Knight (2001; 2002).

Tree-to-string translation model

The tree-to-string model is an attempt to apply the rigorous statistical modeling techniques from the IBM models in a syntactic setting. This means a return to conditional translation modeling, where an English constituency tree T generates a French sentence $\vec{f}$ according to $\Pr(\vec{f}|T)$. Like IBM-3 and 4, this model is based on a generative process centered around the English sentence, but now the generative process begins with and operates on a

tree. The hope is that by modeling movement and insertion in the tree, as opposed to the string, the syntactic model can accurately characterize the long-range movement observed in more distant language pairs, such as Japanese-English. Just like the IBM models can be generalized using finite state machinery (Knight and Al-Onaizan, 1998), this notion of transforming a tree can be generalized using a tree transducer (Graehl and Knight, 2004).

The generative story begins by permuting the children of each parent node according to a distribution conditioned on the original sequence of children. Next, a French word can be inserted to the left or right of any node in the tree. Finally, English leaf nodes are replaced with French tokens, drawn from a familiar IBM-style translation distribution. This process provides a probability model for the transformation of an English tree into a new tree with French leaf nodes. The transformed tree is then flattened into a string as a free operation. This creates some redundancies, as several transformed trees will flatten to the same French string, so all of these possible trees need to summed to arrive at $\Pr(\vec{f}|T)$.

The tree-to-string model is trained using EM. Because all re-orderings are based on the tree structure, and because the model is designed to not violate CFG-style independence assumptions, an efficient E-step can be conducted using something resembling the inside-outside algorithm. The alignment task is formulated as a search for the most likely alignment according to the model for a tree-sentence pair, which also benefits from the tree structure. However, relying on this structure has its disadvantages. All re-ordering is limited to permutations of children, and in the case of trees with deep syntactic structure, this may not correspond to the ordering in the French sentence. To mitigate this problem, one can selectively flatten the input tree as a pre-processing step.

The tree-to-string model showed promising alignment performance on a small dataset (Yamada and Knight, 2001), and was followed by a string-to-tree decoder in (Yamada and Knight, 2002), which actually parsed a French sentence into an English tree. This decoder has since been replaced by a more general tree-transducer decoder, which is powered by GIZA++ alignments. One of the advantages of these transducers is that they allow the transformation of multiple levels of the tree at once, bypassing the limitation of only being able to permute children within the existing tree structure. These transducers are discussed in more detail § 2.6.4.

Syntactic cohesion

When concept movement is limited to re-ordering children in a fixed tree structure, this has the effect of forcing phrases, defined by subtrees, to move together during transla-

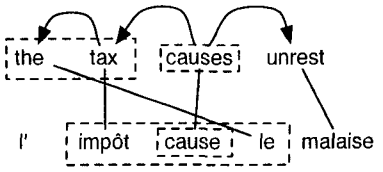

Figure 2.8: A violation of phrasal cohesion in a dependency tree.

tion. Fox (2002) measured this **syntactic cohesion** in gold standard alignments by counting crossings.[2] Crossings occur when the projections of two disjoint phrases overlap. For example, Figure 2.8 shows a head-modifier crossing: the projection of the "the tax" subtree, "impôt ... le", is interrupted by the projection of its head, "cause". Alignments with no crossings maintain syntactic cohesion. Fox's experiments show that cohesion is generally maintained for English-French, and that dependency trees produce a higher degree of cohesion than constituency trees. However, her experiments also show that there are cases where one would prefer to ignore syntactic cohesion. These include discontiguous phrases such as "ne... pas", parse errors, linguistic exceptions, and paraphrase.

Lin and Cherry (2003) used the notion of syntactic cohesion to constrain a beam search word-aligner according to an English dependency tree. They showed that a competitive-linking-style aligner, guided with the ϕ^2 statistic and a limited position model, could benefit greatly by ruling out all alignments that violated syntactic cohesion. Figure 2.8 is an example of a case where a word-to-word model can benefit from this constraint, as the incorrect *the—le* link is ruled out. This cohesion constraint is generally a more strict constraint on concept re-ordering than the ITG constraints (Wellington et al., 2006b). We explore the relationship between the two directly in § 5.3. Zhang and Gildea (2004) conducted experiments comparing ITG to the tree-to-string model, concluding that the ITG was better suited to alignment because it does not rely on a fixed tree. We challenge this conclusion in § 5.3. Lopez et al. (2002) attempted to constrain the head-transducer translation model (Alshawi et al., 2000) with a fixed dependency tree. However, they were not able to out-perform GIZA++, even when using its alignments in training. This may be because they constrain according to head assignment as well as syntactic cohesion, which we will show to have stronger movement restrictions than syntactic cohesion alone in § 5.3.

In the mean time, Gildea (2003) developed a method to allow the tree-to-string model to form alignments that violate syntactic cohesion at a penalty. He did this by introducing

[2]Fox used the term "phrasal cohesion" to refer to this phenomenon. However, since the non-syntactic meaning of the term "phrase" is already in heavy use in MT, referring to any contiguous string of tokens, we adopt the term "syntactic cohesion" instead.

a cloning operator to the model. This allowed entire subtrees of the English tree to be inserted during the insertion stage of the generative story, essentially duplicating parts of the tree elsewhere. This coupled with deletion allows subtree movement beyond child re-ordering, without increasing the asymptotic complexity of the search or learning processes. We develop our own method to treat syntactic cohesion as a soft constraint in § 5.4.

2.3.4 Tree-to-tree Methods

If there are advantages to aligning a tree to a string, then the next logical step is to parse both languages and align two trees. There are a few problems with this idea. First and foremost is the lack of accurate parsers in many languages; however, treebanks and therefore probabilistic parsers do exist for some non-English languages, notably Chinese and Czech. Second, two grammars that each maximize monolingual coverage and accuracy for different languages may not be compatible with each other for translation modeling. This means that trees need to undergo internal structural transformations in addition to child re-ordering and re-labeling during translation. That is, the alignment algorithm must have some mechanism to relate two non-isomorphic trees. We summarize three tree-to-tree alignment methods here. The summaries are very brief, as we will be working with at most one parse tree in the research proposed here, both to avoid problems with matching internal tree structure, and to increase language independence.

The tree-to-string model of translation (Yamada and Knight, 2001) is already transforming an English tree into a French tree. If we remove the flattening operation at the end, it is a model for isomorphic tree transformation. Therefore, it can be extended to transform its input into a non-isomorphic tree. One method to do this is Gildea's cloning operator (2003); however, that does not help in non-terminal correspondences, and it cannot easily match trees with different depths. In (Gildea, 2003), the tree-to-string model is explicitly extended for tree-to-tree alignment. This is done by adding two new operators. One has a single English non-terminal node produce two nodes in the French tree, while the other has two English nodes grouped together to produce one node in the French tree. These operators do not provide much in the way of concept re-ordering power, but they are valuable in matching internal tree structure. Gildea also tested the belief that dependency trees are better suited to translation modeling than constituency trees (2004). He did so by adapting the tree-to-tree model to dependency trees, and testing the two models on the same data set. The flattened constituency tree was shown to perform better. One potential reason for the decreased performance of the dependency structure is because the two tree

structures must agree in the selection of head words, in addition to agreeing on the basic tree structure.

A generalization of the tree-to-tree operations introduced in (Gildea, 2003) is synchronous tree-substitution grammar (Eisner, 2003). In this formalism, one tree is transformed into another non-isomorphic tree using a series of productions. The productions take the form of pairs of tree fragments, where some nodes are labeled and some are left as variables to be assigned. Transformation is conducted by applying productions until no more variables remain. In (Eisner, 2003), a rich theoretical formalism is provided for both synchronous and monolingual tree-substitution grammars, and an algorithm is sketched to learn productions from unaligned tree pairs. However, no results in alignment or translation quality are provided.

Finally, in (Ding et al., 2003) it is observed that a model might conduct many-to-many alignment and tree-to-tree alignment at the same time. Tree structure is observed and obeyed at a high level, but once non-isomorphisms begin to make exact word-to-word alignments difficult, the algorithm has the option to back off to a bag-of-words model, and simply link the two non-isomorphic subtrees in their entirety. The only statistical model trained here is IBM-1, and they begin by training it on the entire dependency-parsed corpus, ignoring any tree structure. Then high-certainty links are established in the tree structures according to heuristics based on the IBM-1 probabilities. These links are fixed, and used to partition the trees into bilingual fragment-pairs. These fragments are used as the new corpus for the next round of IBM-1 training. Training continues until no more partitions can be safely established. This method is shown to produce superior word alignments to IBM-4, even though it does not model movement directly, and only considers a greedy decomposition of the bitext.

2.4 Phrasal Bitext Analysis

Up until this point, the word alignment systems we have reviewed have been primarily one-to-one or one-to-many systems. That is, in at least one of the languages being aligned, tokens can participate in at most one link. This restriction makes model storage tractable by limiting the translation table to bilingual word pairs, and it enables efficient computation by constraining the number of allowed alignments.

However, this restriction is unrealistic from a linguistic point of view. One does not need to look far to find cases where one would prefer to model several words in the source

language that translate into several words in the target. Non-compositional phrases and named entities often require this sort of modeling. To take an extreme example, the English phrase "Hold your horses" can be translated into Canadian French as "Wo les moteurs" (literally "wo the motors") to retain the phrase's basic meaning, tone and idiomatic nature. In this case, the desired alignment is a component that connects the two phrases completely.

Disallowing many-to-many alignments can also cause problems for a major word alignment consumer: phrase-based statistical translation systems (§ 2.6.1). These systems build phrase translation tables based on all phrases that are consistent with a fixed word alignment; where consistency is defined so that links only ever rule out possible phrase pairs. If input alignments are artificially sparse due to a one-to-many restriction, then the phrase extractor will over-generate, populating the table with useless entries and diluting probability mass away from correct phrase pairs.

Systems that attempt many-to-many alignment are often referred to as phrasal alignment systems or phrasal translation models. We use the terms many-to-many and phrasal interchangeably. Phrasal systems are few and far between, due to the difficulties encountered in storing and enumerating phrasal translation pairs. Five strategies for analyzing bitext in a phrasal setting are reviewed here.

2.4.1 Heuristic Combination

The most common method to create many-to-many alignments is to combine two one-to-many alignments of the same bitext, where the second alignment has its "many" and "one" roles reversed with respect to the first. This technique was introduced in (Och and Ney, 2000b) to improve GIZA++ alignment. There are three primary combination methods: intersection, union, and heuristic combination.

Intersection includes a link in the combined alignment only if it belongs to both one-to-many alignments. This results in high precision, low recall alignments; guessed links are usually correct, but many correct links are missed. Since each component alignment has one side where tokens can participate in at most one link, the combined alignment must be one-to-one. Clearly this does not solve the many-to-many problem, but the high-precision links from this combination can be used as guides for other systems (Taskar et al., 2005; Birch et al., 2006).

Union includes a link in the combined alignment if it belongs to either component alignment. This results in high recall, low precision alignments. Union alignments are many-to-many in the broadest sense of the term, there are no constraints on what sort of link

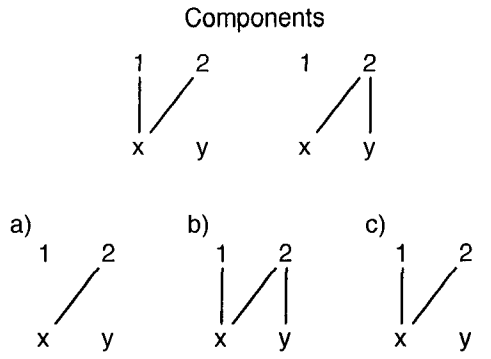

Figure 2.9: Examples of alignment combination for (a) intersection, (b) union and (c) heuristic strategies.

patterns can form; they exist in the unconstrained alignment space (§ 1.2). In this case, the size of the search space is immaterial, as the union alignment is defined deterministically in terms of its fixed component alignments. Union alignments can go too far in their leniency, allowing counter-intuitive links, such as in Figure 2.9b, where it is not clear what the exact relationship is between "1 2" and "x y." How can '2' explain 'x' and 'y', while '1' explains only 'x'? We will make some of these intuitions formal in § 2.4.2.

To strike a favorable balance between precision and recall, Och and Ney (2000b) propose heuristic, or refined alignment combination. These methods start from the intersection and grow the alignment out, adding links from the union one by one. Potential links (e_i, f_j) can be added to the combined alignment A only if:

$$\forall(e_{i'}, f_{j'}) \in A : i' \neq i \text{ or } \forall(e_{i'}, f_{j'}) \in A : j' \neq j \tag{2.19}$$

That is, an added link must always connect at least one previously unaligned token. Although they do not state so explicitly, (2.19) will prevent a system from completely linking a multiword phrase in one language to a multiword phrase in another. Instead, it can form one-to-many sub-alignments in both translation directions at once. This general strategy defines a family of possible heuristic algorithms. The order in which links are considered for addition will affect the output of the combination. For example, the alignment in Figure 2.9c is one of two acceptable combinations. Koehn et al. (2003) specify a well-defined algorithm for heuristic expansion that follows rule (2.19), called **grow-diag-final** (GDF). Their method links token pairs where exactly one token is already linked by A first, and then allows links to form between two previously unlinked tokens. They test several variants of this strategy in the context of phrase-table extraction for phrasal decoding. Interestingly, they found that the heuristic combination strategy had a greater effect on translation perfor-

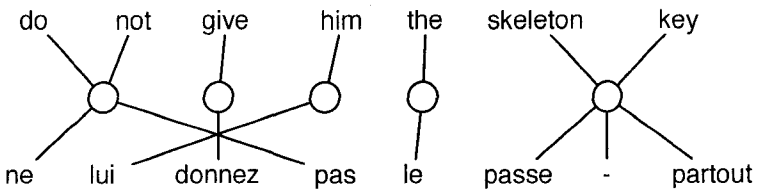

Figure 2.10: Many-to-many alignment with cepts

mance than the alignment method used to create the component alignments.

2.4.2 Matrix Factorization

Goutte et al. (2004) begin their attempt at many-to-many alignment by clearly defining the alignment space they are searching. In particular, they restrict their many-to-many links according to **transitive closure**. Transitive closure states that if f_k and f_l are both linked to e_i, then any other English token e_j linked to f_k must also be linked to f_l. This directly eliminates Figure 2.9b from alignment space. Heuristic combination (§ 2.4.1) is unable to achieve true many-to-many alignment because there is no guarantee that a partial many-to-many component will complete its transitive closure as links are added one by one from the union. To define a process that maintains transitive closure naturally, Goutte et al. (2004) make use of cepts (Brown et al., 1993; Marcu and Wong, 2002). Instead of directly linking tokens between languages, each token is linked to exactly one concept or cept. Tokens linked to the same cept are linked in the corresponding alignment. For example, in Figure 2.10 there are five cepts, and the phrasal translation of "skeleton key" to "passe-partout" is handled with one cept. Note that unlike in (Marcu and Wong, 2002), members of the same cept do not need to be contiguous.

They conduct their search for transitive many-to-many alignments using matrix factorization. Recall that one can view an alignment A as a binary m-by-n matrix, where a link between e_i and f_j in A is indicated by $A_{i,j} = 1$. Given the appropriate number of cepts c, any transitive alignment should factor into two matrices, one m-by-c and one c-by-n, that indicate the corresponding links to cepts. To create word alignments, they use Probabilistic Latent Semantic Analysis or PLSA (Hofmann, 1999) to do a probabilistic factorization of an m-by-n association matrix, where entries are counts indicating word pairs' bilingual affinity. This produces a probability distribution that can be used to assign each token to its most likely cept. This process requires counts for the (e_i, f_j) pairs, which are provided by sampling the K-best GIZA++ alignments of the sentence pair: a token-pair's count is

the number of alignments in which GIZA++ decided to link that pair. In this regard, this method is similar to heuristic combination: no new global model of alignment is determined from data, but instead the information from a one-to-many model is re-used to create many-to-many alignments. To select the number of cepts, they try several cept counts, to find which count maximizes likelihood with a complexity penalty.

Though it produces constrained many-to-many alignments in a theoretically motivated manner, this method is not without its problems. The PLSA alignment search is fueled by EM (Dempster et al., 1977), which is a local maximizer, and therefore not guaranteed to find the optimal alignment. Also, despite employing a probability model and EM, the aligner works on each sentence-pair in isolation. Since no global phrasal model is estimated, evidence for a phrase cannot be accumulated over many sentence pairs.

2.4.3 Recursive Alignment

Synchronous parsing approaches to alignment (§ 2.2) will naturally decompose a sentence pair into smaller and smaller bitext segments with their recursive, parser-like structure. In fact, a binary synchronous parser can be viewed as a divide-and-conquer strategy for word alignment (Vilar and Vidal, 2005). Given a bilingual sequence pair (e_0^m, f_0^n), these methods select a bilingual split-point pair (i, j) where $0 \leq i \leq m$ and $0 \leq j \leq n$, divide each sequence into two, and then recursively process the resulting sequence-pairs, such as (e_0^i, f_0^j). Any phrasal method based off this hierarchical process will create phrases with transitive closure automatically. The phrases will also have the added constraint that they must be contiguous.

Wu (1997) explored many-to-many alignment in this context, using a technique he calls "translation-driven segmentation" to align English and Chinese, while simultaneously segmenting the Chinese sentence[3]. This approach assumes a provided probabilistic, phrasal translation dictionary. Given said dictionary, during Viterbi alignment any sequence pair receives a probability:

$$\Pr(e_i^j, f_k^l) = \max\left[\Pr_{ITG}(e_i^j, f_k^l), \Pr_{Dict}(e_i^j, f_k^l)\right] \tag{2.20}$$

where $\Pr_{ITG}$ provides the probability of (e_i^j, f_k^l)'s best recursive decomposition, and $\Pr_{Dict}$ is supplied by simply looking up (e_i^j, f_k^l) in the phrasal dictionary. In cases where the dictionary wins the max, the words are linked many-to-many. This allows a fluid segmentation

[3]Chinese does not use white space to separate words, making the segmentation task non-trivial. However, the need for many-to-many alignment in any language indicates that the selected segmentation was suboptimal for the alignment task.

for both languages that is aware of the quality of alignment options for a given segmentation. In their experiments, the dictionary was constructed from a sentence-aligned bitext using monolingual Chinese segmentation techniques, IBM-1 alignment, and dictionary filtering (Wu and Xia, 1995). So again, the construction of the translation model does not benefit from the notion of bitext phrases. This many-to-many alignment search is no more expensive than standard synchronous parsing, as it only checks phrase pairs that would have eventually been decomposed anyway. However, the question of how to properly derive the necessary phrasal dictionary remains.

Vilar and Vidal (2005) make use of this same idea, but with a significantly more complicated probability model. In this case, the phrasal dictionary is replaced with an IBM-1 translation model. At any point, their system has the option to stop dividing, create a phrasal alignment, and receive the IBM-1 translation probability for the phrase pair. Their model is quite different from (Wu, 1997); it is a conditional translation model as opposed to a probabilistic synchronous grammar, and it models several factors Wu does not, such as the probability of whether to split or to use IBM-1, and the probability of selecting a particular split point. Furthermore, they train the entire model unsupervised using an EM-like process. By backing off to IBM-1, they allow many-to-many alignment without having to store a potentially overwhelming phrasal translation table. All statistics are word-to-word, but the system is trained in a phrasally aware manner.

Yamada and Knight (2001) present a syntactic tree-to-string translation model (§ 2.3.3). In (Yamada and Knight, 2002), they take advantage of the hierarchical structure of their model to introduce phrases. Their statistical phrase model resembles IBM-3: an English phrase generates a fertility l, and then generates l French words independently. They require a translation table that is phrasal only on the English side. Furthermore, since their technique requires an input English parse tree, only sequences that are constituents in the English tree are attempted as phrasal translations. Attempted phrases are also limited so that phrase lengths would not differ greatly across languages. The model has a choice between phrasal translation and continued decomposition. Like in (Vilar and Vidal, 2005), the phrasal model is incorporated into training.

2.4.4 Information Theoretic Model Re-estimation

Melamed (1997) proposed an iterative method to discover non-compositional phrases in bitext. A phrase is non-compositional with respect to translation if it is translated differently than its component words taken alone. To help motivate this phrase discovery task,

Melamed uses the concept of a "minimum content-bearing unit," the smallest unit of text needed to select an appropriate translation.

Non-compositional phrases are identified using a simple intuition: a joint translation model that contains only word-to-word statistics will be less efficient in assigning probabilities than a joint model that is aware of phrases with non-compositional translations. Melamed makes this notion concrete by measuring the mutual information (MI) of the English vocabulary E with respect to the French vocabulary F under his joint model:

$$I(E;F) = \sum_{e \in E} \sum_{f \in F} \Pr(e,f) \log \frac{\Pr(e,f)}{\Pr(e)\Pr(f)} \tag{2.21}$$

A model with non-compositional phrasal units should receive a higher MI score, as the presence of a phrase means its component words need no longer account for their phrasal interpretation, reducing uncertainty. One can formulate the phrase finding task as the search for phrases that increase $I(E;F)$ when added to the lexicon. A naïve, greedy approach would propose a phrase, induce a model where that phrase is always treated as a single token, and then compare the MI scores of the models built with and without the phrase. If the phrase increases MI, it is added to the lexicon, and the process repeats.

With this framework in mind, Melamed builds up a series of approximations and heuristics to allow intelligent batch additions of phrases. He develops an estimate of how much a proposed phrase will increase MI, and then adds phrases with high estimates to his lexicon, and induces a new word-to-word model using competitive linking (§ 2.1.2). A credit-assignment equation allows him to keep only phrases that contribute to an observed MI improvement. To get phrases with more than two words, he uses an iterative process that begins by combining pairs of words, but which can also combine the phrases added in previous iterations. Finally he has the system alternate between which language (E or F) is being enhanced with phrases, to create phrasal entries in both lexicons. This allows many-to-many alignment. For the sake of tractability, only contiguous phrases are proposed, though he also considers pairs of words that are separated by one or two function words as potential phrases.

This system incorporates the notion of phrases into model construction. Furthermore, the statistical model uses a phrasal table for both languages. However, the phrases to be considered are selected heuristically. Model estimation uses a Viterbi EM process, which is iterative, but only considers the single most likely alignment at any point. Because sentences are re-segmented phrasally before alignment begins, any phrasal alignments natu-

rally obey transitive closure. However, this introduces a new restriction, where phrases cannot compete with their composite words, they are always treated as a unit.

2.4.5 Joint Phrasal Translation Model

Marcu and Wong (2002) construct a joint phrasal translation model (JPTM) with all of the statistical rigor of the IBM conditional, word-based models (§ 2.1.1). Like the IBM models, the JPTM is learned unsupervised from bitext using EM. Though theoretically very attractive, this model requires several approximations due to its ambitious alignment space and storage requirements.

The joint model is designed according to a straight-forward generative process. First, a bag of cepts C is generated. Each $c_i \in C$ corresponds to a bilingual phrase pair, $c_i = (\bar{e}_i, \bar{f}_i)$. The phrases are independently permuted in each language to create two sequences of phrases. This is similar to the word-to-word joint model for competitive linking (§ 2.1.2). Note that phrases are permuted, not words, meaning phrases must be contiguous. Also note that the use of cepts guarantees transitive closure. For their Model 1, they assume that the number of cepts, as well as the phrase permutation, are drawn from uniform distributions. A joint translation distribution $\Pr(\bar{e}_i, \bar{f}_i)$ determines which phrase pairs are selected. Given lexicon of phrase pairs and a predicate $L(\vec{e}, \vec{f}, C)$ that determines if a bag of cepts C can be bilingually permuted to create $\vec{e}$ and $\vec{f}$, the probability of a sentence pair is:

$$\Pr(\vec{e}, \vec{f}) \propto \sum_{\{C|L(\vec{e},\vec{f},C)\}} \left[\prod_{c_i \in C} \Pr(\bar{e}_i, \bar{f}_i) \right] \tag{2.22}$$

With an unconstrained lexicon, this calculation will consider every possible segmentation of $\vec{e}$ and $\vec{f}$, and every possible alignment between those segments. It is completely enumerating the contiguous phrasal alignment space. Their Model 2 adds an IBM-2-style absolute position distortion model to Model 1.

Reality asserts itself as they begin training the model. A translation table that considers all observed phrase pairs in a corpus would be enormous, far too big to fit into memory. They restrict themselves to phrase pairs for which the component monolingual phrases have been observed at least five times throughout the corpus. Furthermore, no monolingual phrase is more than six tokens long. We will refer to these restrictions on possible phrases as **lexicon constraints**. Marcu and Wong provide a Stirling-number-based approximation for expectation when all phrase pairs are weighted uniformly, allowing the first E-step of EM to be performed quickly. However, after the first M-step, model parameters are non-uniform,

and during subsequent E-steps, a hill-climbing alignment search is used to sample alignment space around a high-probability point for each sentence pair. This heuristic search is fast enough to allow them to train on 100K sentence pairs.

Birch et al. (2006) propose a constraint to allow training on larger corpora by restricting the phrasal alignment space according to agreement with a high-precision alignment. For a phrase pair $(\bar{e}_i, \bar{f}_i)$ to be considered, any high-precision link beginning in the pair must also end within the pair; links cannot straddle phrases. This has the effect of both reducing the search space for their hill-climber, and reducing the number phrase-pairs considered within their lexicon. The high precision alignments are provided by GIZA++ intersection. We refer to this constrained joint model as the C-JPTM. These constraints, along with a faster alignment sampling algorithm, allow them to train on 700K pairs.

2.4.6 Phrasal Alignment Review

We have reviewed several approaches for phrasal bitext analysis. Refined heuristic combination, as the most widely available method, remains the defacto standard. There are few comparison points between these systems in terms of alignment or translation quality. However, in summarizing these methods, we have attempted to call out some qualities that are important to phrasal approaches. In this section, we enumerate those characteristics explicitly, and conduct a qualitative system comparison.

P **Truly phrasal**: One multi-token phrase can be completely linked to another.

T **Transitive Closure**: Any multi-token phrases linked in part must be completely linked; if f_k and f_l are both linked to e_i, then any e_j linked to f_k must also be linked to f_l.

D **Discontiguous**: Monolingual phrases are **not** constrained to be contiguous sequences.

L **Phrases present in learning**: The translation model construction process benefits somehow from awareness of phrases.

M **Phrasal model**: The translation model explicitly contains multi-word phrases for both language pairs.

B **Bilingual phrase selection**: Phrase usage decisions are informed by the availability of translations for the phrase.

S **Perfect search**: The phrasal alignment space can be completely enumerated.

Table 2.2 compares the reviewed systems according to these qualities. A $\surd$ indicates that a quality is present. Examining the table, we see that only the syntactic methods allow

Table 2.2: A qualitative comparison of several phrasal word alignment systems.

System \ Quality	P	T	D	L	M	B	S
GIZA++ Union	√		√			√	
GIZA++ Refined		√	√			√	
Matrix Factorization	√	√	√			√	
Recursive (Wu, 1997)	√	√			√	√	√
Recursive (Vilar and Vidal, 2005)	√	√		√		√	√
Tree-to-string	√	√		√		√	√
MI-based Model Re-estimation	√	√		√	√		
Joint Phrasal Model	√	√		√	√	√	

perfect searches, and only contiguous methods attempt phrasal learning or phrasal tables. In our work, we will not try to overcome either of these rules of thumb. Instead we will fill in the logical bottom row of the table, by using a syntactic aligner to conduct a complete search with the sound statistical methods from the joint phrasal model (§ 4).

2.5 Discriminative Alignment

Recently, a number of discriminative word alignment methods have been proposed. These methods allow arbitrary, overlapping feature representations, but require word-aligned training data. The use of labeled data is a large departure from the alignment methods we have described up to this point, which require only sentence-aligned bitext. The use of labeled data is often justified by the fact that the popular unsupervised IBM models used in GIZA++ (§ 2.1.1) require a small labeled development set, in order to tune hyperparameters. Furthermore, most discriminative methods leverage the strong correlations found in large, sentence-aligned bitexts to construct a small set of rich features, allowing them to learn discriminatively from very small labeled sets.

Generative models are traditionally difficult to extend with new features because the dependencies between features need to be considered explicitly; for example, Brown et al. (1993) had to create an entirely different generative story from that of IBM Model 2 to incorporate fertility into Model 3. In a discriminative approach, the designer is free to include whatever features may prove useful, such as bilingual dictionaries, morphological features, or parts of speech, without concern for how features interact.

Most discriminative methods are guided by a feature vector $\Psi(x, y)$, which describes a complete candidate alignment y, built for an input sentence pair x. The best alignment

under the model is found using some manner of search:

$$\text{struct}(x; \vec{w}) = \text{argmax}_{y \in \mathcal{Y}} [\vec{w} \cdot \Psi(x, y)] \tag{2.23}$$

where $\mathcal{Y}$ is the set of all possible structures, and $\vec{w}$ is a vector that assigns a weight to each feature in $\Psi(x, y)$. The goal of discriminative training is to learn a weight vector $\vec{w}$ so that $\text{struct}(x; \vec{w})$ optimizes some notion of accuracy on a set of labeled training instances $\{(x_i, y_i)\}_{i=1}^{N}$. Broadly, the learning objective is to have the correct alignments y receive more weight than their incorrect competitors $\bar{y}$:

$$\forall i, \forall \bar{y}_i \in \mathcal{Y}_i - \{y_i\} : \vec{w} \cdot \Psi(x_i, y_i) > \vec{w} \cdot \Psi(x_i, \bar{y}_i) \tag{2.24}$$

This notion can be fine-tuned with a loss function $\Delta(y, \bar{y})$ that calculates the structured distance between an incorrect candidate $\bar{y}$ and the correct y. There are four primary design choices to be made in the construction of a discriminative system:

1. A search algorithm used to find $\text{struct}(x; \vec{w}) = \text{argmax}_{y \in \mathcal{Y}} [\vec{w} \cdot \Psi(x, y)]$
2. A feature representation of an alignment $\Psi(x, y)$
3. A discriminative training method used to find a good $\vec{w}$
4. A loss function $\Delta(y, \bar{y})$ that determines how closely $\bar{y}$ matches y.

Since we employ discriminative training in this thesis, this section reviews a number of discriminative alignment approaches. We pay close attention to the two systems that have most influenced our work: Moore's perceptron aligner (2005b) and Taskar et al.'s discriminative matching system (2005). Following the review of discriminative aligners, we briefly review our chosen discriminative training method, SVM Struct (Tsochantaridis et al., 2004).

2.5.1 Perceptron Alignment

Perhaps the most straight-forward discriminative word-alignment system, the perceptron aligner (Moore, 2005b) uses an averaged perceptron (Collins, 2002) to weight features in an alignment beam search. It is a natural extension of the more complex competitive-linking approaches (Cherry and Lin, 2003; Moore, 2005a), which are similar, despite employing non-discriminative feature-weighting schemes.

The search to find $\bar{y} = \text{struct}(x; \vec{w})$ is handled by a beam search. Links are added one-by-one to an initially empty alignment hypothesis. Each hypothesis is assessed as a complete alignment, and the highest scoring alignment found during the search is returned as $\bar{y}$. The main advantage of this heuristic search is that each alignment hypothesis is

fully aware of all the links it contains, which allows for interesting features regarding link interaction. However, the beam search is not guaranteed to find the optimal solution to the argmax in (2.23).

The feature representation $\Psi(x, y)$ broadly covers the main features of the IBM translation models: lexical translation affinity, distortion, and fertility. Like in competitive linking (§ 2.1.2), bilingual lexical affinity is measured by a score assigned to word pairs (e, f). This score is provided by conditional link probability mined from a large sentence-aligned corpus, which has been automatically word-aligned by an earlier version of the perceptron aligner. The value of this feature for a complete alignment y is the sum of scores for all links in y. A distortion feature measures non-monotonicity by counting the number of times links cross backwards over themselves. A fully monotonic alignment displays no concept re-ordering. Fertility is monitored by features counting the number of one-to-many links, with another feature counting the number of words that are left unaligned. Many-to-many links are not allowed. New features are added in (Moore et al., 2006), including lexical features that directly indicate if specific word-pairs are linked in the current hypothesis.

The weight vector $\vec{w}$ is learned using an averaged perceptron. This is an online learning method that iterates through examples in the 200-sentence-pair, word-aligned training set $\{(x_i, y_i)\}_{i=1}^{200}$. For each training example, the learner runs the alignment search with the current $\vec{w}$, and the search returns $\bar{y}_i = \text{struct}(x_i; \vec{w})$. After each search, $\vec{w}$ is updated to prefer the correct y_i over an incorrect $\bar{y}$:

$$\vec{w} \leftarrow \vec{w} + \Psi(x_i, y_i) - \Psi(x_i, \bar{y}_i) \tag{2.25}$$

The algorithm cycles through the training set until $\vec{w}$ converges. To increase robustness, the algorithm ultimately returns the average of all weight vectors seen throughout training. This learning method provides no mechanism to specify a loss function $\Delta(y, \bar{y})$, but a 0-1 loss that assigns an equal penalty to all $\bar{y} \neq y$ is employed implicitly.

2.5.2 Discriminative Matching

The discriminative matching system (Taskar et al., 2005) replaces Moore's beam search with the weighted bipartite matching algorithm (West, 2001), and replaces perceptron training with a large margin structured classifier that allows a structured loss function $\Delta(y, \bar{y})$. This more mathematical approach to discriminative alignment leads to both strengths and weaknesses with respect to the perceptron aligner, which are highlighted below.

The search to find $\bar{y} = \text{struct}(x; \vec{w})$ is handled by the bipartite matching algorithm.

This algorithm inputs a bipartite graph that has a node for each token in the sentence pair, with edges connecting every French node to every English node. Values $v(e_j, f_k)$ on edges indicate the affinity for linking the corresponding token pair (e_j, f_k). The bipartite matching algorithm finds a one-to-one matching that maximizes the sum of edge values. Edge values are set according to the features $\psi(e_j, f_k; x_i)$ of the edge: $v(e_j, f_k) = \vec{w} \cdot \psi(e_j, f_k; x_i)$. The matching algorithm is a complete search; that is, $\text{struct}(x; \vec{w})$ always returns the highest scoring $\bar{y}$ for the current $\vec{w}$. However, this search does come with some representational restrictions: it can only produce one-to-one alignments, and potential links (e_j, f_k) must be scored before alignment begins, eliminating the possibility of monotonicity or fertility features. Both of these shortcomings are addressed in (Lacoste-Julien et al., 2006), which makes use of the linear programming formulation of bipartite matching (Papadimitriou and Steiglitz, 1998, pg. 248) to extend the search and learning algorithms at some computational cost. Note that Moore's beam search suffers from neither of these restrictions, but is also not guaranteed to return an optimal alignment.

The feature vector $\Psi(x, y)$ is the sum of the features of y's component links: $\Psi(x, y) = \sum_{(e_j, f_k) \in y} \psi(e_j, f_k; x)$. The features ψ on individual links include a lexical translation score that assess the word-pair (e_j, f_k) based on their bilingual correlation, a distortion penalty given by the absolute difference between the indices j and k, and a number of morphological and lexical features. A final feature indicates if a potential link was included in the alignments output by IBM Model 4 as trained by GIZA++, which significantly improves their results. More details on a similar feature set are provided in § 5.4.4, where we re-implement their system. Later in (Lacoste-Julien et al., 2006), they add new features that are not tied to specific edges, and which help assess fertility and monotonicity.

The objective of their learning method is to find a $\vec{w}$ that assigns the correct alignment y a higher score than all incorrect alignments $\bar{y}$. A large margin objective that accomplishes this goal while allowing for some misclassifications is one that find a $\vec{w}$ that minimizes the impact of the worst alignments:

$$\min_{||\vec{w}|| < \gamma} \sum_{i=0}^{N} \max_{\bar{y}_i \in \mathcal{Y}_i} [\vec{w} \cdot \Psi(x_i, \bar{y}_i) + \Delta(y_i, \bar{y}_i) - \vec{w} \cdot \Psi(x_i, y_i)] \tag{2.26}$$

Note that the worst alignments $\bar{y}$ have high scores $\vec{w} \cdot \Psi(x, \bar{y})$, and are distant from the correct answer y according to the structured loss $\Delta(y, \bar{y})$. γ is a regularization parameter on the size of $\vec{w}$. For $\Delta(y, \bar{y})$, they use a Hamming loss function that counts missed and incorrect links in $\bar{y}$, because this loss decomposes nicely over links. The $\vec{w}$ that minimizes (2.26) can be found by applying the extra-gradient method (Taskar et al., 2005) to the linear

programming formulation of weighted bipartite matching. The application of this learning method depends on specific characteristics of the search algorithm that allow the learning objective to be factored. This tightly couples the learning method to the search. In contrast, the perceptron in (Moore, 2005b) requires only the output of the search, allowing the designer to freely switch search algorithms.

2.5.3 Other Discriminative Solutions

There have been a host of other proposals for discriminative solutions to word alignment in the past two years. We outline a number of them here, beginning with probabilistic approaches, and then moving on to others that are more difficult to categorize.

Two early discriminative aligners are powered by maximum entropy learning (Liu et al., 2005; Ittycheriah and Roukos, 2005). In these cases, the normalizing term required by maximum entropy is approximated by sampling high-scoring alignments, making the system more similar to a K-best re-ranker. Liu et al. use a greedy search driven by the IBM Model 3 score of the candidate alignment and several linguistic features. Meanwhile, Ittycheriah and Roukus use a number of probabilistic models inspired by the alignment HMM, along with features specific to handling unsegmented Arabic, in a beam search. Refining these ideas with a more principled approach, Blunsom and Cohn (2006) implemented the alignment HMM (§ 2.1.1) as a Conditional Random Field (CRF). This framework uses the Viterbi algorithm for search, the forward algorithm to calculate the exact normalization term, and forward-backward to calculate marginals that are necessary during training. They employ a feature set similar to those of (Moore, 2005b; Taskar et al., 2005), but include first-order features made available by the HMM's algorithmic structure. As probabilistic methods, none of these solutions make use of a structured loss function Δ.

Fraser and Marcu (2006b) learn their weights $\vec{w}$ with the Minimum Error Rate Training (MERT) system commonly employed for machine translation decoding (Och, 2003). MERT requires a search that returns a K-best list of alignment candidates. Their learning framework allows them to use a weighted F-measure, which is known to correlate well with translation quality, for their $\Delta(y, \bar{y})$. They use an interesting feature set, breaking IBM Model 4 into a number of component models to be re-weighted. Their search is an improved version of GIZA++'s local search, augmented with a priority queue and transposition table. Finally, they iterate multiple times between feature construction (where features are built by taking statistics from an automatically word-aligned corpus) and discriminative training, hoping that improvements in one will improve the other.

Ayan et al. (2005) do not use a structured learning method at all to do discriminative alignment. Instead, they break the problem down into n^2 classification problems, where each link is independently classified as present or absent. A multi-layer neural network is learned on these independent instances. What makes their system work is their use of structured features. Their method is intended only for alignment combination, so entire noisy alignments are included as input for a given sentence pair. Features for a particular link decision indicate which input alignments include the link, and provide information regarding the link's neighborhood in the input alignments.

2.5.4 SVM for Structured Output

To discriminatively train our alignment systems, we adopt SVM Struct,[4] the Support Vector Machine (SVM) for Structured Output (Tsochantaridis et al., 2004). We have selected this system because we feel it strikes a strong middle ground between the averaged perceptron in § 2.5.1 and the large margin approach in § 2.5.2. Like an averaged perceptron, the search and learning algorithms are decoupled; we are free to select any search and feature representation so long as we can compute the argmax in (2.23). Like discriminative matching, SVM Struct incorporates structured loss and a large margin objective. We summarize the learning mechanism briefly in this section, but readers should refer to (Tsochantaridis et al., 2004) for more details.

As in all structured learning methods, our ultimate objective is to find a weight vector $\vec{w}$ that separates the correct y from all incorrect $\bar{y}$. SVM Struct employs a soft-margin learning objective for $\vec{w}$ that is similar to (2.26), which can be formulated as a quadratic program:

$$\min_{\vec{w},\xi} \quad \frac{1}{2}||\vec{w}||^2 + \frac{\gamma}{N}\sum_{i=1}^{n}\xi_i \tag{2.27}$$

$$\text{s.t.} \quad \forall i : \xi_i \geq 0 \tag{2.28}$$

$$\forall i : \forall \bar{y}_i \in \mathcal{Y}_i - \{y_i\} : [\vec{w} \cdot \Psi(x_i, y_i) - \vec{w} \cdot \Psi(x_i, \bar{y}_i) \geq \Delta(y_i, \bar{y}_i) - \xi_i]$$

The constraints in (2.28) specify that the correct y must score higher than every incorrect $\bar{y}$ for all training instances. Furthermore, the size of the required margin is determined by the structured loss $\Delta(y, \bar{y})$.[5] A slack variable ξ_i is introduced for each training example, to allow the learner to build incorrect alignments at a penalty. The quadratic optimizer attempts to minimize this penalty, along with the size of $\vec{w}$, under these constraints. The trade-off

[4] At http://svmlight.joachims.org/svm_struct.html

[5] Though Tsochantaridis et al. (2004) provide two ways to incorporate loss into the SVM objective, we use margin re-scaling. We find it to be more modular than its alternative, slack re-scaling.

between training accuracy ($\sum \xi_i$) and generalization to unseen data ($||\vec{w}||$) is determined by the regularization parameter γ.

The quadratic program in (2.27) cannot be solved directly, because (2.28) provides one constraint for every possible structure for every training example. Enumerating these constraints explicitly is infeasible. Faced with the same problem, Taskar et al. (2005) applied the extra-gradient method to a linear programming formulation of their search. We wish to use searches that do not necessarily have such a formulation. Fortunately, SVM Struct employs an online training method that requires only a solvable argmax, and is agnostic regarding how the search is performed.

In reality, only a subset of the constraints on $\vec{w}$ are necessary to achieve the objective in (2.28). Re-organizing (2.28) produces:

$$\forall i : \forall \bar{y}_i \in \mathcal{Y}_i - \{y_i\} : [\xi_i \geq \vec{w} \cdot \Psi(x_i, \bar{y}_i) + \Delta(y_i, \bar{y}_i) - \vec{w} \cdot \Psi(x_i, y_i)] \tag{2.29}$$

which is equivalent to:

$$\forall i : \left[\xi_i \geq \max_{\bar{y}_i \in \mathcal{Y}_i - \{y_i\}} \text{cost}_i(\bar{y}_i; \vec{w})\right] \tag{2.30}$$

where cost_i is defined as:

$$\text{cost}_i(\bar{y}_i; \vec{w}) = \vec{w} \cdot \Psi(x_i, \bar{y}_i) + \Delta(y_i, \bar{y}_i) - \vec{w} \cdot \Psi(x_i, y_i)$$

Provided that the max cost structure can be found in polynomial time, we have all the components needed for a **constraint generation** approach to this optimization problem. We can guarantee that the max cost search is feasible by choosing a decomposable loss function $\Delta(y, \bar{y})$, such as Hamming distance, and applying our original structured search to a loss-augmented formulation of the problem.[6]

Constraint generation places an outer loop around an optimizer that minimizes (2.27) repeatedly for a growing set of constraints. It begins by minimizing (2.27) with an empty constraint set in place of (2.28). This provides values for $\vec{w}$ and $\vec{\xi}$. The max cost structure

$$\bar{y} = \text{argmax}_{\hat{y}_i \in \mathcal{Y}_i - \{y_i\}} \text{cost}_i(\hat{y}_i; \vec{w})$$

is found for the first training pair (x_i, y_i) with the current $\vec{w}$. If $\text{cost}_i(\bar{y}; \vec{w}) > \xi_i$, then this represents a violation of the complete constraints in (2.30),[7] and a new constraint of the form $\xi_i \geq \text{cost}_i(\bar{y}; \vec{w})$ is added to the constraint set. The algorithm then iterates: the optimizer

[6] With a K-best list formulation of the structure search, we can easily apply any loss function, but we are trying to make as few assumptions about the search as possible.

[7] The test $\text{cost}_i(\bar{y}; \vec{w}) > \xi_i$ is usually approximated as $\text{cost}_i(\bar{y}; \vec{w}) > \xi_i + \epsilon$ for a small constant ϵ.

minimizes (2.27) again with the new constraint set, and solves the max cost problem for $i = i + 1$ with the new $\vec{w}$, growing the constraint set if necessary. Note that the constraints on ξ change with $\vec{w}$, as cost is a function of $\vec{w}$. Once the end of the training set is reached, the learner loops back to the beginning. Learning ends when the entire training set can be processed without needing to add any constraints. It can be shown that this will occur within a polynomial number of iterations (Tsochantaridis et al., 2004), providing a $\vec{w}$ that satisfies the original soft margin objective.

2.6 Decoding Algorithms

Up until this point, our review has focused on word alignment; the reviewed material has been described in the context of the analysis of bitext. However, word alignment is seen by most to be only a knowledge acquisition task, intended to fuel the real goal, statistical machine translation. In fact, many of the models described in the past sections, such as the IBM models or GMTG, were conceived with translation as their primary purpose. Recently there have also been many proposed solutions for the decoding problem in SMT, which is the task of finding a target sentence that maximizes the score of a translation model for a given source sentence. Almost all of these decoders use GIZA++ word alignments to collect statistics for alternate translation models, tailored to their specific approach to the decoding search. In this section, we review a number of decoding techniques. We first focus on the standard phrasal SMT decoder, as our translation contributions will take the form of extensions to this system. We then briefly describe a number of syntactic decoders, which share some properties with our cohesive decoder described in § 6.

2.6.1 Phrase-based Statistical Machine Translation

Phrase-based statistical machine translation (Koehn et al., 2003) is the current dominant decoding framework. The idea is to build a phrase-based translation model $\Pr(\bar{f}|\bar{e})$ from a word-aligned bitext, and to decode with multi-word, contiguous phrases as the base unit of translation. These phrasal units provide several advantages over word-based decoders:

1. Context-specific or non-compositional translations are handled naturally.
2. Local re-ordering phenomena for frequent cases are handled by memorizing movement within phrases.
3. Insertion and deletion are handled by linking smaller phrases to larger phrases. No explicit modeling of insertion or deletion is required.

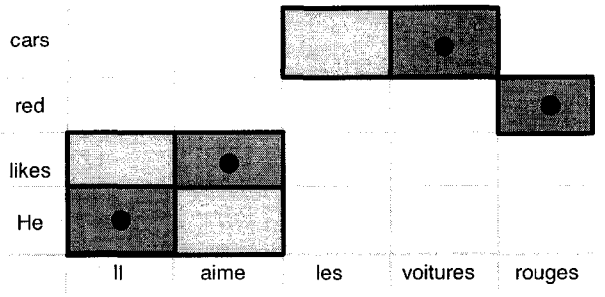

Figure 2.11: A number of phrases that could be extracted from a fixed alignment.

The translation model $\Pr(\bar{f}|\bar{e})$ is constructed using a heuristic phrase extraction process, which we will refer to as the **surface heuristic**. First, all consistent bilingual phrase pairs $(\bar{e}_i, \bar{f}_j)$ are extracted from each word-aligned sentence pair in a large bitext. A phrase pair $(\bar{e}_i, \bar{f}_j)$ is consistent with a word alignment A if:

- There is at least one link $(e_k, f_l) \in A$ such that $e_k \in \bar{e}_i$ and $f_l \in \bar{f}_j$ and
- If there is any link $(e_k, f_l) \in A$ such that $e_k \in \bar{e}_i$, then $f_l \in \bar{f}_j$. Similarly, if $f_l \in \bar{f}_j$ then $e_k \in \bar{e}_i$.

The first requirement asserts that there must be at least one word-to-word link connecting the phrase pair. The second asserts that any link beginning in the pair must also end in the pair. Unlinked tokens can freely attach onto any neighboring phrases. For example, Figure 2.11 shows six of the eleven extractable phrase pairs from the displayed alignment, using boxes to represent phrases. Four are word-to-word correspondences, the sort of statistics that would be tracked by IBM-1, but we also observe the pairs ("les voitures", "cars") and ("he likes", "il aime"). Larger phrases are possible, including one that simply memorizes the entire sentence pair. Each extracted phrase pair is counted once, and counts $count(\bar{e}_i, \bar{f}_j)$ are compiled throughout the corpus. Then a conditional translation model over phrases is constructed using:

$$\Pr(\bar{f}_j|\bar{e}_i) = \frac{count(\bar{e}_i, \bar{f}_j)}{\sum_{\bar{f}_k} count(\bar{e}_i, \bar{f}_k)} \tag{2.31}$$

At decoding time, if we are translating the French $\vec{f}$ into English, first all phrase pairs $(\bar{e}_i, \bar{f}_j)$ such that $\bar{f}_j \in \vec{f}$ are loaded into memory. Substrings of the French sentence are consumed as the English sentence is constructed from left to right using the English halves of the corresponding phrase-pairs. The process stops when each French token has been used exactly once. The decoder generally takes the form of a beam search, which maintains several priority queues of incomplete translation hypotheses, which are explored in best-first

order. Each hypothesis tracks the current partial English translation, and the list of French tokens that have been covered by this translation. Both partial and complete hypotheses are scored according to a generalization of (2.1) on page 10.

Recall that the original motivation of statistical machine translation was to return a translation that maximized the product of translation and language models:

$$\begin{aligned} translation(\vec{f}) &= \text{argmax}_{\vec{e}} \Pr(\vec{e}|\vec{f}) \\ &= \text{argmax}_{\vec{e}} \Pr(\vec{f}|\vec{e}) \Pr(\vec{e}) \end{aligned}$$

The phrase-based decoder described above provides an alternate, phrasal formulation for $\Pr(\vec{f}|\vec{e})$. Because the decoder builds the English hypothesis from the left-to-right, one can easily apply an N-gram language model $\Pr(\vec{e})$. It was pointed out in (Och and Ney, 2002) that the noisy-channel model could be generalized to a log-linear framework, where:

$$translation(\vec{f}) = \text{argmax}_{\vec{e}} \left[\sum_{k=1}^{K} \lambda_k h_k(\vec{e}, \vec{f}) \right]$$

and h_1^K is understood to be a vector of feature functions, and λ_1^K provides weights on those features. We can recover the noisy channel model in this framework with $K = 2$, $h_1 = \log \Pr(\vec{f}|\vec{e})$, $h_2 = \log \Pr(\vec{e})$ and $\lambda_1 = \lambda_2 = 1$. Och and Ney (2002) demonstrated that the λ weights could be set to create more accurate translations using a discriminative training method, such as Maximum Entropy. Later, Och (2003) developed a method known as Minimum Error Rate Training, or MERT, to set the weights in order to maximize BLEU score, the primary translation evaluation metric.

In addition to allowing discriminative weighting, the log linear scoring model for SMT allows the inclusion of new feature functions. The standard feature package as defined by the SMT Workshop baseline system (Koehn et al., 2005) is as follows:

Language model We will generally assume a single 3-gram language model. Many competition systems include multiple language models, and include higher order language models, such as 5-grams, or vary the domain of language model training data.

Translation models Modern decoders generally employ at least four translation models. Two are phrasal models as described in (2.31). These characterize $\Pr(\bar{f}|\bar{e})$ and $\Pr(\bar{e}|\bar{f})$ respectively based on the observed frequencies of consistent phrases in the training bitext. Two other translation models, $\Pr_{lex}(\bar{f}|\bar{e})$ and $\Pr_{lex}(\bar{e}|\bar{f})$ provide lexical weighting. These are designed to indicate if the phrase-pair in question has a strong word-to-word explanation. Two formulations for lexical weighting are common, one that considers observed word alignments (Koehn et al., 2003), and another that uses unnormalized IBM-1 probability (Vogel et al., 2003).

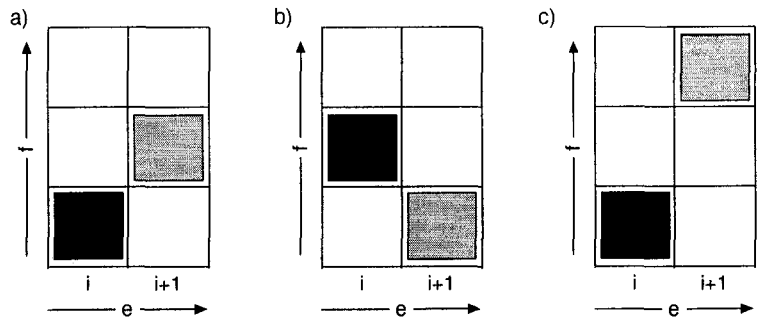

Figure 2.12: Three possible orientations of the grey block $\bar{e}_{i+1}$: (a) straight, (b) inverted and (c) disjoint.

Distortion penalty When adding a phrase $\bar{e}_i$ to the current hypothesis, one can determine distortion according to the distance of the source phrase $\bar{f}_i$ from the source of the previously translated phrase $\bar{f}_{i-1}$. The sum of these HMM-like absolute positional differences provides a value that grows with the amount of re-ordering.

Word and Phrase counts Finally, two features count the number of words used in the English translation, and the number of phrases used to create this translation. The former can be used to penalize longer translations, or to give them a boost, off-setting the language model's default preference for shorter translations. The latter can be used to express a preference for translations that involve fewer phrases, and therefore, leave less work for distortion and language models.

Many other possible features can be employed. A large number of syntactic features were explored in an K-best re-ranking setting in (Och et al., 2004), but most of these were not deemed to be sufficiently effective to offset the computational cost. This work also indicated that the MERT algorithm is prone to over-fitting when given too many features. Even so, all of the decoders that we describe in this section also make use of log-linear feature models, tuned with MERT.

Another potentially useful feature is a more complex, data-driven distortion model, described in (Tillman, 2004; Koehn et al., 2005). This model maintains a probability distribution over the possible orientations a phrase-pair can exhibit with respect to its previous target phrase. These orientations are usually straight, inverted or disjoint, as shown in Figure 2.12. Parameters for the distribution are determined by counting the orientations observed in an aligned bitext. The distribution is conditioned bilexically, on both the target phrase and its source phrase. The contributions of these distributions are re-weighted by minimum error

rate training during tuning. Lexicalized re-ordering models are already implemented in the Moses decoder (Koehn et al., 2007), and are considered important enough to include in competition-grade translation systems (Koehn et al., 2005).

Regardless of the features used in scoring hypotheses, phrase-based decoders build their English translations from left to right, consuming and translating French phrases with each addition to the growing hypothesis. The order in which French phrases can be consumed determines the re-orderings allowed by the decoder. Standard phrase-based decoders (Koehn et al., 2003) allow limited string-based re-ordering, where the translation is only allowed to perform fixed deviations from the default monotone order, as determined by a **distortion limit**. This limit dictates how far the decoder can progress, moving from left to right through f_1^n, after it has skipped over some source word f_i, leaving it to be translated later. With a distortion limit d, the decoder can translate up to f_{i+d} before it has to double back and cover f_i. Completely monotone dynamic programming methods, which allow no re-ordering at all, have been shown to work for close language pairs, like English-French. These methods rely on having any necessary re-orderings captured inside their phrases (Zens and Ney, 2004). The ITG re-orderings can also be applied to SMT decoding, resulting either in a search that behaves like a parser (Wu, 1996; Huang et al., 2005), or a re-ordering constraint on standard left-to-right beam decoding (Zens et al., 2004). Recently, the notion of phrases has been extended to hierarchical phrases (Chiang, 2005). In this case, phrases can contain place holders for other phrases. This has the interesting result of allowing context-specific movement. This approach introduces the theory and structure of synchronous CFGs to the decoding process, with no linguistic syntax present during alignment or model construction.

2.6.2 Syntactic Preprocessing

One straight-forward method to reduce the impact of concept movement on phrase-based SMT is to re-order the input sentence. The hope is that in doing so, one might create a source sentence with word order that is already similar to the target word order. IBM's original statistical decoder employed some pattern-based preprocessing, where certain English constructions were re-ordered deterministically to more closely resemble French (Berger et al., 1994). We briefly discuss two approaches with similar goals, but which re-order the source sentence by manipulating its parse tree.

Xia and McCord (2004) apply automatically-derived rules to a source parse tree in order to re-order the source sentence. The rules are learned from word-aligned bitext, by inferring syntactic transformations based on movement observed in the word-alignment. The rules

are applied before both training and testing, but otherwise their phrasal SMT infrastructure remains the same. They found that in order to get meaningful improvements, they had to disable the decoder's internal movement capabilities; that is, the only movement that occurs during translation occurs during syntactic pre-processing.

Collins et al. (2005) present another pre-processing approach to adding syntax, but they target a specific language-pair, German-English. Their transformations are not derived from data; instead, they apply a small number of linguistically-motivated clause re-orderings to German input, in order to make the German word-order more like English. They were able to show substantial improvements in translation quality, but the work is highly dependent on the German-English language-pair.

The above methods adopt a sort of pipeline approach to the use of syntax in SMT. Syntax is applied first, then translation happens. This approach uses only a one-best syntactic re-ordering, which is not informed by the translation model in any way. The methods reviewed in the remainder of this section adopt syntax as a primary assumption, incorporated fully into the translation model, and optimized jointly during translation.

2.6.3 Dependency-based Statistical Machine Translation

Some approaches use techniques that are similar to those of phrasal SMT, but with a dependency tree in place of a flat string for input. These methods traditionally translate English into other languages. Lin (2004) presents a method that learns from a word-aligned English-French bitext, with parse trees provided for the English half. The alignment is used to project a dependency tree onto the French half of the training corpus. A translation model is learned to track how paths in the English dependency tree becomes paths in the French tree. Decoding begins by parsing the English input sentence. Then the English halves of the learned path pairs are used to build a highest-weight path covering of the input dependency tree, which in turn specifies a French translation. Re-ordering decisions, when they are not specified by the selected paths, are resolved deterministically with a heuristic.

Quirk et al. (2005) follow a similar approach with some marked differences. They extend the notion of path pairs to treelet pairs, where a treelet is any connected portion of a tree. They also learn a re-ordering model to assign probabilities to permutations of a parent's children, as in (Yamada and Knight, 2001), however they use a decision tree in place of table look-up. At decoding time, all treelet coverings of an input tree are attempted using a decoder with a parser-like structure. The process is similar to the tree-substitution grammars described in (Eisner, 2003). A log-linear model incorporating an N-gram language

model, the treelet translation model, the decision-tree re-ordering model, and a number of other features borrowed from phrasal SMT, is used to rank candidate translations.

2.6.4 Tree Transducers

The string-to-tree decoder described in (Yamada and Knight, 2002) transforms a French input string through a parser-like process into an English tree. It is guided by the tree-to-string translation model in (Yamada and Knight, 2001). However, the child re-ordering, node insertion, and leaf-translation operations used by that model are only a subset of the operations available to a broad class of machines known as tree transducers (Graehl and Knight, 2004). Using this more general formalism allows the decoder to have access to operations that are substantially more powerful than child re-ordering, as rules can now operate on larger tree fragments than local trees.

The translation rules used in these transducers are learned from aligned bitext, where the English side has been parsed (Galley et al., 2004). The manner in which the alignments are interpreted is quite interesting. The alignments are viewed as a byproduct of a generative process that creates the English tree from the observed French string using tree transducer operations. If during a single step in the transducer derivation, a French word is deleted at the same time an English word is created, then those two words are linked in the alignment. Using this interpretation and a fixed word alignment, one can derive the operations that must have been used to generate the English from the French. Like in phrasal SMT training, alignments serve to limit the number of possible derivations, but several compatible derivations may be possible for a given alignment. These rules can affect several French tokens at the same time, giving this approach many of the advantages of phrasal SMT.

A tree transducer decoder has a parser-like structure. As in the original string-to-tree system, a French source string is transformed into a target English tree. Like in most successful phrasal decoders, it is possible to incorporate a number of helpful features into the translation scoring system using a log-linear model. It has been shown that decoding can be made more efficient by binarizing the transducer rules using intuitions from ITG parsing (Zhang et al., 2006). This increases translation quality by allowing language model probabilities to be integrated into the search. It has also been shown that one can run the transduction process in reverse; Huang et al. (2006) use the same transducer-rule extraction infrastructure to learn rules for syntax-directed decoding. In this case, the source sentence is parsed, and then transformed by a tree transducer into a target language string. This work is similar to the dependency-based approach (Quirk et al., 2005), but it uses constituency trees

in place of dependency trees, and uses probabilistic transducer rules in place of specialized decision-tree order models.

Chapter 3

Evaluation Metrics

In the coming chapters, we present and then test a number of novel methods for modeling, alignment and translation, all of which take advantage of syntactic restrictions on the movement that can occur during translation. We evaluate our systems according to two main criteria. The first, alignment quality, measures a word-aligner's ability to match manually-created gold-standard alignments. The second, translation quality, measures the ability of a translation system to match a human reference translation for the same input sentence.

3.1 Alignment

Automatic word alignment is generally evaluated by comparison to gold-standard word alignments created by bilingual humans. Unfortunately, due to the nature of human translation, alignment can be an ambiguous task, where annotators do not always agree. To account for this, gold standards are often annotated with both **sure** and **possible** links (Och and Ney, 2003). The sure set S is used when multiple annotators feel confident about a particular link. The possible set P is used for other cases, where annotators are unsure or disagree. Any sure link is also possible ($S \subseteq P$). Alignment results are reported in terms of **precision**, **recall**, **F-measure** and **alignment error rate** or **AER**.

- Precision is the proportion of guessed links that are possible:

$$\text{Prec}(A, P) = \frac{|A \cap P|}{|A|}$$

- Recall is the proportion of sure links that were found:

$$\text{Rec}(A, S) = \frac{|A \cap S|}{|S|}$$

- Balanced F-measure is an unweighted combination of precision and recall:

$$\mathrm{F}(A, S, P) = \frac{2 \cdot \mathrm{Prec}(A, P) \cdot \mathrm{Rec}(A, S)}{\mathrm{Prec}(A, P) + \mathrm{Rec}(A, S)}$$

- AER is an alternate synthesis of precision and recall. Lower error rates are better:

$$\mathrm{AER}(A, S, P) = 1 - \frac{|A \cap P| + |A \cap S|}{|A| + |S|}$$

Since its introduction in (Och and Ney, 2000a), AER using sure and possible links has been the primary method used to report alignment quality.

A recent report by Fraser and Marcu (2006a) has criticized AER as an alignment-quality metric. Spurred by evidence indicating that alignment quality does not correlate with translation quality, they point out problems with AER, and propose alternatives. In particular, they show algebraically that AER does not penalize unbalanced precision and recall in the same way as balanced F-measure. This can lead AER to prefer alignments that link infrequently and thereby achieve high precision. Balanced F-measure is shown experimentally to correlate slightly better with alignment quality. Furthermore, a gold standard created according to the Blinker annotation guidelines (Melamed, 1998), using only sure links ($S = P$), is shown to provide F-measures that have far greater correlation with translation quality. Finally, they show that by weighting F-measure to prefer either precision or recall, they can further increase correlation. However, the required precision-recall tradeoff appears to vary according to the language pair, the data set, and most disappointingly, the amount of available training data.

Informed by this report, we generally report balanced F-measure in addition to AER. We also produce our own test set labeled using the Blinker guidelines and only sure links, described in § 4.4.1. However, we do still believe that there is value in reporting intrinsic alignment quality. Therefore, we do not go so far as to report weighted F-measure, because these weights appear to depend on specifics of the training data and the translation decoder. When it is relevant, we report the translation scores achieved when our alignments and models are used to produce translation models for decoding.

3.2 Translation: BLEU

As indicated above, an alternate method for evaluating word alignment is to measure the translation quality of an SMT system trained using the alignment. Of course, translation quality is also relevant to approaches that only affect end translation output, such as our

cohesive decoder (§ 6). Our translation results are reported using BLEU score (Papineni et al., 2002). BLEU score is a precision-based metric that is calculated by comparing system translations to human-created reference translations. The unit of comparison is the number of matching N-grams for several values of N.

Formally, BLEU works with modified N-gram precision values. This counts the number of N-grams in the system translation that match an N-gram from the reference, where each reference N-gram is allowed to be used at most once. We represent this modified match count with $match(ng; tr, tr')$ for a specific N-gram ng, a reference tr and, a system translation tr'. Precision for a particular value of N using a collection of K single-reference translation examples is given by:

$$\text{Prec}_N = \frac{\sum_{i=0}^{K} \sum_{\{ng | ng \in tr'_i, len(ng)=N\}} match(ng; tr_i, tr'_i)}{\sum_{i=0}^{K} \sum_{\{ng | ng \in tr'_i, len(ng)=N\}} count(ng; tr'_i)}$$

Since BLEU is precision-based, to limit the ability to game the system by reporting only short, precise translations, a multiplicative breadth penalty is applied. This penalty punishes translations that are too short by comparing the complete system translation length to the reference length. Given $c = \sum_i |tr_i|$ and $c' = \sum_i |tr'_i|$, the breadth penalty is:

$$BP = \begin{cases} \exp\left(1 - \frac{c}{c'}\right) & \textbf{if } c' \leq c \\ 1 & \textbf{otherwise} \end{cases}$$

The BLEU score itself is a geometric mean of N-gram precision values for $N = 1 \ldots 4$, modified by the breadth penalty:

$$BLEU = BP \cdot \exp\left(\sum_{N=1}^{4} \log \text{Prec}_N\right)$$

Note that BLEU scores are calculated over the entire test set; the score does not decompose nicely over individual sentence pairs. However, one can estimate a sentence's contributions to both N-gram precision and the breadth penalty for the sake of technologies like minimum error rate training (Och, 2003). We report $100 \cdot BLEU$ as our metric of translation quality. Formulations of BLEU that use multiple reference translations exist, but all of our test sets only have single reference translations.

The BLEU metric has its own set of problems that we will not begin to address in this thesis. Conventional wisdom, discussed in (Callison-Burch et al., 2006), states that BLEU is best employed in the comparison of several similar systems that are trained and tested under the same setting. In isolation, a BLEU score is not a particularly useful measurement of translation quality, but it does provide an effective measurement of improvement over a related baseline.

Chapter 4

Phrasal ITG

In this chapter, we present, discuss, and test our **phrasal ITG**, an extension to the inversion transduction grammar described in § 2.2. We have already emphasized the importance of phrasal analysis and described several competing techniques in § 2.4. Among those techniques, the joint phrasal translation model, or JPTM, appears to be the most promising; it uses the same statistical rigor that has been successful in IBM's word-to-word approaches, but replaces words with phrases as the primary unit of alignment and translation. Unfortunately, the phrasal nature of the JPTM's models introduces algorithmic complications far earlier in the modeling process than in its word-based brethren. The joint problem of segmenting text into phrases and aligning those phrases ties phrasal alignment decisions together, and this lack of independence between decisions makes finding efficient algorithms difficult. The simplest JPTM must use approximate hill-climbing and sampling methods to collect its expectations for EM, where its word-to-word equivalent, IBM-1, has an efficient $O(n^2)$ algorithm. However, independence can be re-introduced by adopting the context-free assumptions inherent in an ITG. As an ITG divides the sentence pair into nested constituents or blocks, the alignments inside a block become independent of any alignments outside the block. Therefore, at the cost of enforcing the ITG constraints for concept reordering, our phrasal ITG will gain efficient algorithms for both search and expectation, while retaining all of the modeling advantages of the JPTM.

One of the advantages of joint phrasal modeling is the ability to extract a phrasal translation lexicon directly using EM, without ever committing to a single alignment. The hope is that by inferring global distributions over phrasal alignments, rather than extracting consistent phrase pairs from each sentence pair independently, we can produce a smaller, more accurate phrase table for phrasal decoding. We refer to this table construction problem as **translation modeling**, and consider it to be distinct from, but related to word alignment.

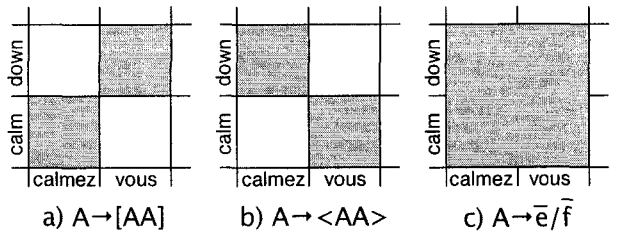

Figure 4.1: Three ways in which a phrasal ITG can analyze a multi-word span or phrase.

This chapter describes several algorithmic contributions. First and foremost, we present a phrasal ITG trained using the inside-outside algorithm. Secondary contributions include a fixed-link pruning method to scale up the phrasal ITG, and a non-compositional constraint to improve alignment quality by encouraging small, intuitive phrasal links. We begin by describing our phrasal ITG in § 4.1. We follow this in § 4.2 with a qualitative comparison to the JPTM, which will also highlight and address weaknesses introduced by the ITG formalism. In § 4.3, we describe how our phrasal ITG can be used both as a phrasal translation model and a phrasal word alignment technique. § 4.4 describes our experiments testing the new model, and discusses our results.

4.1 Adding Phrases

This section uses the ITG formalism to implement something similar to the joint phrasal model of translation or JPTM (§ 2.4.5). We do so by defining a phrasal ITG that will be trained unsupervised using the EM algorithm. ITG parsing and inside-outside algorithms consider every bitext span of a sentence pair, and each of these spans corresponds to a bilingual phrase pair. In normal ITG processing, each multi-token span is analyzed in terms of how it could be built from two smaller spans using a straight or inverted production, as illustrated in Figures 4.1 (a) and (b). To extend ITG to a phrasal setting, we add a third option for span analysis: that the span under consideration might have been drawn directly from the lexicon of translation pairs. This option can be added to the bracketing grammar in (2.10) by altering the definition of a terminal production to include phrase pairs $(\bar{e}, \bar{f})$:

$$\mathrm{A} \rightarrow [\mathrm{AA}] \mid \langle \mathrm{AA} \rangle \mid \bar{e}/\bar{f} \tag{4.1}$$

This third option is shown in Figure 4.1 (c). Viewed from the top-down, ITG parsing can be interpreted as divide-and-conquer word alignment (Vilar and Vidal, 2005), dividing the initial sentence pair into smaller and smaller phrase pairs that are processed recursively.

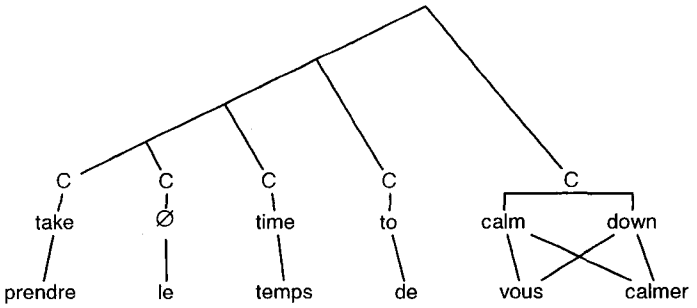

Figure 4.2: Example of a phrasal ITG decomposition.

This modification allows us to stop dividing early, proclaiming a phrase to exist without need for further explanation.

We can extend the canonical bracketing grammar in (2.11) just as easily by modifying the behavior of the pre-terminal C. The modified grammar becomes:

$$\begin{array}{lll} \mathrm{S} \rightarrow & \mathrm{A} \mid \mathrm{B} \mid \mathrm{C} \\ \mathrm{A} \rightarrow & [\mathrm{AB}] \mid [\mathrm{BB}] \mid [\mathrm{CB}] \mid [\mathrm{AC}] \mid [\mathrm{BC}] \mid [\mathrm{CC}] \\ \mathrm{B} \rightarrow & \langle \mathrm{AA} \rangle \mid \langle \mathrm{BA} \rangle \mid \langle \mathrm{CA} \rangle \mid \langle \mathrm{AC} \rangle \mid \langle \mathrm{BC} \rangle \mid \langle \mathrm{CC} \rangle \\ \mathrm{C} \rightarrow & \bar{e}/\bar{f} \end{array} \tag{4.2}$$

The model implied by (4.2) can be trained using the inside-outside algorithm and EM, as described in § 2.2.3. Figure 4.2 shows a phrasal alignment of a English-French sentence pair, and its corresponding canonical ITG decomposition.

Our approach differs from the previous attempts to use ITGs with phrases (§ 2.4.3), because of its use of EM to train a phrasal lexicon. Wu (1997) used a binary bracketing ITG to segment a sentence while simultaneously word-aligning it to its translation, but the model was trained heuristically with a fixed segmentation. Vilar and Vidal (2005) used ITG-like dynamic programming to drive both training and alignment for their recursive translation model, but they employed a conditional model that used IBM-1 phrase scores in place of a phrasal lexicon.

4.2 Comparison to Joint Phrasal Translation Model

Our phrasal ITG is quite similar to Marcu and Wong's joint phrasal translation model, or JPTM (§ 2.4.5). Both models are trained with EM, and both employ generative stories that create a sentence and its translation simultaneously. The similarities become more apparent when we consider the particulars of (4.2), which uses the pre-terminal C to generate all phrase pairs. When (4.2) is parameterized as a stochastic ITG, the conditional distribution $\Pr(\bar{e}/\bar{f}|C)$ is equivalent to the JPTM's $\Pr(\bar{e}, \bar{f})$; both are joint distributions over all possi-

ble phrase pairs. The distributions conditioned on the remaining three non-terminals, S, A and B, assign probability to concept movement by tracking inversions. Like the JPTM's distortion model, these parameters grade each movement decision independently. With terminal productions producing phrase pairs, and inversions measuring distortion, our phrasal ITG is essentially a variation on the JPTM with an alternate distortion model.

The phrasal ITG has three main advantages over the JPTM. Most significantly, we gain polynomial-time algorithms for both Viterbi alignment and EM expectation, through the use of ITG parsing and inside-outside algorithms. These phrasal ITG algorithms are no more expensive asymptotically than their word-to-word counterparts, since each potential phrase needs to be analyzed anyway during constituent construction. We hypothesize that using these methods in place of the JPTM's heuristic search and sampling will improve the phrasal translation model learned by EM. Also, we can easily incorporate links to $\emptyset$ by including the symbol among our terminals. To minimize redundancy, we allow only single tokens, not phrases, to align to $\emptyset$. The JPTM does not allow links to $\emptyset$; instead, tokens that are deleted during translation are incorporated into phrases that also contain tokens that are not deleted. Finally, since the phrasal extension to ITG affects only terminal productions, we have the option of adding syntactic constraints and features to the model, like lexicalization (§ 2.3.1) or syntactic cohesion (§ 5.1). However, we do not explore this third advantage here, so we can maintain as clean a comparison as possible with the JPTM.

The phrasal ITG also introduces two new complications not suffered by the JPTM. Our Viterbi and inside-outside algorithms have polynomial time complexity, but that polynomial is $O(n^6)$, where n is the length of the longer sentence in the pair. This is too slow to train on large data sets without massive parallelization. Also, ITG algorithms explore their alignment space perfectly, but that space has been reduced by the ITG constraint described in § 2.2.4. Fortunately, both of these complications can be overcome. The solutions are discussed below.

4.2.1 Improving Efficiency

ITG algorithms work by analyzing $O(n^4)$ bitext spans, and each analysis takes $O(n^2)$ time. An effective approach to speed up these algorithms is to eliminate unlikely spans as a preprocessing step, assigning them 0 probability and saving the time spent processing them. Zhang and Gildea (2005) improved the efficiency of the ITG inside-outside algorithm with tic-tac-toe pruning, as described in § 2.2.3. We address our scaling problem with a novel span pruning technique, which does not employ Zhang and Gildea's IBM-1 figure of merit.

In its place, we check consistency with a high-confidence word alignment. We refer to the resulting pruning method as fixed-link pruning.

Fixed-Link Pruning

If it is known that a certain link (e_i, f_j) must exist in the final alignment produced by an ITG, a naïve way to use this information would be to ensure that the terminal production $\mathrm{C} \rightarrow e_i/f_j$ is used by the parser. This can be accomplished by pruning word-to-word spans that would have allowed terminal productions of the form $\mathrm{C} \rightarrow e_i/*$ or $\mathrm{C} \rightarrow */f_j$. However, this first attempt does not work well with our phrasal extension; furthermore, it is not fully leveraging the information contained in a fixed link. The presence of a link also says a lot about which multi-token spans are available to the ITG. We propose a new ITG pruning method that leverages high-confidence links by pruning all bitext spans that are inconsistent with a provided fixed alignment. To do so, we make use of the same notion of phrasal consistency that is used for heuristic phrase extraction in phrasal decoding (§ 2.6.1). Recall that consistency is defined so that alignment links are never broken by phrase boundaries; so by pruning all bitext spans that correspond to inconsistent phrase pairs, we are pruning all bitext spans that include one token from a link without including the other. Under this fixed-link pruning, a single centrally-placed link can eliminate more than 50% of the spans processed by a normal ITG, while doing something as simple as linking end-of-sentence punctuation can still eliminate more than 15% of spans. Furthermore, since fixed-link pruning selectively eliminates bilingual spans and not entire monolingual spans, it naturally works with the canonical-form grammar in (2.11), unlike the syntactic pruning methods discussed in § 5.

Fixed-link pruning will speed up ITG processing, and it will do so harmlessly, so long as we trust our fixed links. In general, one could try adding fixed links according to any number of sources, such as tokens pairs that fulfill an exact word match, a dictionary match, or the punctuation match alluded to above. Doing so can leverage information not normally available to an ITG, in addition to increasing efficiency. However, we take this idea one step further, and actually incorporate a complete high-confidence one-to-one alignment, provided by GIZA++ intersection (§ 2.4.1). As we show in our experiments, this has a dramatic affect on efficiency. Using GIZA++ intersection as fixed links can also be seen as leveraging external information, as GIZA++ uses a feature set that is quite different from that of an ITG, especially regarding distortion (§ 2.1.1).

In addition to improving time efficiency, fixed-link pruning reduces the size of the po-

tentially huge phrasal translation lexicon by constraining the ITG's choices for phrasal links. This sort of lexicon pruning is similar and comparable to the consistency constraint used in the C-JPTM (Birch et al., 2006), but we eliminate inconsistent spans completely, and not only as potential phrase-to-phrase links. The C-JPTM re-introduces inconsistent phrase-pairs in cases where the sentence pair could not be aligned otherwise. We allow links to $\emptyset$ to handle these situations, completely eliminating the pruned spans from our alignment space. Like other joint phrasal models, we also employ a direct **lexicon constraint**, which eliminates any phrase-pair from our translation lexicon that contains a monolingual phrase occurring fewer than 5 times.

Detecting Inconsistent Spans

We run fixed-link pruning online as the ITG processes new sentence pairs; therefore, it is important to have an efficient algorithm to detect inconsistent spans. We sketch one such algorithm here.

First, alignment vectors $\vec{a}^{\mathrm{E}}$ and $\vec{a}^{\mathrm{F}}$ are built for the sentence pair $(\vec{e}, \vec{f})$, according to a fixed alignment A. The entry a_i^{E} stores both the left and right-most indices of tokens in $\vec{f}$ that are linked to e_i, and a_j^{F} is defined similarly. Since there are at most n^2 links in A, vector construction is bound by $O(n^2)$. These vectors can then be used to compute the left and right bounds of the smallest consistent translation for each monolingual span in the sentence pair. For example, $bnd(e_s^t)$ would return the smallest French span $f_{s'}^{t'}$ that contains all French tokens linked to any token in e_s^t. These bounds can be calculated efficiently using the following recurrence:

$$\forall s : bnd(e_s^{s+1}) = a_s^{\mathrm{E}}$$

$$\forall s, t \mid t > s+1 : bnd(e_s^t) = f_{s'}^{t'}, \text{ where: } \begin{array}{l} t' = \max[bnd(e_s^{t-1}).right,\ bnd(e_{t-1}^t).right] \\ s' = \min[bnd(e_s^{t-1}).left,\ bnd(e_{t-1}^t).left] \end{array}$$

The French recurrence is similar. Computing all minimum spans can be done in $O(n^2)$ time for each language. At this point, each bilingual span pair can be assessed in constant time. A bitext span (e_s^t, f_u^v), where $bnd(e_s^t) = f_{s'}^{t'}$ and $bnd(f_u^v) = e_{u'}^{v'}$, is consistent with A so long the following holds:

$$u \leq s' \text{ and } s \leq u' \text{ and } v \geq t' \text{ and } t \geq v'$$

Some additional special handling is required for unlinked tokens, which place no constraints on the available spans; this has been omitted for the sake of clarity. The complexity of the entire process is $O(n^2)$, making fixed-link pruning quite affordable, given an alignment.

4.2.2 Handling the ITG Constraint

Our remaining concern is the ITG constraint. There are some alignments that we just cannot build, and sentence pairs requiring those alignments will occur. These could potentially pollute our training data; if the system is unable to build the right alignment, the counts it collects from that sentence pair must be wrong. Furthermore, if our high-confidence links are not ITG-compatible, our fixed-link pruning will prevent the aligner from forming any alignments at all, as the parser will be unable to complete even one derivation.

However, these two potential problems cancel each other out. Sentence pairs containing non-ITG translations will tend to have high-confidence links that are also not ITG-compatible. Our EM learner will simply skip these sentence pairs during training, avoiding pollution of our training data. We can use a linear-time algorithm (Zhang et al., 2006) to detect non-ITG movement in our high-confidence links, and remove the offending sentence pairs from our training corpus. This results in only a minor reduction in training data; in our Europarl English-French training set, we lose less than 1% of sentence pairs.

4.3 Applying the Model

Any phrasal translation model can be used for two tasks: translation modeling and phrasal word alignment. The former constructs a translation model or phrase table for use in decoding without necessarily committing to a single alignment, while the latter builds a single alignment structure that allows for many-to-many links. Previous work on the JPTM has focused only on translation modeling. The hope is that the JPTM's strength in reasoning about distributions over all possible phrasal alignments will provide it leverage over traditional phrase-extraction techniques. However, it is possible that a true phrasal alignment may also provide better input for traditional phrase-extraction techniques. Therefore, we detail how our model can be used for both tasks here, and we provide a comparison between the two approaches in a translation task in our experiments in § 4.4.

4.3.1 Translation Modeling

Once the phrasal ITG has been trained with EM on a parallel corpus, we can employ the model directly for translation by transforming its parameters into a phrase table for a phrasal decoder, such as Pharaoh (Koehn et al., 2003). Any joint model can produce the necessary conditional phrase tables by conditionalizing the joint table in both directions. We use our

joint $\Pr(\bar{e}/\bar{f}|C)$ distributions to do so, producing the two conditional distributions:

$$\Pr(\bar{e}|\bar{f}) = \frac{\Pr(\bar{e}/\bar{f}|C)}{\sum_{\bar{e}} \Pr(\bar{e}/\bar{f}|C)} \quad \text{and} \quad \Pr(\bar{f}|\bar{e}) = \frac{\Pr(\bar{e}/\bar{f}|C)}{\sum_{\bar{f}} \Pr(\bar{e}/\bar{f}|C)}$$

Pharaoh decoding generally includes lexical weighting parameters that are derived from the alignments used to induce its phrase pairs (Koehn et al., 2003). Using the phrasal ITG as a direct translation model, we do not commit to any one alignment for a given sentence pair. To compensate for our lack of alignments with which to generate this lexical weighting, we provide a lexical preference to the decoder with an IBM Model 1 feature p_{M1} that penalizes unmatched words (Vogel et al., 2003).

$$p_{\text{M1}}(\bar{f}|\bar{e}) = \prod_{j=1}^{n} \sum_{i=0}^{m} \Pr(f_j|e_i)$$

We will generally include both $p_{\text{M1}}(\bar{e}|\bar{f})$ and $p_{\text{M1}}(\bar{f}|\bar{e})$, trained independently on the same data set, using IBM Model 1 training as described in § 2.1.1.

4.3.2 Phrasal Word Alignment

A trained phrasal ITG can provide a phrasal word alignment through a simple application of the Viterbi parsing algorithm described in § 2.2.3. This returns the maximum likelihood parse under the model, which also provides an alignment. Despite its strengths derived from using phrases throughout training, the alignments produced by the phrasal ITG are usually unsatisfying. For example, the fragment pair ("I cannot", "Je ne puis") shown in Figure 4.3a is aligned as a phrase pair by our system, linking every English word to every French word. This is frustrating, since there is a clear compositional relationship between the tokens "I" and "Je". This happens because the system seeks only to maximize the likelihood of its training corpus, and phrases are far more efficient than word-to-word connections. Any joint phrasal translation model will suffer from this problem. When aligning text, annotators are told to resort to many-to-many links only when no clear compositional relationship exists (Melamed, 1998). If we could tell the phrasal aligner the same thing, it could greatly improve the intuitive appeal of its alignments. Fortunately, we can turn to the high-confidence links used in fixed-link pruning for help.

In the high-confidence alignments provided by GIZA++ intersection, each token participates in at most one link. Links only appear when two word-based IBM translation models can agree. Therefore, they occur at points of high compositionality: the two words clearly account for one another. GIZA++ intersection links only two token-pairs in our running

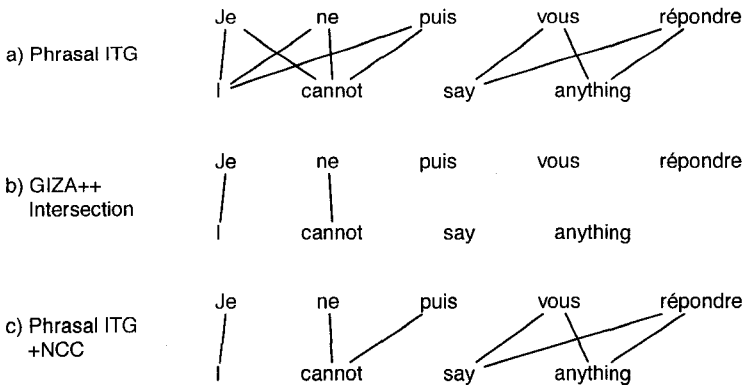

Figure 4.3: Three alignments illustrating the non-compositional constraint.

example, shown in Figure 4.3b. We adopt an alignment-driven definition of compositionality; any phrase pair containing two or more high-confidence links is compositional, and can be separated into at least two non-compositional phrases. By removing any phrase pairs that are compositional by this definition from our pool of terminal productions, we can ensure that our phrasal ITG aligner never creates such phrases during training or alignment. That is, the model only builds non-compositional phrases. Doing so produces far more intuitive alignments. Aligned with a phrasal ITG trained using this **non-compositional constraint** (NCC), our example has now separated out the ("I","Je") concept into a distinct unit, eliminating three spurious links, as shown in Figure 4.3c. The phrases produced with this constraint are very small, and include only non-compositional context. Therefore, we use the constraint only to train models intended for Viterbi alignment, and not when generating phrase tables directly for the translation modeling task described above.

4.4 Experiments

We have presented a phrasal extension to ITG, expanding its alignment space to allow for phrasal non-terminals, and therefore allowing many-to-many alignments. This phrasal ITG is similar to the JPTM (§ 2.4.5), as both methods generate joint phrasal translation lexicons. However, the ITG has the advantage of having efficient, exact algorithms for alignment and expectation, which we hope will produce superior translation models. We begin our phrasal experiments by verifying the effectiveness of fixed-link pruning, which is essential to EM-learning the parameters for our phrasal ITG. We then test our phrasal ITG, both as a phrasal word-alignment method and as a translation model for use in phrasal decoding. These experiments were originally published in (Cherry and Lin, 2007).

4.4.1 Experimental Details

This section describes the infrastructure used to evaluate our phrasal ITG. We begin by describing our Europarl corpus and our baseline translation infrastructure. Both of these are also used in our cohesive decoding experiments in § 6.3. To avoid repetition, we provide sufficient details for both cases here. Finally we describe the implementation of both our phrasal ITG and our C-JPTM baseline. The designs for individual experiments are presented in their respective subsections.

Europarl Corpus

The training sets for the HLT-NAACL 2006 SMT Shared Task (Koehn and Monz, 2006) are a good source of publicly-available sentence-aligned bitext. They are drawn from the proceedings of the European Parliament, or Europarl. For these experiments, we make use of their French-English bitext, which has 688K sentence pairs. We also employ a smaller 393K subset of this bitext, derived by applying a 25-token sentence-length limit to the corpus. Finally, we have a smaller subset still, employed only during development, that is derived from the first 50K sentence pairs of the 393K corpus. Results are not reported using this smallest subset.

This English-French bitext comes with development and test sets for translation evaluation, which are not word-aligned. These are withheld from training in order to determine translation quality using BLEU (§ 3.2). There is a 2000 sentence-pair set provided for development, which we have divided into a 500-pair tuning set for MERT training, and a 1500-pair development set. There is also a distinct 2000 sentence-pair test set. For the sake of these experiments, we employ only the tuning and test sets.

We have manually word-aligned the first 100 sentence-pairs from our 393K training set. These are aligned by the author, using the Blinker annotation guidelines (Melamed, 1998), using only sure links. These alignments are used to evaluate word-alignment quality for systems trained with our Europarl data. This alignment test set follows the recommendations found in (Fraser and Marcu, 2006a).

Translation Infrastructure

Our baseline phrase-based translation system is constructed from publicly-available components, which include a weighted phrase table, a decoder, a language model, a distortion model, and a minimum error rate training method. Below we describe each of these components as they are implemented in our baseline system. The majority of our bitext processing

tools are drawn from the SMT Workshop 2006 baseline system (Koehn and Monz, 2006), which we refer to as SMTW.

Phrase table Our baseline phrase table is constructed using heuristic phrase extraction as described in § 2.6.1. The bitext is aligned in both translation directions using GIZA++, with five iterations each of IBM-1, HMM and IBM-4. Those alignments are combined with the grow-diag-final combination heuristic using a script provided by the SMTW. Another SMTW script handles phrase extraction, and collects conditional phrase-based translation probabilities as well as lexical weighting scores. Collected phrases have a maximum length of 7 tokens. In our phrasal ITG experiments, we replace either the alignment step or both the alignment and phrase extraction steps with our phrasal ITG.

Decoding We employ two different decoders for historical reasons. In this chapter, where we vary only the phrase table, we employ the Pharaoh decoder (Koehn et al., 2003), as that is the decoder provided by the SMTW. Later work in § 6.3 involves inserting a constraint into the decoder itself, so we shift to the open-source, Pharaoh-feature-complete Moses decoder (Koehn et al., 2007). Both decoders implement standard left-to-right phrasal decoding as described in § 2.6.1.

Language model The SMTW provides trigram language models learned using the SRILM toolkit for all of its test languages. We use the appropriate target language model without modification in each of our experiments.

Distortion model Our decoders implement a simple distortion penalty that measures linear non-monotone transitions in the source sentence. We use this as our distortion model by default. Our decoders also implement hard distortion limits. Unless stated otherwise, we use a distortion limit of 4. In our cohesive decoding experiments in § 6.3, we also employ a lexical re-ordering model. The Moses decoder contains code to process such a model, and the SMTW provides scripts to build the necessary tables from word-aligned bitext.

Minimum Error Rate training The weights in our log-linear feature model are tuned with minimum error rate training (Och, 2003) as implemented by Venugopal and Vogel (2005). We minimize the translation error observed on our 500-sentence-pair tuning set according to the BLEU metric. We use scripts developed in-house to it-

erate between K-best-list decoding, K-best-list merging, and model tuning until a local maximum is reached.

Model Implementation Details

Our phrasal ITG is an in-house implementation that extends our common ITG architecture with fixed-link pruning, an optional non-compositional constraint, and phrasal nonterminals. The most relevant baseline for comparison is the C-JPTM, which as outlined in § 4.2, is very similar to our phrasal ITG, except for its use of heuristic search and sampling when collecting expectations. The version of the C-JPTM used in our experiments was supplied by Alexandra Birch through personal communication. We run their code with default parameters, but with the maximum phrase length increased to 5 tokens.[1]

We train all translation models with our 393K English-French Europarl corpus. For the phrasal ITG, 3,376 of the sentence pairs are omitted because their high-confidence alignments have ITG-incompatible constructions. This limits the impact of the ITG re-ordering constraints on our inside-outside training algorithms (§ 4.2.2). Following (Marcu and Wong, 2002; Birch et al., 2006), we apply a lexicon constraint to both our baseline C-JPTM and our phrasal ITG: no monolingual phrase can be used by either model unless it occurs at least five times in our training set. In our phrasal ITG, this is implemented during the initialization step of the forward algorithm, where each proposed $\mathrm{C} \rightarrow e/f$ production is checked to ensure that both e and f meet the required monolingual frequency.

High-confidence alignments for fixed-link pruning are provided by intersecting GIZA++ alignments trained in each direction. All GIZA++ alignments are trained with 5 iterations each of Model 1, HMM, and Model 4. Furthermore, all GIZA++ alignments, including the ones we compare against in our alignment experiments, are trained with no sentence-length limit, using the full 688K Europarl corpus.

4.4.2 Pruning Speed Experiments

To measure the speed-up provided by fixed-link pruning, we time our phrasal inside-outside algorithm on the first 100 sentence pairs in our training set, with and without pruning. The results are shown in Table 4.1. Tic-tac-toe pruning is included for comparison. With fixed-link pruning, on average 95% of the possible spans are pruned, reducing the unpruned

[1]Our phrasal ITG requires no limit on phrase length, while heuristic phrase extraction has a maximum of 7 tokens. These phrase length limits are present only for efficiency concerns. Heuristic extraction is already memory-intensive, and would be impossible without a length limit or alternate data structures (Callison-Burch et al., 2005). In the cases of our phrasal ITG and the C-JPTM these length limits have very little effect on translation performance, as their frequency-based lexicon constraint already eliminates most long phrases.

Method	Seconds	Avg. Spans Pruned
No Prune	415	-
Tic-tac-toe	37	68%
Fixed link	5	95%

Table 4.1: Inside-outside run-time comparison of pruning methods.

running time by two orders of magnitude. This improvement makes ITG training feasible, even with large bitexts. Tic-tac-toe pruning also makes a large difference, but it remains roughly seven times slower than fixed-link pruning.

4.4.3 Alignment Experiments

Any phrasal translation model can be used for two tasks: translation modeling and phrasal word alignment. Previous work on the JPTM has focused on the translation modeling task, decoding with the learned models without ever checking alignment quality. We are interested in phrasal alignment because it may be the case that these phrasal alignments are more valuable to translation than the phrase table itself. Heuristic phrase extraction (§ 2.6.1) has intrinsic advantages over any EM-learned phrase table, as its flat counts capture translation relationships at several levels of resolution: a phrase can be memorized as a phrasal translation and as a number of component word-to-word translations simultaneously, while the EM-learned joint model will naturally prefer to select one over the other. It may be the case that a phrasal alignment is better suited to heuristic phrase-extraction than word-based alignment models. Therefore, the goal of this experiment is to validate our non-compositional constraint (§ 4.3.2) and to select good alignments for heuristic phrase extraction. We do this by comparing our Viterbi phrasal ITG alignments to gold standard human alignments. Later in § 4.4.4 we test our hypothesis regarding phrase extraction.

We use our word-aligned, 100 sentence Europarl test set as a gold standard. For comparison purposes, we include the results of three types of GIZA++ combination, including the grow-diag-final heuristic (GDF). We test our phrasal ITG with fixed-link pruning, and then add the non-compositional constraint (NCC), which is described in § 4.3.2. During development we determined that performance levels off for both of the ITG models after 3 EM iterations. The results are shown in Table 4.2.

The first thing to note is that GIZA++ Intersection is indeed very high precision. Our confidence in it as a constraint is not misplaced. We also see that both phrasal models have significantly higher recall than any of the GIZA++ alignments, even higher than the per-

Method	Prec	Rec	F-measure
GIZA++ Intersect	**96.7**	53.0	68.5
GIZA++ Union	82.5	69.0	75.1
GIZA++ GDF	84.0	68.2	75.2
Phrasal ITG	50.7	**80.3**	62.2
Phrasal ITG + NCC	75.4	78.0	**76.7**

Table 4.2: Comparison of the phrasal ITG aligner to GIZA++ combination.

missive GIZA++ union. One factor contributing to this is the phrasal model's use of cepts: it completely interconnects any phrase pair, completing the transitive closer of multi-word translation relationships, while GIZA++ union and GDF may not. The phrasal ITG's global view of phrases also helps in this regard: evidence for a phrase can be built up over multiple sentences, while GIZA++ combination handles each sentence separately based on word-to-word links. Finally, we note that in terms of alignment quality, the non-compositional constraint is an unqualified success for the phrasal ITG. It produces a 25 point improvement in precision, at the cost of 2 points of recall. This produces the highest balanced F-measure observed on our test set, but the utility of these alignments will depend largely on one's desired precision-recall trade-off.

4.4.4 Translation Experiments

In this section, we compare a number of different methods for phrase table generation in a French to English translation task. We are interested in answering three questions:

1. Does the phrasal ITG improve on the C-JPTM?
2. Can joint phrasal translation models outperform the phrase extraction surface heuristic described in § 2.6.1?
3. Do Viterbi phrasal alignments provide better input for the surface heuristic?

With this in mind, we test five phrase tables. Two are conditionalized phrasal translation models, each EM trained until performance degrades. These methods build phrase tables without committing to a single Viterbi alignment:

- C-JPTM as described in (Birch et al., 2006) and outlined in § 2.4.5
- Phrasal ITG as described in Section 4.3.1

The three remaining methods provide Viterbi alignments for standard phrase extraction using the surface heuristic:

Method	BLEU	+M1	Size
Conditionalized Phrasal Model			
C-JPTM	26.27	28.98	1.3M
Phrasal ITG	28.85	30.24	2.2M
Alignment with Surface Heuristic			
GIZA++ GDF	30.46	30.61	9.8M
Phrasal ITG	30.31	30.39	5.8M
Phrasal ITG + NCC	**30.66**	**30.80**	9.0M

Table 4.3: Translation comparison of various phrase tables.

- GIZA++ with grow-diag-final (GDF)
- Viterbi Phrasal ITG with and without the non-compositional constraint (NCC)

We use our translation infrastructure described in § 4.4.1 with Pharaoh to decode using our various phrase tables. Results on our 2000-sentence Europarl test set are reported using the BLEU metric. For all methods, we report performance with and without the IBM Model 1 features (M1) described in § 4.3.1. We also report the size of the resulting phrase tables in millions of phrase pairs. The results of all experiments are shown in Table 4.3.

Among the conditionalized joint phrasal models, we see that the Phrasal ITG surpasses the C-JPTM by more than 2.5 BLEU points. A large component of this improvement is due to the ITG's use of inside-outside for expectation calculation, though there are other differences between the two systems.[2] This improvement over search and sampling is demonstrated by the ITG's larger table size; by exploring more thoroughly, it is extracting more phrase pairs from the same amount of data.

Both systems improve drastically with the addition of IBM Model 1 features for lexical preference. These features also narrow the gap between the two systems. To help calibrate the contribution of the IBM-1 features, we parameterized the ITG's phrase table using only IBM-1; that is, the conditionalized joint probabilities learned by the ITG were removed, leaving only the IBM-1 scores in both translation directions. This phrase table scores 27.17, indicating that the IBM-1 features are fairly strong on their own; in fact, they outperform the C-JPTM alone.

Although ITG+M1 comes close, neither phrasal model matches the performance of the surface heuristic. Whatever the surface heuristic lacks in sophistication, it makes up for in

[2]Unlike our system, the Birch implementation does table smoothing and internal lexical weighting, both of which should help improve their results. In particular, smoothing allows them to re-introduce phrases previously eliminated by the lexicon constraints, which we do not do. The systems also differ in distortion modeling and $\emptyset$ handling, as described in Section 4.2.

sheer coverage, as demonstrated by its huge table sizes. Even the Phrasal ITG Viterbi alignments, which over-commit wildly and have horrible precision, result in extracted phrase tables that score slightly higher than the best EM-learned phrasal model. The surface heuristic benefits from capturing as much context as possible, while still covering smaller translation events with its flat counts. Also, it is not held back by any lexicon constraints. When GIZA++ GDF+M1 is forced to conform to a lexicon constraint by dropping any phrase with a frequency lower than 5 from its table, it scores only 29.26, for a reduction of 1.35 BLEU points. The surface heuristic's lack of lexicon constraints appears to be a major factor in its success over our EM-learned tables from the phrasal ITG.

Phrases extracted from our non-compositional Viterbi alignments receive the highest BLEU score, but they are not significantly better than GIZA++ GDF. The two methods also produce similarly-sized tables, despite the ITG's substantially higher recall. Unfortunately, since we are altering the alignment process in the hopes of receiving better phrase tables, and we are not targeting a specific problem with decoding, it is difficult to design a human evaluation to check whether or not translations are improving more than is indicated by BLEU score.

4.5 Summary

We have presented a phrasal ITG as an alternative to the JPTM. This syntactic solution to phrasal modeling admits polynomial-time training and alignment algorithms. We demonstrated that our novel fixed-link pruning method dramatically speeds up ITG training, producing an 80-times faster inside-outside algorithm. We have shown that when used to learn phrase tables for the Pharaoh decoder, the phrasal ITG is superior to the C-JPTM, producing tables that result in a 2.5 point improvement in BLEU when used alone, and a 1 point improvement when used with IBM-1 features. This suggests that the phrasal ITG's perfect expectation does matter; other phrasal models could benefit from either adopting the ITG formalism, or improving their sampling heuristics. We have also demonstrated that the lexicon constraints on both the C-JPTM and the phrasal ITG appear to be holding them back. Finally, we have explored, for the first time, the utility of a joint phrasal model as a word alignment method. We have presented a non-compositional constraint that turns the phrasal ITG into a high-recall phrasal aligner, producing F-measures that are comparable to those of GIZA++.

Chapter 5

Syntactic Constraints for Word Alignment

With our phrasal ITG in § 4, we use the syntactic assumptions inherent in an ITG to gain efficient algorithms for phrasal bitext modeling. This demonstrates the algorithmic value of making some assumptions regarding how concepts move during translation. However, the ITG's assertions regarding context movement are actually fairly weak: the use of an ITG asserts that there is some context-free decomposition of the sentence pair, but the ITG is free to find the most convenient syntactic structure. Zhang and Gildea (2004) describe this sort of syntactic influence with the term **unsupervised syntax**, since the syntactic structure is not fixed in advance of alignment. They contrast this with **supervised syntax**, where the structure of one of the two sentences in a pair is determined before alignment begins, using a monolingual parser. The tree-to-string translation model described in § 2.3.3 is an example of a syntactically supervised alignment method.

In this chapter, we introduce supervised syntax to ITG parsing. We do so by dependency parsing the English side of each sentence pair, and then constraining the ITG to respect the syntactic structures found in this fixed, English parse tree. This use of the dependency tree leverages only its structure, ignoring any part-of-speech tags or dependency labels induced during parsing. Unlike our phrasal ITG, where we introduced constraints to create polynomial algorithms, in this case, we hope to further constrain the ITG in order to improve accuracy. One weakness of the binary bracketing ITG is the lack of information available when making decisions regarding non-terminal productions such as $A \rightarrow [AB]$. The corresponding stochastic ITG parameter $\Pr([AB]\,|A)$ considers only the three non-terminals involved, and whether or not the left-hand-side is inverted. The non-terminals are selected for purely mechanical reasons, and carry very little information beyond predicting inver-

sion.[1] One way to add information at non-terminal nodes is to have some constraints or preferences on the constituents the ITG can produce, based on the English dependency tree.

Adding new syntactic constraints to a basic ITG will allow us to revisit the question of supervised versus unsupervised syntax. Zhang and Gildea (2004) compared the tree-to-string alignment model to ITGs. They concluded that the syntactically unsupervised ITGs perform better than methods with a fixed tree established before alignment begins. However, the use of a fixed tree is not the only difference between the tree-to-string model and ITGs; the probability models are also very different. By using a fixed dependency tree to constrain an ITG, we are able to revisit the question of whether using a fixed tree is harmful, but in a controlled environment.

Throughout this chapter, we enhance a one-to-one binary bracketing ITG with syntactic information from an English dependency tree. We do not continue to extend our phrasal ITG, because we wish to isolate the issue of fixed syntax without adding further complications. We present three algorithmic and two experimental contributions, each outlined in its own section. § 5.1 describes two methods to enforce the syntactic cohesion of a dependency tree inside an ITG parse, and also contains a substantial discussion on the relationship between ITG and cohesion constraints. § 5.2 pulls in even more information from the dependency tree by adding a constraint to respect head relationships. In § 5.3 we pause and test the new alignment spaces we have created, using simple weighted aligners to measure the guidance and expressiveness of these spaces. In § 5.4, we adopt a discriminative training method for use with ITGs to create a soft cohesion constraint for ITG alignment, and § 5.5 tests this new aligner against other strong, discriminative baselines.

5.1 Syntactic Cohesion

A cohesion constraint limits concept movement according to a fixed tree structure; syntactic subtrees are forced to move together during translation (§ 2.3.3). Previous work in (Lin and Cherry, 2003) has shown that enforcing the syntactic cohesion of an English dependency tree during alignment can drastically improve alignment quality. In that work, the alignment search was conducted by a beam search in one-to-one alignment space. The search added links one at a time to a growing alignment hypothesis, and each new link was checked to see if it violated syntactic cohesion with respect to the current hypothesis and a fixed dependency tree.

[1] In the canonical bracketing ITG (2.11), A always produces a straight non-terminal pair, while B produces only inverted pairs, and C is a pre-terminal.

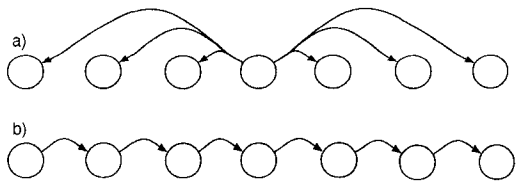

Figure 5.1: Maximally (a) and minimally (b) permissive dependency trees.

The primary objective of the work presented in this subsection is to embed the cohesion constraint inside an ITG. In this way, we can benefit from the guidance of syntactic cohesion with respect to a fixed dependency tree, while also enjoying the benefits of ITG, such as perfect search and a clean unified formalism. Such an embedding would also help elucidate the relationship between the cohesion constraint and the ITG framework. We begin by discussing the qualitative relationship between the alignment spaces defined by ITG and dependency-tree syntactic cohesion, before presenting our two solutions to the problem of constraining an ITG with syntactic cohesion.

5.1.1 Qualitative Alignment Space Comparison

We use the term **cohesion space** to refer to the set of all alignments that maintain syntactic cohesion with respect to a dependency tree provided for the English sentence. The number of alignments in cohesion space depends largely on the provided dependency tree. During a cohesive translation, all permutations of a head and its modifiers are possible; therefore, the number of permutations allowed by a tree can be found by taking a product of the possible ordering decisions. In general, there are $\prod_h [(c(h)+1)!]$ permutations for a given tree, where h stands for a head node in the tree, and $c(h)$ counts h's modifiers. Therefore, a depth-1 tree, where all modifiers have the same head, provides no extra guidance to an aligner. The alignment space is equal to **permutation space**, allowing all $n!$ permutations of concepts. Conversely, if the tree is a chain, where every head has exactly one modifier, the alignment space is tightly constrained, allowing $2^{(n-1)}$ re-orderings, far fewer than permutation space. Trees leading to these two extremes are illustrated in Figure 5.1. Cohesion space is not a subspace of ITG space, as it can create both of the forbidden alignments in Figure 2.6 when given a depth-1 tree.

ITG space, the alignment space enumerated by a binary ITG, is discussed at length in § 2.2.4. This space is also related to syntactic cohesion, but the tree used to determine cohesion is no longer fixed. All ITG alignments maintain cohesion with respect to at least one of the possible binary-bracketing constituency trees that can be built over the English

sentence. Zens and Ney (2003) show that the size of this space is equal to the $(n-1)th$ large *Schröder* number, making the number permutations explored by an ITG $O(5.83^n)$. In terms of size, this places ITG space between the two extremes of cohesion space.

By embedding a cohesion constraint inside an binary bracketing ITG, we introduce a new alignment space defined by a cohesion constrained ITG, or C-ITG. The set of possible alignments in this space is the intersection of cohesion space for a fixed dependency tree and ITG space. Equivalently, the space respects the phrases specified by a dependency tree, but attempts all ITG re-orderings of a head and its modifiers, rather than all possible permutations. Since it is a subspace of ITG space, we will be able to search the space completely using a polynomial time ITG parser. This places an upper bound on the search complexity equal to ITG complexity. This upper bound is very loose, as the ITG will often be drastically constrained by the structure of the dependency tree. We present two methods that implement such an embedding, one through chart modification, and another through a custom grammar.

5.1.2 Chart Modification

Wu (1997) suggests that in order to have an ITG take advantage of a known partial structure, one can prune any spans that would violate the structure. In an ITG chart parsing framework, this can be accomplished by assigning the **invalid spans** a probability of 0 before parsing begins. The sentence pair can then be parsed normally, automatically respecting the known structure.

Our English dependency tree qualifies as a partial structure, as it does not specify a complete binary decomposition of the English sentence. Instead, through its subtrees, it specifies a number of syntactic phrases e_s^t that must remain contiguous after translation. We can ensure that e_s^t is left contiguous by forcing the ITG to build a constituent $\mathrm{X}_{s,u}^{t,v}$ for some non-terminal X and French indices u and v. With the entire phrase collected under a constituent, any inversion above the constituent's root in the tree will move the phrase as a unit, while any inversion under it will move only concepts within the phrase, leaving cohesion intact.

One can force the ITG to build a constituent over a phrase by invalidating any span that has one endpoint inside the phrase, but another outside of it. In this way, no constituent can include part of the phrase without including all of it. To put this notion formally, we first define some terms: given a subtree $T_{[i,k]}$, where i is the left index of the leftmost leaf in $T_{[i,k]}$ and k is the right index of its rightmost leaf, we say any index $j \in (i,k)$ is **internal**

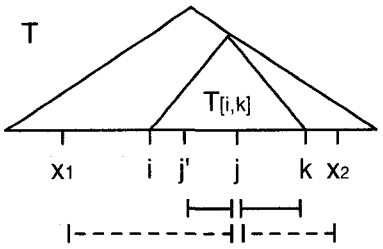

Figure 5.2: Example spans. $[j', j]$ and $[j, k]$ are valid, while $[x_1, j]$ and $[j, x_2]$ are not.

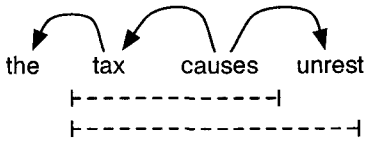

Figure 5.3: The invalid spans induced by a dependency tree.

to $T_{[i,k]}$. Similarly, any index $x \notin [i, k]$ is **external** to $T_{[i,k]}$. An invalid span is any span for which our provided tree has a subtree $T_{[i,k]}$ such that one endpoint of the span is internal to $T_{[i,k]}$ while the other is external to it. Figure 5.2 illustrates this definition, while Figure 5.3 shows the invalid spans induced by a simple dependency tree. Note that the two illegal spans are those that would interrupt the "the tax" subtree.

Invalid English spans can be detected efficiently using an algorithm inspired by the definition of invalid spans given above. For each index j in our English sentence, we find the deepest multi-node subtree $T_{[i,k]}$ that contains j. We can then bound the endpoints of any span that begins or ends with j using $[i, k]$. Any containing subtree with a more shallow root will produce less strict bounds. The pseudo-code shown in Algorithm 3 collects bounds for each index, and given these bounds, provides a constant-time check for invalid-spans. Algorithm 3(a) provides a recursive function to return the index of the left or right-most child in a subtree with a given head. A $O(m)$ pre-processing step can memoize this function so that each call takes constant time. Part (b) takes advantage of a top-down traversal of heads to mark each index with the bounds of its deepest containing subtree. This process has a time complexity of $O(m^2)$. Part (c) checks to see if each endpoint is within the bounds of its partner in order to test validity.

The chart modification method to constrain an ITG with dependency cohesion is easy to add to an existing ITG parser. Since it eliminates spans, it can be clearly interpreted as a method of ITG pruning, like the tic-tac-toe method in (Zhang and Gildea, 2005) or our own fixed-link pruning (§ 4.2.1). In the C-ITG case, we prune multiple bitext spans

Algorithm 3 Algorithm to detect invalid spans given a dependency tree $T_{[0,m]}$.

```
{(a) Function to recursively determine the maximal left or right projection of a node}
proj(head, dir=left|right):
  child ← dir-most child of head
  if child exists return proj(child, dir)
  else return dir index of head

{(b) Find bounds for each index}
for each head in a pre-order traversal of T do
  for j : proj(head, left) < j < proj(head, right) do
    bound(j, left) ← proj(head, left)
    bound(j, right) ← proj(head, right)
  end for
end for

{(c) Test for invalidity, compare each endpoint to the bound of the other endpoint}
isInvalid(s, t):
  return (s < bound(t, left) or t > bound(s, right))
```

for each invalidated English span, one for each possible French span with which it could be paired. Because this approach prevents an otherwise complete ITG from exploring all spans, it will also lend itself easily to a soft constraint (§ 5.4). However, this method does have a disadvantage. Recall that an ITG using the binary bracketing grammar in (2.10) can construct several structures for the same alignment. This redundancy is necessary for the chart modification approach to work. Should we require a canonical representation, the canonical bracketing grammar in (2.11) cannot be used to eliminate redundancy, because it selects one ITG structure to represent each alignment without knowledge of the fixed dependency tree. It can potentially select a structure that uses invalid spans to be the unique structure for an alignment, when other valid structures could produce the same alignment. This results in missed cohesive alignments. To handle cases where a canonical form is required, we also present a custom grammar method.

5.1.3 Custom Grammar

We stated earlier that C-ITG space is equivalent to only allowing movement that corresponds to ITG-compatible re-orderings of dependency subtrees. We can implement this notion directly with a recursively-constructed custom grammar. This will lead to a C-ITG with a canonical representation for each possible alignment. Let a **local tree** be the tree formed by a head node and its immediate modifiers. We begin our recursive process by considering the local tree at the root of our dependency tree, and marking each phrasal

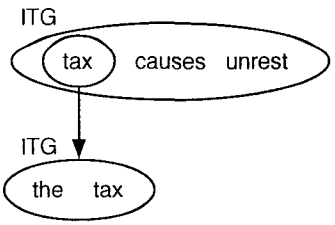

Figure 5.4: A recursive ITG.

modifier with a labeled placeholder. We then create a string by flattening the local tree. The top oval of Figure 5.4 shows the result of this operation on our running example. Because all phrases have been collapsed to placeholders, an ITG built over this string will naturally respect the dependency tree's syntactic boundaries. Since we do not need to invalidate any spans, we can parse this string using the canonical ITG in (2.11). The phrasal modifiers can in turn be processed by applying the same algorithm recursively to their root nodes, as shown in the lower oval of Figure 5.4.

This procedure will explore the exact same alignment space as the chart modification solution, but because it uses a canonical ITG at every ordering decision point, it will produce exactly one structure for each alignment. This recursive approach can be implemented inside a traditional ITG framework using **grammar templates**. The templates take the form of whatever grammar will be used to permute the local trees. They are instantiated over each local tree before ITG parsing begins. Each instantiation has its non-terminals marked with their corresponding spans, and its pre-terminal productions are customized to match the modifiers of the local tree. Phrasal modifiers point to another instantiation of the template. In our case, the template corresponds to the canonical form grammar in (2.11). The result of applying the templates to our running example is:

$$
\begin{aligned}
S_{0,4} &\rightarrow A_{0,4} \mid B_{0,4} \mid C_{0,4} \\
A_{0,4} &\rightarrow [A_{0,4}B_{0,4}] \mid [B_{0,4}B_{0,4}] \mid [C_{0,4}B_{0,4}] \mid \\
&\quad\ [A_{0,4}C_{0,4}] \mid [B_{0,4}C_{0,4}] \mid [C_{0,4}C_{0,4}] \\
B_{0,4} &\rightarrow \langle A_{0,4}A_{0,4}\rangle \mid \langle B_{0,4}A_{0,4}\rangle \mid \langle C_{0,4}A_{0,4}\rangle \mid \\
&\quad\ \langle A_{0,4}C_{0,4}\rangle \mid \langle B_{0,4}C_{0,4}\rangle \mid \langle C_{0,4}C_{0,4}\rangle \\
C_{0,4} &\rightarrow \text{causes}/f \mid \text{unrest}/f \mid \emptyset/f \mid S_{0,2}
\end{aligned}
$$

$$
\begin{aligned}
S_{0,2} &\rightarrow A_{0,2} \mid B_{0,2} \mid C_{0,2} \\
A_{0,2} &\rightarrow [A_{0,2}B_{0,2}] \mid [B_{0,2}B_{0,2}] \mid [C_{0,2}B_{0,2}] \mid \\
&\quad\ [A_{0,2}C_{0,2}] \mid [B_{0,2}C_{0,2}] \mid [C_{0,2}C_{0,2}] \\
B_{0,2} &\rightarrow \langle A_{0,2}A_{0,2}\rangle \mid \langle B_{0,2}A_{0,2}\rangle \mid \langle C_{0,2}A_{0,2}\rangle \mid \\
&\quad\ \langle A_{0,2}C_{0,2}\rangle \mid \langle B_{0,2}C_{0,2}\rangle \mid \langle C_{0,2}C_{0,2}\rangle \\
C_{0,2} &\rightarrow \text{the}/f \mid \text{tax}/f \mid \emptyset/f
\end{aligned}
$$

Grammar templates provide a conceptual framework to easily transfer grammars for flat

sentence pairs to situations with fixed phrasal structure. We have used the framework here to ensure only one structure is constructed for each possible alignment. This recursive view of the solution also makes it easier to visualize the space that the C-ITG is searching.

Implementation Although conceptually useful, the custom grammar described above can be cumbersome to implement in practice, especially when one considers the challenge of doing so without having it negatively affect the ITG's grammar constant with its large number of non-terminals and productions. However, if we examine the custom grammar and distill out its essential functionality, we can create the same effect of a canonical C-ITG with a small extension to our chart modification solution. The canonical grammar template maintains cohesion with the input dependency tree through two mechanisms:

1. Span-annotated S non-terminals force ITG constituents to be built over English spans corresponding to dependency subtrees.

2. C $\rightarrow$ S productions allow these dependency-induced (or forced) constituents to participate in productions as if they were pre-terminals. This means that the canonical-form rule that allows right recursion only in cases of alternating straight and inverted productions is relaxed for constituents that must be built to maintain cohesion.

Instead of implementing the custom grammar wholesale, we can implement these two mechanisms directly. Item 1 simply forces constituents to be built over dependency subtree spans. We already know this objective can also be achieved by modifying the chart with invalid spans; therefore, we re-introduce chart modification. Item 2 alters how forced constituents behave when appearing on the right-hand-side (RHS) of a production. If we mark the **required spans** corresponding to these forced constituents, we can alter their RHS behavior by assigning them alternate non-terminals. Instead of using A or B symbols, these required spans are over-ridden to be generated by A$'$ or B$'$. These new symbols stand in for the unary production chains C $\rightarrow$ S $\rightarrow$ A and C $\rightarrow$ S $\rightarrow$ B respectively. The canonical grammar in (2.11) is then augmented so that the alternate symbols avoid restrictions on right recursion: A$'$ and B$'$ behave as normal A and B symbols when on the left-hand-side of a production, but on the right-hand-side, they behave like C. All other spans either use the normal symbols or are invalid. The required spans can be easily marked using the same data structure used to mark invalid spans. The resulting canonical-form grammar uses only 2 new symbols, but its production count has increased significantly. These extra productions do not negatively impact the efficiency of our implementation, since any given span

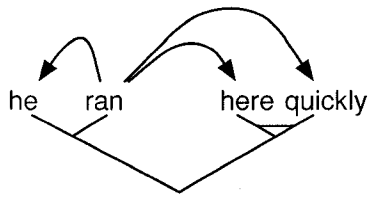

Figure 5.5: A counter-intuitive ITG structure.

can only be covered by either the original A and B symbols or the new ones, and never both. Given a span and a split point over which productions are being applied, the number of productions available is no greater than in the original (2.11).

5.2 Head-aware Dependency Match

Throughout our discussion of C-ITG, we have been careful to state that we are maintaining syntactic cohesion with respect to the provided dependency tree. This is because we have insisted only that the ITG match the syntactic phrases indicated by the dependency tree's subtrees, and not the head structure indicated by the directed dependencies. We introduce head pruning here, where the binary ITG constituents reflect head-modifier relationships in addition to syntactic cohesion. It was hoped that by constraining the ITG according to this additional syntactic information, we could provide further guidance to the alignment system. As our experiments will show, this is not the case. We describe the resulting head-constrained ITG or H-ITG in this section, but it will receive less detail than the C-ITG. These constraints are important in illustrating what phrasal cohesion does not do, which we discuss qualitatively below and quantitatively later in our experimental results.

Each dependency tree has an implied K-arry constituency tree (Gaifman, 1965) that discards head information, with flat constituents built over what we have referred to above as dependency subtree phrases. This K-arry constituency structure will have several possible binarizations. Syntactic cohesion allows the C-ITG to select any of these binarizations. However, some will not agree with the head-modifier relationships indicated by the original dependency tree. For example, the ITG constituency structure shown in Figure 5.5 respects syntactic cohesion, and will be considered by a C-ITG as it enumerates alignment space. The resulting "here quickly" subtree disagrees with the provided dependencies, which specify that "ran" is modified by "here" and "quickly" individually, and not by a phrasal concept that includes both. This is allowed because we have forced the ITG to respect the dependency tree's syntactic boundaries, but it has no notion of heads or modifiers.

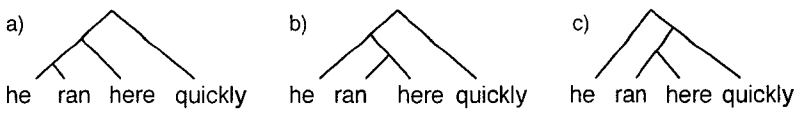

Figure 5.6: Structures allowed by the head constraint.

5.2.1 Chart Modification

The most straight-forward way to eliminate these modifier constituents is to parse with the redundant bracketing grammar in (2.10), and to add another set of invalid spans to the set described in § 5.1. The new invalidated chart entries eliminate all spans that include two or more modifiers without their head. With this solution, the structure in Figure 5.5 is no longer possible. Like chart modification for syntactic cohesion, this method allows multiple structures for each alignment; for example, to represent an alignment with no inversions, this head-aware H-ITG can produce any of the three structures shown in Figure 5.6.

5.2.2 Custom Grammar

Given a grammar that produces canonical head-aware structures for local dependency trees, we can easily extend it to complete dependency trees using the concept of grammar templates described in § 5.1. Such a grammar requires a notion of head, so we can ensure that every binary production involves the head or a phrase containing the head. A redundant, head-aware grammar is shown here:

$$\begin{aligned} A &\rightarrow [MA] \,|\, \langle MA \rangle \,|\, [AM] \,|\, \langle AM \rangle \,|\, H \\ M &\rightarrow \text{he}/f \,|\, \text{here}/f \,|\, \text{quickly}/f \\ H &\rightarrow \text{ran}/f \end{aligned} \tag{5.1}$$

Note that two modifiers can never be combined without also including the A symbol, which always contains the head. This grammar still considers all the structures shown in Figure 5.6, but it requires no chart preprocessing to account for head information.

We can create a canonical-form grammar by expanding (5.1). Inspired by (2.11), we restrict the productions so that the grammar has a default preference for recursion, broken only for alternating straight and inverted productions. To specify the necessary inversion combinations, our ITG will need more expressive non-terminals. Split A into two non-terminals, L and R, to produce left and right modifiers respectively. Then split L into $\bar{\text{L}}$ and $\hat{\text{L}}$, to indicate straight and inverted left modifiers. We now have a rich enough non-terminal set to design a grammar with a default behavior: it will generate all right modifiers deeper in the bracketing structure than all left modifiers. This rule is broken only in sequences that

alternate between straight and inverted productions. A grammar that accomplishes this goal is shown here:

$$
\begin{aligned}
&S \rightarrow \bar{\mathrm{L}} | \hat{\mathrm{L}} | \mathrm{R} \\
&\mathrm{R} \rightarrow \left[\hat{\mathrm{L}}\mathrm{M}\right] | \langle \bar{\mathrm{L}}\mathrm{M} \rangle | [\mathrm{RM}] | \langle \mathrm{RM} \rangle | \mathrm{H} \\
&\bar{\mathrm{L}} \rightarrow \left[\mathrm{M}\bar{\mathrm{L}}\right] | \left[\mathrm{M}\hat{\mathrm{L}}\right] | [\mathrm{MR}] \\
&\hat{\mathrm{L}} \rightarrow \langle \mathrm{M}\bar{\mathrm{L}} \rangle | \left\langle \mathrm{M}\hat{\mathrm{L}} \right\rangle | \langle \mathrm{MR} \rangle \\
&M \rightarrow \mathrm{he}/f \mid \mathrm{here}/f \mid \mathrm{quickly}/f \\
&H \rightarrow \mathrm{ran}/f
\end{aligned}
\tag{5.2}
$$

This grammar generates one structure for each alignment. In the case of an alignment with no inversions, it produces the tree shown in Figure 5.6c. A grammar to cover a complete dependency tree can be generated by using (5.2) as a template instantiated over all local trees of the dependency tree.

5.2.3 Alignment Space

Modifiers of the same head can no longer occur at the same level of any ITG tree, because we have limited how the ITG can combine modifiers. All three valid structures in Figure 5.6 have "quickly" attached higher in the tree than "here". As a result of this, no combination of inversions can bring "quickly" between "here" and "ran". In general, the alignment space enumerated by the H-ITG is constrained so that, among modifiers, relative distance from head is maintained. More formally, let M_i and M_o be modifiers of H such that M_i appears between M_o and H in the dependency tree. No alignment will ever place the outer modifier M_o between H and the inner modifier M_i. This constraint is in addition to syntactic cohesion, which is also maintained.

5.3 Alignment Space Experiments

In § 5.1 and 5.2, we introduce two syntactic constraints for ITGs, each based on a provided English dependency tree. The C-ITG adds a syntactic cohesion constraint to ITG processing: subtrees in the source tree must remain contiguous when projected onto the target string by the alignment. The H-ITG adds a head constraint: no constituent built by the ITG can contain two or more source modifiers without also including their head. These two constraints effectively create new alignment spaces. In particular, the C-ITG is an intersection of cohesion space and ITG space. These experiments, originally published in (Cherry and Lin, 2006a), are intended to assess these new spaces along with old ones on a common French-English alignment data-set. We compare a number of one-to-one alignment spaces

with varying degrees of syntax-limited concept re-ordering. First we test the **guidance** provided by a space, or its capacity to stop an aligner from selecting bad alignments. We also test **expressiveness**, or how often a space allows an aligner to select the correct alignment.

We have several goals for our alignment-space comparison. First, other groups have gone to great lengths to avoid ITG constraints (Melamed, 2003), and in our phrasal ITG we side-step this issue by deleting training pairs likely to exhibit ITG constraint violations (§ 4.2.2). We would like to assess the ITG constraints empirically to determine their effect on word alignment quality. We also want to compare our C-ITG, which combined ITG and cohesion spaces, to a cohesion constraint alone; our hypothesis is that the addition of ITG constraints should have a minimal effect on the already restrictive syntactic cohesion. If this were true, it would allow a cohesion-constrained aligner to benefit from the algorithmic advantages of ITG without suffering any major drawback. Finally, we want to understand the difference between our two syntactically constrained ITGs, to determine if there is a benefit to accounting for head data. Any difference between the C-ITG and H-ITG may help explain the problems observed in other dependency-based alignment approaches, such as those in (Lopez et al., 2002; Gildea, 2004).

5.3.1 Experimental Details

This section outlines the experimental materials, framework, and systems for our alignment space comparison. We begin by discussing the corpus used for these experiments. Since the same corpus is used in our discriminative experiments in § 5.5, we provide sufficient details here to avoid repetition later. We follow this with a description of the experimental framework used to evaluate our alignment spaces. Finally, we describe each alignment system that we have selected for comparison.

Canadian Hansards Corpus

The primary bitext used for reporting alignment results in the literature is a set drawn from the English-French Canadian Parliamentary Debates, or Hansards. This is our preferred corpus when evaluating alignment quality alone. We use a 50K sentence-pair subset of this corpus as unsupervised alignment data; this same subset was used for experiments in (Och and Ney, 2000b; Cherry and Lin, 2003). We find this training set is normally sufficient to conduct comparisons of alignment factors that are distinct from unsupervised training, such as this constraint analysis.

This Hansards data has a 500 sentence-pair word-aligned test set, provided to us by

Franz Och by personal communication. This test set was originally aligned by Och and Ney (2003), and makes use of both sure and possible links. A partitioning of this set, consisting of a 447 sentence-pair test set and a corresponding 37 sentence-pair development set was distributed as a part of the WPT 2003 alignment evaluation (Mihalcea and Pedersen, 2003). This partition is generally used for comparisons to the state of the art. For this alignment space comparison, we evaluate using the original 500-sentence test set. For our discriminative experiments in § 5.5, we follow Taskar et al. (2005), and use the first 100 aligned sentence pairs of the WPT split for training, saving the remaining 347 for testing.

Experimental Framework

In all cases, we report our results in terms of alignment quality, using the standard word alignment error metrics: precision, recall, F-measure and alignment error rate (§ 3.1). We use the Hansards manually-aligned data set as our gold standard. English dependency trees are supplied by Minipar (Lin, 1994).

In these experiments, we hold all variables constant except for the alignment space being searched, and in the case of imperfect searches, the search method. In particular, all of the methods maximize the same alignment scoring function to select an alignment from alignment space. The score of a proposed alignment A is:

$$f_{align}(A, \vec{e}, \vec{f}) = \sum_{a \in A} v(a, \vec{e}, \vec{f}) \qquad (5.3)$$

Note that this objective function evaluates each link a independently according to the link score v, which is unaware of the other links in A. Our two experiments will vary the definition of v to test different aspects of alignment spaces.

Comparison Systems

We test six methods. The first three are increasingly thorough searches of permutation space, the one-to-one alignment space that allows unconstrained concept movement. The remaining four systems explore distinct syntax-constrained spaces.

Greedy: A greedy search of permutation space. Links are added in order of descending link scores. This corresponds to the competitive linking algorithm (Melamed, 2000).

Beam: A beam search of permutation space, where links are added to a growing alignment, biased by their link scores. Beam width is 2 and agenda size is 40.

Match: The weighted bipartite matching algorithm (West, 2001). This is a perfect search of permutation space.

Method	Prec	Rec	F	AER
Greedy	78.1	81.4	79.5	20.47
Beam	79.1	82.7	80.7	19.32
Match	79.3	82.7	80.8	19.24
ITG	81.8	83.7	82.6	17.36
Dep	88.8	84.0	86.6	13.40
C-ITG	88.8	84.2	86.7	13.32
H-ITG	89.2	84.0	86.9	13.15

Table 5.1: Alignment space comparison with an unsupervised, learned link score.

ITG: The alignment resulting from ITG parsing with the canonical grammar in (2.11). This is a perfect search of ITG space.

Dep: A beam search of the dependency space. This is implemented as **Beam** with an additional syntactic cohesion constraint. It is equivalent to the system tested in (Lin and Cherry, 2003).

C-ITG: The result of ITG parsing with a syntactic cohesion constraint. This is a perfect search of the intersection of the ITG and dependency spaces, as described in § 5.1.

H-ITG: The result of ITG parsing with syntactic cohesion and an additional head matching constraint, as described in § 5.2.

5.3.2 Guidance Test

The link score v is usually imperfect, because it is learned from data. Appropriately defined alignment spaces may rule out bad links even if they are assigned high v scores, improving alignment quality by guiding the search to better alignments. We define the following simple link score to test the guidance provided by different alignment spaces:

$$v(a, \vec{e}, \vec{f}) = \phi^2(e_i, f_j) - K|i - j| \tag{5.4}$$

Here, $a = (e_i, f_j)$ is a link and $\phi^2(e_i, f_j)$ returns the ϕ^2 correlation metric (§ 2.1.2) between the linked words. The ϕ^2 scores were obtained using co-occurrence counts from our Hansards data. The second term is a small distortion penalty. K is a small constant selected to be just large enough to break ties in favor of similar positions. Links to $\emptyset$ are given a flat score of 0, while token pairs with no value in our ϕ^2 table are given a flat score of -1. We prune all bilingual pairs with a ϕ^2 score less than 0.0005 from our table.

The results of maximizing f_{align} on our test set are shown in Table 5.1. As one can see, alignment accuracy improves as the syntactic constraints grow more restrictive. The

ITG method scores an AER of 17.4, a 10% relative reduction in error rate from bipartite matching. This indicates that the constraints established by ITG space are beneficial, even before adding an outside parse. The three dependency-tree guided methods all have AERs of around 13.3. This is a 31% improvement over maximum matching. Note that enforcing head constraints in the **H-ITG** produces only a small improvement over the **C-ITG**. One should also note that, with the exception of the **H-ITG**, recall goes up as smaller spaces are searched. In a one-to-one alignment, enhancing precision can also enhance recall, as every error of commission avoided can present up to two new opportunities to avoid an error of omission.

The small gap between the beam search of permutation space and bipartite matching indicates that for this link score v, the beam search is a good approximation to complete enumeration of a space. This is important, as the only method we have available to search dependency space is also a beam search. Knowing this, we can conclude that combining the phrasal cohesion constraint with the ITG constraint in the **C-ITG** does not reduce alignment quality with respect to those alignments found using only the phrasal cohesion constraint in **Dep**. In fact, we appear to be benefiting slightly from either the combination of constraints or the complete search.

We can gain further insight into the differences between **Dep** and **C-ITG** by looking at their success in solving the search problem of finding the maximal f_{align}. For each sentence pair, we can compare the f_{align} values found, instead of error rates. In 35 sentence pairs, **C-ITG**'s perfect search allows it to find a higher f_{align}, even though it is searching a smaller space. **Dep** finds a higher f_{align} than the **C-ITG** only 3 times. Each instance is caused by a fairly flat dependency tree allowing non-ITG re-orderings in dependency space. This never results in the **Dep** method finding a correct link that was not found by **C-ITG**.

5.3.3 Expressiveness Test

Any time we limit an alignment space, we risk ruling out correct alignments. We now test the expressiveness of an alignment space according to the best alignments that can be found there when given an oracle link score. With these perfect link scores, the model no longer requires guidance, any limits placed on alignment can only hold the system back. This sort of test is similar to the experiments in (Fox, 2002), but instead of counting crossings, we count how many links an oracle alignment misses when confined to a particular space. These tests are also similar to those carried out concurrently in (Wellington et al., 2006b).

We create a tailored v for each sentence pair, based on the gold standard alignment for

Method	Rec	Missed	F	AER
Dep	94.1	260	97.0	3.02
H-ITG	94.2	258	97.0	3.00
C-ITG	94.8	232	97.3	2.69
ITG	96.3	165	98.1	1.90
Match	96.4	162	98.1	1.86

Table 5.2: Alignment space comparison with an oracle link score.

that pair. Gold standard links are broken up into two categories in Och and Ney's evaluation framework (2003). S links are used when the annotators agree and are certain, while links in $P - S$ are meant to handle ambiguity. Since only S links are used to calculate recall, we define our v to mirror the S links in the gold standard:

$$v(a, E, F) = \begin{cases} 1 & \text{if } a \text{ is an } S \text{ in } (E, F) \\ 0 & \text{if } a \text{ is a link to } \textit{null} \\ -1 & \text{otherwise} \end{cases}$$

Table 5.2 shows the results of maximizing summed v values in our various alignment spaces. The greedy and beam searches of permutation space have been omitted, as they are simply approximating bipartite matching. The precision column has also been omitted, as it is trivially 100 in all cases. A new column has been added to count missed links.

Bipartite matching sets the upper bound for this task, with a recall of 96.4. It does not achieve perfect recall due to the one-to-one constraint. Of the constrained systems, **ITG** fairs the best, showing only a tiny reduction in recall, due to 3 missed links throughout the entire test set. Considering the non-trivial amount of guidance provided by the **ITG** in the previous experiment, this small drop in expressiveness is quite impressive. For the most part, the ITG constraints appear to rule out only incorrect alignments. The **C-ITG** isn't doing much worse, but the noticeable drop in achievable recall may be a problem for some applications. It may be surprising to see **C-ITG** outperforming **Dep**, as the alignment space of **Dep** is larger than that of **C-ITG**. However, the incomplete nature of **Dep**'s beam search is holding it back.

The **H-ITG** makes 26 fewer correct links than the **C-ITG**, each corresponding to a single missed link in a different sentence pair. These misses occur in cases where two modifiers switch position with respect to their head during translation. Surprisingly, there are regularly occurring, systematic constructs that violate the head constraints. An example of such a construct is when an English noun has both adjective and noun modifiers. Cases like "Canadian Wheat Board" are translated as, "Board Canadian of Wheat",

switching the modifiers' relative positions. These switches correspond to discontinuous constituents (Melamed, 2003) in lexical bitext parsing. The **C-ITG** can handle these discontinuities by freely grouping constituents to create continuity, but the **H-ITG**, with its fixed head and modifiers, cannot. Given that the **H-ITG** provides only slightly more guidance than the **C-ITG**, we feel that its systematic elimination of simple, correct translation constructions makes it a poor constraint for use with ITGs.

Discussion

From our alignment space experiments, it appears that the ITG constraints actually place very few limits on expressiveness beyond those established by a one-to-one constraint. This increases our confidence in using the ITG as our base aligner. Furthermore, the additional cohesion constraints in the C-ITG have been shown to greatly increase alignment accuracy, at the cost of some expressiveness. This leaves at least two questions for investigation:

1. Can we incorporate the syntactic cohesion from the C-ITG as a soft constraint, to receive its benefits without necessarily suffering the reduction to expressiveness?
2. Our guidance experiments were conducted with a weak, ϕ^2-weighted aligner. Would the results still hold on a strong aligner with complex distortion features?

Discriminative training provides a principled method to implement a soft cohesion constraint inside a strong ITG aligner, allowing us to address both questions at once.

5.4 Discriminative ITG with Soft Cohesion Constraint

In this section we describe a discriminative training method for ITGs using SVM Struct (§ 2.5.4). To our knowledge, this is the first use of discriminative training for ITG word alignment. A discriminative training method will allow us to introduce syntactic cohesion as an alignment feature. This will create a discriminatively-weighted penalty for violating syntactic cohesion, creating a soft cohesion constraint.

There are, of course, several other benefits to discriminative training. First, as outlined in (§ 2.5), the linear models used in discriminative training allow arbitrary and overlapping features, which will allow us to incorporate a cohesion feature in addition to other powerful features from the literature. Second, we can train specifically to optimize alignment quality, as opposed to optimizing the likelihood of a sentence-aligned bitext. Finally, ITG algorithms can be slow, and iteratively training unsupervised models on hundreds of thousands

of sentence pairs can be time consuming. We prefer the smaller, 100 sentence-pair training sets used in discriminative training, which allow for much faster system development. We are interested in the discriminative training of ITGs, as opposed to another alignment inference engine, such as bipartite matching or the IBM models, because an ITG exposes syntactic features that are not available in these other aligners.

There are several viable options for the discriminative training of ITGs. Any online method, such as the averaged perceptron (Collins, 2002) or SVM Struct (Tsochantaridis et al., 2004), will work, because they require only the output of the alignment search under various weight vectors $\vec{w}$. Conditional random fields (Lafferty et al., 2001) are also possible, since we have access to the inside algorithm to efficiently calculate the log linear model's normalization term, but this solution would likely be prohibitively slow to train. We select SVM Struct with margin re-scaling because it has a well-founded, large-margin objective that allows a structured loss function. Furthermore, because of its modularity, we can also re-implement the discriminative matching system (Taskar et al., 2005) using SVM Struct in § 5.4.4, allowing us to compare our ITG system to a bipartite matching system, while leaving all variables regarding learning constant.

We describe our SVM Struct training framework for ITGs in § 5.4.1. Our feature set, including the syntactic cohesion feature, is described in § 5.4.2, while § 5.4.3 provides details necessary for the design of an effective gold standard for SVM-ITG training. Finally, in § 5.4.4 we outline our baseline discriminative aligner: an SVM Struct re-implementation of the discriminative matching system.

5.4.1 SVM Training

Having selected a learning method, to fully specify our SVM ITG, we must describe the three remaining components of discriminative training, as outlined in § 2.5. To build an alignment y over a sentence pair x, these components are:

1. A search to find $\text{struct}(x; \vec{w}) = \text{argmax}_{y \in \mathcal{Y}} \vec{w} \cdot \Psi(x, y)$
2. A feature representation for complete alignments $\Psi(x, y)$
3. A structured loss function $\Delta(y, \bar{y})$

Throughout this discussion we assume a one-to-one ITG, powered by the simple binary bracketing grammar in (2.10).

In our SVM-ITG, the alignment search for $\text{struct}(x; \vec{w})$ is a weighted ITG parser (§ 2.2.3). This requires us to slightly alter our interpretation of (x, y); x remains a sen-

tence pair, but y becomes an ITG derivation, which implies a word alignment through its terminal productions. Given a value[2] v for each production that can be applied during a derivation, the ITG parser finds the derivation $\bar{y}$ that yields x, and which maximizes the sum of production values:

$$\bar{y} = \text{argmax}_{\{y|yield(y)=x\}} \sum_{r \in y} v(r; x)$$

Note that the value of a production is parameterized not only on the production rule r, but also on the sentence pair x. This indicates that, in a departure from traditional parsing, the value of a rule will change depending on the context in which it is applied. In the next step, we define the feature representation $\Psi(x, y)$ and the production values $v(r; x)$ so that the ITG parser solves the argmax in item 1.

Since the parser reasons about productions, we define our features on instances of productions. That is, we factor the feature vector $\Psi(x, y)$ for an entire derivation y into component feature vectors $\psi(r; x)$ on instances of production rules:

$$\Psi(x, y) = \sum_{r \in y} \psi(r; x)$$

Given this definition of $\Psi(x, y)$, if the values of productions are set according to their weighted features:

$$v(r; x) = \vec{w} \cdot \psi(r; x),$$

then the derivation found by the ITG parser will be the derivation with the maximum weighted score, as specified in item 1. We describe the feature vectors $\psi(r; x)$ for individual productions in § 5.4.2.

Having specified a search and a feature representation, the last item on our list is the structured loss $\Delta(y, \bar{y})$. Recall that SVM Struct requires a **max cost** search that finds

$$\bar{y} = \text{argmax}_{y'} \left[\vec{w} \cdot \Psi(x, y') + \Delta(y, y') - \vec{w} \cdot \Psi(x, y)\right]$$

for a labeled training instance (x, y). Fortunately, our use of margin re-scaling allows us to factor a decomposable loss function into our original search framework (§ 2.5.4). For the sake of modularity and simplicity, we define our loss only in terms of the alignment implied by y, and not in terms of the entire derivation; so we temporarily go back to interpreting our structure y as a set of links. We select weighted Hamming loss over links as our $\Delta(y, \bar{y})$,

[2] We use the term **value** to disambiguate between the values on productions and the **weights** on features, even though the term weight is traditionally used for both.

because it is decomposable and it has been used previously in (Taskar et al., 2005):

$$\Delta(y, \bar{y}) = c_o|y - \bar{y}| + c_c|\bar{y} - y| \tag{5.5}$$

where c_o is a penalty for incurred for errors of omission, and c_c is a penalty for errors of commission. A loss-augmented ITG parser is used to conduct the max cost search. Terminal productions of the form A $\rightarrow e/f$, which correspond to one-to-one links, have their values augmented to incorporate structured loss. For a known correct derivation y, the value of each production with the form $r =$ A $\rightarrow e_j/f_k$ is augmented as follows:

$$v(r; x) = \vec{w} \cdot \psi(r; x) + \begin{cases} c_c & \text{if } (e_j, f_k) \notin y \\ -c_o & \text{if } (e_j, f_k) \in y \end{cases} \tag{5.6}$$

A weighted ITG parser that is given loss-augmented values for its productions will find the max cost structure for any labeled training instance (x, y).

5.4.2 Features

We divide our features on productions into two disjoint sets: terminal and non-terminal features. Terminal features measure various qualities of the link implied by a one-to-one terminal production A $\rightarrow e_j/f_k$. Non-terminal features indicate qualities of non-terminal productions, and qualities of the bitext spans covered by these productions. We describe each in turn below.

Terminal Features

For terminal productions r, we use a feature set in our vector $\psi(r; x)$ that is very similar to the set used by the discriminative matching system (Taskar et al., 2005) for its edge features. Each terminal feature measures a quality of a specific link. Terminal productions $r_\emptyset$ that correspond to unaligned tokens are given blank feature vectors $\psi(r_\emptyset) = \vec{0}$. For all other terminal productions, let the production under discussion be A $\rightarrow e_j/f_k$; its features are described below.

Lexical translation affinity: To measure the affinity of the bilingual word-pair (e_j, f_k) we use conditional link probability, as described in § 2.1.2:

$$cor(e_j, f_k) = \frac{link(e_j, f_k) - d}{cooc(e_j, f_k)}$$

The link counts $link(e, f)$ are obtained by automatically word-aligning a large sentence-aligned bitext with a version of our in-house discriminative matching system that uses the ϕ^2 correlation measure (§ 2.1.2) in place of conditional link probability. d is an absolute discount parameter like the one used in Moore's perceptron aligner (2005b).

Distortion: We include several features to capture the relationship between the indices of linked tokens; for example, in English-French, linked tokens tend to have similar indices. Our primary distortion feature compares length-normalized indices:

$$dist(e_j, f_k) = \text{abs}\left(\frac{j}{m} - \frac{k}{n}\right)$$

We also use various non-linear transformations of this feature: $dist^2$, $\sqrt{dist}$ and $dist \cdot cor$.

Orthographic: We include a number of features that account for the characters in the two words being linked. Three separate indicators fire if e_j and f_k are an exact character match, an exact match ignoring accents, or an exact match ignoring vowels. Another feature reports the longest common subsequence ratio, or LCSR (Melamed, 2000) of e_j and f_k. A final indicator fires only if both words have fewer than 3 characters; this feature is intended to be a penalty to counter the inflated LCSR scores that these short cases would obtain.

First-order approximation: The alignment HMM (§ 2.1.1) has shown that it is valuable to condition decisions regarding the current link based on the location of the link that came before it. We cannot easily account for these first-order relationships between links in our ITG. Instead, we include an approximation that measures the affinity of the following token-pair, hoping to capture the likelihood of the ITG linking that pair in a separate decision. The resulting feature is $cor(e_{j+1}, f_{k+1})$. Note that the non-terminal features described later will be able to capture some syntactic interactions between links, as well as a preference for monotonicity.

Other features: To capture the preference for common words to link to common words and uncommon to link to uncommon, we include a feature that is the difference in the log rank of e_j and f_k - where a type's rank is determined by its monolingual token frequency in a large sentence-aligned corpus. We also include lexical indicators for all pairings of the five most frequent types in each language, so the SVM can directly learn a weight for linking (or not linking) these common pairs. Finally, we include a bias feature, which is an indicator that is always on for all token pairs. By assigning this feature a negative weight, the aligner can set a threshold on the summed weight of all other features, below which the token pair will not be linked, as the aligner will opt for links to $\emptyset$ instead.

Non-terminal Features

Up until this point, there has not been much room for innovation in feature design. Terminal productions correspond to links, so there is no reason to deviate from the tested feature sets of other link-focused methods such as (Taskar et al., 2005; Moore, 2005b). However, with the adoption of the ITG formalism, we now have access to an entirely new layer of abstraction, as represented by non-terminal productions and their corresponding constituents. This layer ties link decisions together in interesting ways, and allows us to model syntactic interactions between links within a complete search.

ITG Distortion: This feature indicates an inverted production $A \rightarrow \langle AA \rangle$. This allows the SVM to assign a penalty or bonus to deviations from a monotonic translation. It is unclear how much stronger or weaker this is than the first-order relationships modeled by an HMM. A single ITG inversion can create huge deviations in word order when inverting large enough constituents, but it also models the phenomenon of concepts moving together during translation more effectively than an HMM.

Invalid Spans: Recall that each production is aware not only of the rule being applied, but the context in which it is being applied in x. In particular, for any constituent, we can always find the bitext span (e_s^t, f_u^v) over which it is about to be built. It is easy to add an indicator that fires if that span would have been pruned by any ITG pruning method, such as tic-tac-toe or fixed-link pruning. This allows us to build soft constraints.

We include an indicator that fires whenever the ITG attempts to build a constituent over an English span that is invalid according to the phrasal structure of a fixed English dependency tree (§ 5.1). That is, if a span-pruning C-ITG would not allow the constituent, then the indicator will fire. As we show in § 5.3, constraining an alignment to maintain phrasal cohesion with a monolingual dependency tree provides a lot of guidance to an aligner, but also limits its ability to reproduce human alignments. A soft cohesion constraint, or cohesion penalty, could provide the same guidance, without necessarily holding the aligner back. This feature ties links together, as cohesion violations in alignment are inherently multi-link relationships. This sort of distortion information is orthogonal to the information we can provide through terminal features.

5.4.3 Gold Standard Design

SVM Struct requires complete Ψ vectors for its gold standard training structures. Unfortunately, our training set contains gold standard alignments, not ITG parse trees. Also, the gold standard may be divided into sure and possible link sets S and P (§ 3.1). Recall that all links in the gold standard are in P. Links in S must be included in a correct alignment, while links in $P - S$ can be included, but are not necessarily required for a correct alignment. That is, the aligner is never penalized for leaving out or including a link in $P - S$. Both the lack of gold standard ITG trees and the $P - S$ issue require special handling.

We create "gold standard" ITG trees using a modified weighted ITG that has its link values defined in terms of the known gold standard alignment. The following sorted priorities are used during tree construction:

- maximize the number of links from S
- minimize the number of English dependency span violations
- maximize the number of links from $P - S$
- minimize the number of inversions

This creates trees that represent high-scoring alignments, using a minimal number of invalid spans. Since we already have gold-standard alignments, only the span and inversion counts of these trees are used to construct feature vectors for training. Therefore, we need not achieve a perfect tree structure. We continue to evaluate all methods with the original alignment gold standard.

During training, we do not wish to penalize the SVM for missing links in $P - S$, as the aligner will not be penalized for these mistakes in the final evaluation. Therefore, we have altered SVM Struct so that, during training, any of these links that may be generated by the alignment search are removed before the construction of the complete feature vector Ψ, whether it be a vector for the correct alignment y or a candidate $\bar{y}$. Since the features for these links are never included in any vector, these links are essentially free: they are in the gold standard when the aligner includes them, and absent from the gold standard when the aligner leaves them out. For the loss-augmented search, we consider all links in $P - S$ to be special cases, their corresponding terminal productions do not have their values augmented in any way, because these links do not affect cost.

5.4.4 Re-implementation of Discriminative Matching

To provide a baseline for our discriminative ITG, we have re-implemented the discriminative matching system (§ 2.5.2) within SVM Struct. Due to SVM Struct's modularity, we are able to re-use many of the components from our SVM-ITG, including all link features. This means that the only differences between this baseline and our SVM-ITG are the ITG's syntactic alignment search and its unique non-terminal features. Recall that SVM Struct requires that its user provide an alignment search, a feature representation, and a structured loss function.

To re-implement discriminative matching, we use the weighted bipartite matching algorithm for our alignment search. Since SVM Struct does not tie us to the linear programming formulation of bipartite matching, we employ the Hungarian Method. Our implementation follows the pseudo-code in (Papadimitriou and Steiglitz, 1998, pg. 251). Like in the original discriminative matching system described in (Taskar et al., 2005), features are decomposed over links. Links are given the same features they would receive in our discriminative ITG; that is, each potential link (e_j, f_k) is given a feature vector ψ according to its implied terminal production: $\psi(e_j, f_k; x) = \psi(\mathrm{A} \rightarrow e_j/f_k; x)$. All non-terminal features from our discriminative ITG are discarded; they have no analog in bipartite matching. Finally, we use the same Hamming loss as we use for the discriminative ITG, since we were careful to define that loss only over links and not trees. This allows us to conduct our loss-augmented search through a loss-augmented matching algorithm, where each potential link has its value augmented according to the loss it will incur under the gold standard y. Because we are using the same Hamming loss with margin re-scaling, and because we have deliberately employed a similar feature representation, we believe that this is a faithful reproduction of the work by Taskar et al. (2005).

5.5 Discriminative Experiments

We have shown how an ITG bitext parser can be trained discriminatively using SVM Struct. This supervised paradigm has the potential to greatly increase the performance of the unsupervised, correlation-weighted ITGs tested in § 5.3, at the cost of a small amount of labeled training data. We have three primary goals in the following evaluation of our SVM-ITG, originally published in (Cherry and Lin, 2006b). First, we wish to confirm that the proposed SVM Struct alignment training does work, and our re-implementation of discriminative matching can match the performance reported in (Taskar et al., 2005). Second, we

Method	Prec	Rec	AER
Match	91.6	86.0	11.0
C-ITG	94.0	85.4	10.0
SVM-ITG	94.4	87.8	8.6

Table 5.3: Performance of SVM aligners with various degrees of cohesion constraint.

wish to assess the value of our cohesion feature, in particular, we want to compare it to a hard cohesion constraint. Finally, Taskar et al. required features that encode the already strong alignments found by GIZA++ in order to achieve an AER lower than 10.7 on our test set. We hope to show improvement from the same point, but by using syntactic information from a source dependency tree.

5.5.1 Experimental Details

We conduct our experiments using our English-French Hansard data from § 5.3.1. Our unsupervised ϕ^2 scores, link probabilities and word frequency counts are determined using 50K sentence-aligned bitext. Our supervised training set for the discriminative aligners is the first 100 sentence pairs from the WPT competition split of the word-aligned gold standard. For evaluation we compare to the remaining 347 gold standard pairs using the alignment evaluation metrics: precision, recall and alignment error rate or AER (§ 3). SVM learning parameters are tuned manually using the 37-pair development set. English dependency trees are provided by Minipar (Lin, 1994).

We test against two strong baselines. The first baseline, **Match** is our re-implementation of discriminative matching. The second baseline, **C-ITG** is an ITG aligner with hard cohesion constraints, but which uses the weights trained by **Match** to assign its link values. This is the most straight-forward way to combine discriminative training with the hard syntactic constraints.

5.5.2 Alignment Evaluation

The results are shown in Table 5.3. One should note that our **Match** baseline is achieving scores in line with (Taskar et al., 2005), which reports an AER of 10.7 using similar features and the same training and test sets. One can also see that the effect of the hard cohesion constraint has been greatly diminished after discriminative training. In our previous experiments using a simple, unsupervised link score, we were seeing a 30% relative error reduction in error rate by switching from **Match** to **C-ITG**. Now, with a far more

informative, learned v, the **C-ITG** produces only a 9% reduction from 11.0 to 10.0.

The soft-constrained **SVM-ITG** is faring substantially better. Its AER of 8.6 represents a 22% relative error reduction compared to the matching system. The improved error rate is caused by gains in both precision and recall. This indicates that the invalid span feature is doing more than just ruling out links; perhaps it is de-emphasizing another, less accurate feature's role. The **SVM-ITG** overrides the cohesion constraint in only 41 of the 347 test sentences, so we can see that it is indeed a soft constraint: it is obeyed nearly all the time, but it can be broken when necessary. The **SVM-ITG** achieves the strongest ITG alignment result reported on this data set; surpassing the 16 AER reported in (Zhang and Gildea, 2004), where a one-to-one ITG was trained unsupervised using inside-outside.

5.6 Summary

We have presented two new alignment spaces based on constraining an ITG with a fixed, English dependency tree. We have given grammars to conduct a perfect search of these spaces using an ITG parser. The grammars derive exactly one structure for each alignment. We have shown that syntactic constraints can have a very positive effect on alignment quality. With a learned objective function, ITG constraints reduce bipartite matching's error rate by 10%, while C-ITG constraints produce a 31% reduction. This gap in error rate demonstrates that a dependency tree over the English sentence can be a very powerful tool when making alignment decisions. We have shown that while cohesion constraints might limit alignment expressiveness too much for some tasks, enforcing ITG constraints results in almost no reduction in achievable recall.

We have also presented a discriminative, syntactic word alignment method. Discriminative training is conducted using the highly modular SVM Struct, which allows code reuse between the syntactic aligner and a bipartite matching baseline. An ITG parser is used for the alignment search, exposing two syntactic features: the use of inverted productions, and the use of spans that would not be available in C-ITG. This second feature creates a soft syntactic cohesion constraint. Discriminative training allows us to maintain all of the features that are useful to the bipartite matching baseline in addition to the new syntactic features. We have shown that these features produce a 22% relative reduction in error rate with respect to a strong flat-string model. This work demonstrates that a cohesion constraint can be useful to a strong aligner, when incorporated as a feature.

Chapter 6

Cohesive Phrasal Decoding

We have demonstrated that syntactic cohesion can increase word alignment accuracy by limiting the concept movement that can be displayed in an alignment. Syntactic cohesion is also implicit to some degree in several types of complete translation systems. The tree-to-string decoder (§ 2.3.3) represents movement during translation as a permutation of children, and hence has to obey cohesion completely. String-to-tree transducers (Graehl and Knight, 2004), tree-to-string transducers (Huang et al., 2006), and dependency treelet systems (Quirk et al., 2005) tend to produce cohesive translations, but their multi-level tree operations can overcome the limitations of cohesion by applying multi-level re-orderings observed in the training data. However, all of these examples have completely embraced a syntax-based model of translation: phrases are defined in the tree, and the models benefit from syntactic generalizations and tree-based movement models. This is all in addition to any implicit notion of syntactic cohesion. But how much benefit does the concept of cohesion provide to a decoder when taken alone? Can it provide any assistance to the still-dominant phrase-based translation models (§ 2.6.1) such as Moses or Pharaoh?

Our goal in this chapter is to improve the output of a left-to-right, phrase-based SMT system by making it aware of syntactic cohesion, according to a provided source dependency tree. Since we have an English dependency parser, we assume English to French translation. Previous work has shown that restricting the phrases used in phrasal SMT to syntactic constituents is harmful (Koehn et al., 2003); therefore, we are careful to add syntactic cohesion in a manner that still allows the use of arbitrary contiguous phrases. The dependency tree will only constrain the movement that can occur during translation. We hope that by respecting a known syntactic structure, we can improve translation quality.

We begin in § 6.1 by defining a cohesion violation in the context of phrasal SMT output. We also propose a K-best filtering approach to cohesive phrasal translation. In § 6.2, we

provide a linear-time algorithm to check for cohesion violations within a standard phrasal decoder. We present both a hard cohesion constraint as well as a soft cohesion feature for left-to-right phrase-based decoding. We thoroughly test both our cohesive output filter and our cohesive decoder in § 6.3.

6.1 Cohesive Phrasal Output

Previous approaches to measuring or checking the cohesion of a sentence pair have worked with a word alignment (Fox, 2002; Lin and Cherry, 2003). This alignment is used to project the spans of subtrees from the source tree onto the target sentence. If a modifier and its head, or two modifiers of the same head, have overlapping spans in the projection, then this indicates a cohesion violation (§ 2.3.3). To check phrasal translations for cohesion violations, we need a way to project the source tree onto the decoder's output.

Standard phrasal decoders work by repeating the following steps inside a beam search:

1. Select an unused phrase from the source sentence, and mark it as used.
2. Translate the source phrase into the target language using a phrase table.
3. Place the translated phrase at the end of the current hypothesis.

Each phrase used to create the target sentence can be tracked back to its original source phrase, providing an alignment between source and target phrases. Since each source token belongs to exactly one of the source phrases used during translation, we can transform this phrasal alignment into a word-to-phrase alignment, where each source token is linked to a target phrase. We can then project the source subtree spans onto the target phrase sequence. Note that we never consider individual tokens on the target side, as their connection to the source tree is obscured by the phrasal abstraction that occurred during translation.

Let e_1^m be the input source sentence, and $\bar{f}_1^p$ be the output target phrase sequence. Our word-to-phrase alignment $a_i \in [1, p], 1 \leq i \leq m$ maps a source token position i to a target phrase position a_i. Next, we introduce our source dependency tree T. Each source token e_i is also a node in T. We define $T(e_i)$ to be the subtree of T rooted at e_i. With this notation in place, we can define our projected spans. Following (Lin and Cherry, 2003), we define a head span to be the projection of a single token e_i onto the target phrase sequence:

$$spanH(e_i, T, a_1^m) = [a_i, a_i]$$

and the subtree span to be the projection of a subtree rooted at e_i:

$$spanS(e_i, T, a_1^m) = \left[\min_{\{j|e_j \in T(e_i)\}} a_j, \max_{\{k|e_k \in T(e_i)\}} a_k \right]$$

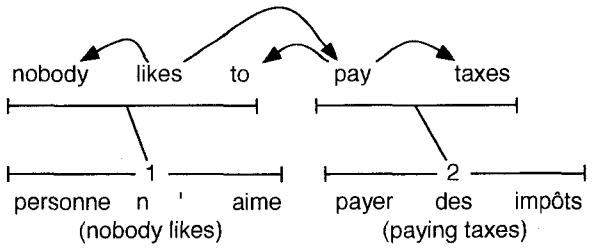

Figure 6.1: An example English source tree with translated French output. Segments are indicated with underlined spans.

Take for example, the simple phrasal translation shown in Figure 6.1 along with the provided English dependency tree. If we examine the local tree rooted at *likes*, we get the following projected spans:

$$\begin{aligned} spanS(nobody, T, a) &= [1, 1] \\ spanH(likes, T, a) &= [1, 1] \\ spanS(pay, T, a) &= [1, 2] \end{aligned}$$

For any local tree, we always consider only the head span of the head, and the subtree spans of any modifiers.

Typically, cohesion would be determined by checking these projected spans for intersection. However, one can see from our example that, at this level of resolution, avoiding intersection becomes highly restrictive. The simple monotone translation in Figure 6.1 would become non-cohesive: *nobody* intersects with both its sibling *pay* and with its head *likes* at phrase index 1. This complication stems from the use of multi-word phrases that do not correspond to syntactic constituents. We want to maintain the use of arbitrary phrases, so we tighten our definition of a violation to disregard cases where the sole point of overlap is obscured by our phrasal resolution. To do so, we replace span intersection with a new notion of span **innersection**.

Assume we have two spans $[u, v]$ and $[x, y]$ that have been sorted so that $[u, v] \leq [x, y]$ lexicographically. We say that the two spans **innersect** if and only if $x < v$. So, $[1, 3]$ and $[2, 4]$ innersect, while $[1, 3]$ and $[3, 4]$ do not. One can think of innersection as intersection, minus the cases where the two spans share only a single boundary point, where $x = v$. When two projected spans innersect, it indicates that the second syntactic constituent must begin before the first ends. If the two spans in question correspond to nodes in the same local tree, innersection indicates an unambiguous cohesion violation. Under this definition, the translation in Figure 6.1 is cohesive. Though the spans of the local tree rooted at *likes*

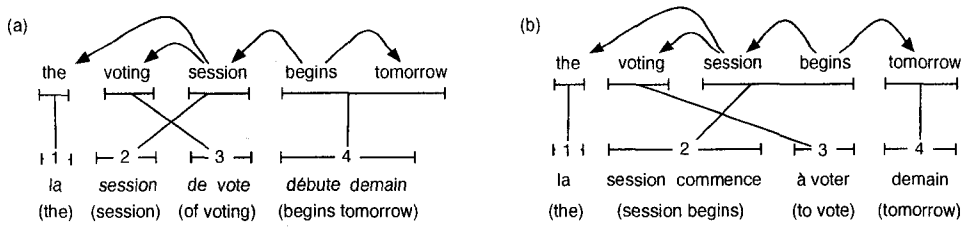

Figure 6.2: Two candidate translations for the same parsed source, along with their phrase-based decoding trace. (a) is cohesive, while (b) is not.

Span	(a)	(b)
$spanS(session, T, a)$	[1,3]	[1,3]*
$spanH(begins, T, a)$	[4,4]	[2,2]*
$spanS(tomorrow, T, a)$	[4,4]	[4,4]

Table 6.1: Spans of the local trees rooted at *begins* from Figures 6.2 (a) and (b). Innersection is marked with a "*".

all intersect, none of them innersect; they share exactly one point, which is a boundary point for all three spans.

Our hope is that syntactic cohesion will help the decoder make smarter distortion decisions. An example with distortion is shown in Figure 6.2. In this case, we present two candidate French translations of an English sentence, assuming there is no entry in the phrase table for "voting session." Because the proper French construction is "session of voting", the decoder has to move *voting* after *session* using a distortion operation. Figure 6.2 shows two methods to do so, each using an equal numbers of phrases, but employing different distortions. The projected spans for the local tree rooted at *begins* in each candidate are shown in Table 6.1. Note the innersection between the head *begins* and its modifier *session* in (b). Thus, a cohesion-aware system would receive extra guidance to select (a), which maintains the original meaning of the source sentence much better than (b).

K-best List Filtering

We now have a functional definition of syntactic cohesion as it applies to phrasal SMT output. It is an approximation to true measurements of cohesion, but it is also the best we can do at the phrasal level of resolution provided by a phrasal decoder. A first attempt at using cohesion to improve SMT output would be to apply it as a filter on K-best lists. That is, we could have a standard phrasal decoder output a 1000-best list along with a trace that indicates the generator for each phrase in the translation. We could then use a

monolingual parser and the innersection-based definition to check for syntactic cohesion. The highest-ranked cohesive translation would be returned to the user. This K-best post-processing approach is similar to the use of syntactic models in (Och et al., 2004), but where Och et al. re-ranked according to probabilities from complex statistical models of syntactic translation, we employ a binary notion of cohesion. We test this approach in § 6.3.2.

6.2 Cohesion Constraint for Phrasal Decoding

As our K-best list experiments in § 6.3.2 show, post-processing is not powerful enough to take full advantage of syntactic cohesion. The majority of the non-cohesive translations produced by a phrasal decoder do not contain a cohesive alternative within their 1000-best list. Those cases cannot be helped by K-best list filtering. Other groups have improved performance by pushing features that formerly used K-best list post-processing deep into the decoder (Zhang et al., 2006).

This section describes a modification that can be made to standard left-to-right phrase-based decoding, so that the system is constrained to produce only cohesive output. This will take the form of a check performed each time a phrase is added to the current hypothesis, similar to the ITG constraint for phrasal decoding (Zens et al., 2004). We first provide intuition, and then present the check algorithmically.

Our post-processing check for cohesion violations operates on a tree with projected spans. These spans can be calculated easily given a complete word-to-phrase alignment a_1^m, allowing us to check each local tree for innersection. However, during translation, the decoder has a different perspective. It builds translations incrementally from left to right. Each operation adds a target phrase to the end of the current hypothesis. We want to detect violations early, so the decoder can search the cohesion-constrained translation space as thoroughly as possible. What does a cohesion violation look like from this perspective?

Formal definitions aside, our goal in maintaining syntactic cohesion is to ensure that subtrees in the source tree remain contiguous in the translation. We can take advantage of the decoder's left-to-right construction of the target to implement this idea directly. Once the decoder starts translating part of a source subtree $T(r)$, if any later step translates some $e' \notin T(r)$ before all of $T(r)$ is translated, then we know that $T(r)$ will be finished at some point further to the right, and we know its translation will no longer be contiguous. We say that $T(r)$ is **interrupted** by e'. Any hypothesis containing such an interruption will invalidated and aborted.

For example, in Figure 6.2b, the decoder translates a part of the subtree rooted at *session* in $\bar{f}_1$. In $\bar{f}_2$, it translates *begins*, which is outside $T(session)$. Since we have yet to cover *voting*, we know that the projected span of *session* will end at some index $v > 2$. This interrupts $T(session)$, which eliminates the hypothesis after having proposed only the first two phrases of the translation.

In Appendix B, we show that an interruption is equivalent to a cohesion violation. Therefore, eliminating interruptions from the search space is equivalent to searching the space of cohesive translations. We formalize the notion of an interruption and provide an algorithm for their detection in the next section.

6.2.1 Algorithm

In this section, we formally define an interruption, and present an algorithm to detect one during decoding. During both discussions, we represent each target phrase as a set that contains the English tokens used in its translation: $\bar{f}_j = \{e_i | a_i = j\}$. Formally, an interruption occurs whenever the decoder would add a phrase $\bar{f}_{h+1}$ to the hypothesis $\bar{f}_1^h$, and:

$$
\begin{aligned}
&\exists r \in T \quad \text{such that:} \\
&\qquad \exists e \in T(r) \quad \text{such that } e \in \bar{f}_1^h \quad \text{(a. Started)} \\
&\qquad \exists e' \notin T(r) \quad \text{such that } e' \in \bar{f}_{h+1} \quad \text{(b. Interrupted)} \\
&\qquad \exists e'' \in T(r) \quad \text{such that } e'' \notin \bar{f}_1^{h+1} \quad \text{(c. Not finished)}
\end{aligned}
\tag{6.1}
$$

Thus, an interruption check should be performed before each proposed extension of $\bar{f}_1^h$ with $\bar{f}_{h+1}$. The key to performing such a check quickly is knowing which subtrees to check for completeness. Considered naïvely, we might need to check every subtree that has begun in $\bar{f}_1^h$. For example, Figure 6.3a highlights the roots of all such subtrees for a hypothetical T and $\bar{f}_1^h$. Fortunately, with a little analysis that accounts for $\bar{f}_{h+1}$, we can show that at most two subtrees need to be checked.

For a given interruption-free $\bar{f}_1^h$, we call subtrees that have begun translation in $\bar{f}_1^h$, but are not yet complete, **open** subtrees. Only open subtrees can lead to interruptions. We can focus our interruption check on $\bar{f}_h$, the last phrase in $\bar{f}_1^h$. We do so because any open subtree $T(r)$ must contain at least one $e \in \bar{f}_h$. If this was not the case, then the open $T(r)$ must have began translation somewhere in $\bar{f}_1^{h-1}$, and $T(r)$ would be interrupted by the placement of $\bar{f}_h$. Since our hypothesis $\bar{f}_1^h$ is interruption-free, we know this is impossible. This leaves the subtrees highlighted in Figure 6.3b to be checked. Furthermore, we need only consider

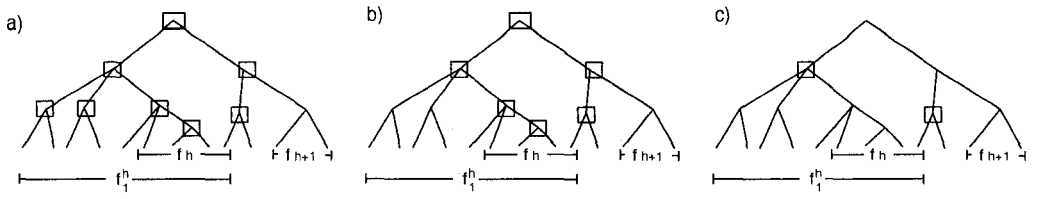

Figure 6.3: Narrowing down the source subtrees to be checked for completeness.

Algorithm 4 An algorithm to check for interruptions.

- Get the left and right-most tokens used to create $\bar{f}_h$, call them e_L and e_R
- For each of $e \in \{e_L, e_R\}$:
  - i. $r' \leftarrow e, r \leftarrow null$
    While $\exists e' \in \bar{f}_{h+1}$ such that $e' \notin T(r')$:
    $r' \leftarrow parent(r), r \leftarrow r'$
  - ii. If $r \neq null$ and $\exists e'' \in T(r)$ such that $e'' \notin \bar{f}_1^{h+1}$, then $\bar{f}_{h+1}$ interrupts $T(r)$.

subtrees that contain the left and right-most source tokens e_L and e_R translated by $\bar{f}_h$. Since $\bar{f}_h$ was created from a contiguous string of source tokens, any distinct subtree between these two endpoints will be completed within $\bar{f}_h$. Finally, for each of these focus points e_L and e_R, only the highest containing subtree $T(r)$ that does not completely contain $\bar{f}_{h+1}$ needs to be considered. Anything higher would contain all of $\bar{f}_{h+1}$, and would not satisfy requirement (6.1:b) of our interruption definition. Any lower subtree would be a descendant of r, and therefore the check for the lower subtree is subsumed by the check for $T(r)$. This leaves only two subtrees, highlighted in our running example in Figure 6.3c.

With this analysis in place, an extension $\bar{f}_{h+1}$ of the hypothesis $\bar{f}_1^h$ can be checked for interruptions with a straight-forward linear-time algorithm shown in Algorithm 4. Step (i) in this algorithm finds an ancestor r' such that $T(r')$ completely contains $\bar{f}_{h+1}$, and then returns r, the highest node that **does not** contain $\bar{f}_{h+1}$. We know this r satisfies requirements (6.1:a,b). If there is no $T(r)$ that does not contain $\bar{f}_{h+1}$, then e and its ancestors cannot lead to an interruption. Step (ii) then checks the coverage vector of the hypothesis[1] to make sure that $T(r)$ is covered in $\bar{f}_1^{h+1}$. If $T(r)$ is not complete in $\bar{f}_1^{h+1}$, then that satisfies requirement (6.1:c), which means an interruption has occurred.

For example, in Figure 6.2b, our first interruption occurs as we add $\bar{f}_{h+1} = \bar{f}_2$ to $\bar{f}_1^h = \bar{f}_1^1$. The detection algorithm would first get the left and right boundaries of $\bar{f}_1$; in

[1]This coverage vector is maintained by all phrasal decoders to track how much of the source sentence has been covered by the current partial translation, and to ensure that the same token is not translated twice.

this case, *the* is both e_L and e_R. Then, it would climb up the tree from *the* until it reached $r' = \textit{begins}$ and $r = \textit{session}$. It would then check $T(\textit{session})$ for coverage in $\bar{f}_1^2$. Since $\textit{voting} \in T(\textit{session})$ is not covered in $\bar{f}_1^2$, it would detect an interruption.

Walking up the tree takes at most linear time, and each check to see if $T(r)$ contains all of $\bar{f}_{h+1}$ can be performed in constant time, provided the source spans of each subtree have been precomputed. Checking to see if all of $T(r)$ has been covered in Step (ii) takes at most linear time. This makes the entire process linear in the size of the source sentence.

Interactions with Distortion Limits

A distortion limit constrains the amount of movment the decoder can perform. It does so by limiting how far the decoder can progress, moving from left to right through e_1^m, after it has skipped over some e_i, leaving it to be translated later. With a distortion limit d, the decoder can translate up to e_{i+d} before it has to double back and cover e_i. This limit has been shown to be essential for high quality translations (Koehn et al., 2005). However, cohesion forces the decoder to work on a subtree until it is finished. If that subtree extends past the distortion limit without covering e_i, then this results in a dead-end: a state which takes up space in our priority queue, but does not lead to any complete, legal translation. Should the queue fill up with these dead-end states, then the decoder will abort the translation.

Fortunately, a dead-end check is also linear, and re-uses much of the code from our interruption check. Starting from the right-most source word covered in $\bar{f}_{h+1}$, we find its highest source ancestor r such that $T(r)$ does not contain the left-most skipped e_i. If r exists, and its right boundary extends past the distortion limit $i + d$, then we invalidate the hypothesis as a dead-end.

6.2.2 Soft Constraint

Unfortunately, syntactic cohesion is not a perfect constraint for translation. Parse errors and systematic violations can create cases where cohesion works against the decoder. Fox (2002) demonstrated and counted cases where cohesion was not maintained in a hand-aligned sentence-pair. In this thesis, we show that a soft cohesion constraint is superior to a hard constraint for word alignment (§ 5.5).

Therefore, we also test a soft version of our interruption-based cohesion constraint. In this version, we perform our interruption check, but do not invalidate any hypotheses. Instead, each hypothesis maintains a count of the number of extensions that have caused interruptions during its construction. This count becomes a feature in the decoder's log-

linear model, the weight of which is trained with minimum error rate training. This count has a meaningful value at each stage of decoding, allowing it to be used in the evaluation of partial hypotheses.

Of course, after the first interruption, all of our theory and proofs are invalidated, making the exact meaning of further interruptions difficult to interpret. But, the number of interruptions does provide a useful estimate of the extent to which the translate is faithful to the source tree structure. Our experiments show that this is valuable in practice.

6.2.3 Implementation

We modify the Moses decoder to translate either sentences or head-annotated sentences. When the decoder detects a head-annotated sentence, it stores the flat sentence in the original sentence data structure, and the encoded dependency tree in an attached tree data structure. The tree structure caches the source spans corresponding to each of its subtrees. We then implement both a hard check inside the decoder for interruptions, a dead-end check to be used in conjunction with the hard constraint, and a soft check for interruptions that is used to calculate an interruption count feature.

6.3 Experiments

We have successfully adapted the notion of syntactic cohesion so that it is applicable to the phrase-based decoding methods described in § 2.6.1. This results in a translation process that can directly benefit from a source dependency-tree by respecting source-side syntactic boundaries during translation. We have provided two methods to take advantage of this modified cohesion, the first being an K-best cohesion filter (§ 6.1), and the second being a cohesion constraint on decoding that works by eliminating or penalizing interruptions (§ 6.2). This section tests both of these approaches in order to determine if our definition of phrase-based cohesion has a positive impact on translation quality.

6.3.1 Experimental Details

We test our various cohesion-modified decoders trained using our complete, 688K Europarl French-English data, as described in § 4.4.1. Since we require source dependency trees, we reverse our standard translation direction; all experiments test English to French translation, unless stated otherwise. English dependency trees are provided by Minipar (Lin, 1994). We use our baseline translation infrastructure (§ 4.4.1) with the Moses decoder, altering or

System	Dev BLEU
Baseline	32.04
Cohesion Filtered	32.08

Table 6.2: BLEU score comparison of K-best list cohesion filter.

filtering the decoder as is appropriate. Results are reported with the BLEU score on both our 1500-pair development set and our 2000-pair test set.

6.3.2 K-best list filtering

Our preliminary experiments test the K-best filtering approach described in § 6.1. We run our baseline Moses decoder on our English to French development set, and generate 1000-best translation lists. We then process the lists, checking for syntactic cohesion violations. The highest-ranked cohesive translation is returned as the final output. Lacking a cohesive translation in the 1000-best list, we output the original 1-best translation. This process is compared to the baseline, unfiltered decoder.

The results of our filter experiment are shown in Table 6.2. The improvement in BLEU score is negligible, but there is a fair amount to be learned by examining Moses' output in terms of cohesion. 330 out of the 1500 development sentences produce non-cohesive output as their 1-best translation; therefore, we could potentially improve roughly 1/5 of our translations with our cohesion constraint. Unfortunately, only 124 of these cases have a cohesive option in their 1000-best list, leaving the remaining cases out of reach. This cuts our potential pool of improvement down to 1/15 of the sentences. We observe the same ratios of cohesion violations and cohesive 1000-best options in our tuning data as well.

However, examining the small remaining area of potential improvement more closely, we can see the positive effects of cohesion. The added syntactic knowledge helps preserve meaning by keeping syntactic constituents together after translation. Furthermore, it does so in ways that often do not affect BLEU score. Two example translations before and after filtering are shown in Table 6.3.

The first example typifies one observed form of improvement: both versions of Moses select the same segmentation for the source sentence, but the cohesion constraint helps place these segments more intelligently, over-riding language model preferences. In this case, the segmentation is "[receive] [significantly larger] [sums of money]". The baseline system prefers to move *receive* between the two other segments. The cohesion constraint prevents this movement, creating a more readable, though still awkward translation. However, the

(1)	receive significantly larger sums of money ...
Ref	des sommes beaucoup plus importantes ont en effet été rendues aux ... *much more important sums were in effect given to ...*
Base	considérablement plus recevoir des sommes d ' argent ... ***considerably more receive** sums of money ...*
Filt	reçoivent considérablement plus des sommes d ' argent ... ***receive considerably more** sums of money ...*
(2)	creating structures that do not currently exist and reducing ...
Ref	l ' établissement de structures aujourd ' hui inexistantes et à la réduction ... *the establishment of structures that are non-existent today and the reduction ...*
Base	de créer des structures qui existent actuellement et ne pas réduire ... *to create structures that **actually exist** and **do not reduce** ...*
Filt	de créer des structures qui n ' existent pas encore et réduire ... *to create structures that **do not yet exist** and **reduce** ...*

Table 6.3: A comparison of baseline and cohesion-filtered English-French translations.

filtered output matches no more reference N-grams than the baseline, due to a heavily paraphrased reference, which prevents any change in BLEU score.

The second example typifies our other observed form of improvement, where we actually select a favorable, but less likely segmentation. In this case, the baseline system moves a phrase out of the way to make use of high-probability phrase-pairs and to improve language model probability. The results are tragic, erroneously removing negation from *exist* and then adding it to *reduce*, completely reversing the meaning of the input sentence. The cohesion-filtered output does not have access to this particular movement choice: the verb phrase rooted at *exist* must remain contiguous. This forces the decoder to select a less likely collection of phrases, "[that do] [not currently exist]", and allows the language model, no longer working against us, to assemble the correct French construction "qui n'existent pas encore". Again, we receive no credit for this change, due to paraphrase in the reference.

At this point we have built a motivating case for this constraint in phrasal decoding. It appears to be doing some good, sometimes saving a translation dramatically. Furthermore, if we were luckier with our references, these changes could have very easily improved our BLEU score. Our current solution is held back by two major disadvantages:

1. We are only affecting a third of the sentences we could possibly improve, due to working with K-best lists, which may not contain cohesive output.
2. The minimum error rate training used to tune our log-linear feature model does not know that the K-best list output will be cohesion-filtered. Therefore, it is not optimizing the final output of the decoder.

System	Dev	Test	Time (s)
Baseline	32.04	32.35	159
Base + hard	32.18	32.36	131
Base + soft	**32.22**	**32.88**	409

Table 6.4: BLEU scores for English to French with integrated cohesion constraint.

Both of these issues are addressed by the cohesive decoding algorithm described in § 6.2, which pushes the cohesion constraint deep into the decoder, so it is aware of cohesion violations throughout hypothesis construction and model tuning.

6.3.3 Constrained Decoding

In this section, we describe experiments with the Moses decoder using the hard and soft cohesion constraints described in § 6.2. We begin by attempting to determine if these deep constraints outperform an unconstrained decoder. These experiments are followed by comparisons using an alternative re-ordering model, and then testing the constraint for English to German translation. Our last two experiments provide a human evaluation of the soft constraint, and a piece-wise analysis of our development and test sets. Unless otherwise stated, we use the same English to French translation infrastructure described in Section 4.4.1 as our baseline. To demonstrate consistency, we report results on our 1500-sentence development set, as well as our 2000-sentence blind test set.

Comparison to Baseline

The first experiment varies the constraints employed in the standard Moses decoder, and examines the effect on BLEU score. The results are shown in Table 6.4, with timing data for the test set in the right-most column. Recall that the 1000-best filtering approach achieved a score of 32.08 on the development set. As one can see, the cohesive decoder has improved our BLEU scores on the development set, but only modestly. The test set, on the other hand, shows an improvement of more than half a BLEU point using the soft constraint. Conversely, the hard constraint does not improve BLEU score at all. From these results, we can draw three tentative conclusions:

1. The cohesion constraint does not appear to harm BLEU score.
2. The contribution of the cohesion constraint is variable, depending on the target references and the quality of the input parse trees.
3. The soft constraint seems more reliable than the hard one.

System	Dev	Test	Time (s)
Baseline	32.04	32.35	159
Lex	32.19	32.71	1687
Lex + hard	**32.47**	32.85	1006
Lex + soft	32.45	**32.90**	2126

Table 6.5: BLEU scores for English to French with lexical re-ordering.

The following experiments are designed to help us better understand the constraint and its interactions with other decoding options and language pairs.

Comparison to Lexical Re-ordering

Another way to improve the distortion behavior of a phrase-based decoder is to incorporate a lexical re-ordering model (§ 2.6.1). This re-ordering component memorizes preferences observed in the training bitext for the placement of bilexical blocks. The lexical re-ordering model is considered to be important enough to be included in competition-grade translation systems (Koehn et al., 2005). Since both systems intend to improve upon the simple distortion modeling implemented in the baseline system, we compare and combine the techniques in this section. The results on our English-French task are shown in Table 6.5.

The first thing to note is that the lexicalized re-ordering model alone is very expensive; it takes an order of magnitude longer to decode than the base system. Also, its BLEU scores of 32.19 and 32.71 are in line with the soft cohesion constraint tested in the previous experiment, which achieved BLEU scores of 32.22 and 32.88.

Things begin to get much more interesting once we examine the combined systems. Looking at our development data, these two types of ordering information appear to be quite complimentary. The combined systems using both the hard and soft constraint exhibit improvements that are greater than the sum of improvements observed with the individual systems. In the test set, we see a similar effect with lex + hard. Only when examining the soft constraint on the test data, do we see no benefit from the lexical re-ordering model. This could be because the soft constraint was already exhibiting a substantial increase in BLEU score on its own. In all cases, the addition of cohesion data improves upon the score of the lexical re-ordering model alone.

English-German experiments

English-French is a very close language pair, that exhibits very little re-ordering. The re-ordering that does occur is very local. We repeated all of our experiments for English to

System	Dev	Test	Time (s)
Baseline	17.98	18.02	343
Base + hard	18.08	18.26	122
Base + soft	18.18	18.19	194
Lex	18.17	18.25	1943
Lex + hard	18.06	18.30	792
Lex + soft	**18.27**	**18.39**	1932

Table 6.6: BLEU scores for English to German.

German translation, in the hope that our method would show a greater improvement in this case. Past results (Collins et al., 2005) have shown that syntactic pre-processing for phrase-based translation can improve German-to-English translation, but no similar result exists for English to German.

We use a very similar translation infrastructure for these experiments, again relying on our baseline Moses translation system. We adjust the distortion limit from 4 to 6 to account for the long-distance re-ordering that can occur in this language pair. The English-German training bitext, also originating from Europarl and provided by the SMT workshop, has 751K sentence pairs, and comes with development and test sets. Again, we use the first 500 development pairs for MERT tuning, the remaining 1500 as a development set, and we leave the 2000 pair test set for blind testing. Our results are shown in Table 6.6.

Unfortunately, we do not observe a wider gap between standard and cohesive decoding in our German-English tests. Instead, we see a similar level of improvement on what appears to be a much more difficult language pair. All BLEU scores are 10 points lower than their French counterparts, but they are in line with scores observed on the same data set with other phrase-based systems (Koehn and Monz, 2006). From Table 6.6, one can see that most of the trends observed in the French experiments still hold for German. Most importantly, the system combining both lexical re-ordering and the soft cohesion constraint continues to perform best, and the soft cohesion constraint alone continues to perform with consistency. The hard constraint displays a little more variation with this data set.

Human Evaluation

In § 6.3.2 we observe that maintaining phrasal cohesion appears to affect sentences in a way that is not detected by BLEU score. To test this hypothesis analytically, we conduct a human evaluation to determine whether bilingual speakers prefer baseline or cohesion-constrained translations.

Annotator	base	+soft	equal
Annotator #1	14	47	14
Annotator #2	21	46	8

Table 6.7: Counts of preferences from the human evaluation.

Our comparison systems are the baseline decoder (base) and our soft cohesion constraint (base+soft). We test on our development set for the English to French task, as it has one of the smaller observed BLEU-score gaps: 32.04 for base, and 32.22 for base+soft. Our experimental set-up is modeled after the human evaluation presented in (Collins et al., 2005). We provide two human annotators[2] a set of 75 English source sentences, along with a reference translation and a pair of translation candidates, one from each system. The annotators are asked to indicate which of the two system translations they prefer, or if they consider them to be equal. To avoid bias, the competing systems were presented anonymously and in random order. Following (Collins et al., 2005), we provide the annotators with only short sentences: those with source sentences between 10 and 25 tokens long. Following (Callison-Burch et al., 2006), we conduct a targeted evaluation; our method directly attempts to improve sentences where we detect a cohesion violation in the baseline translation, so we only draw our evaluation pairs from this set. There are only 75 sentences from the development set that meet these two criteria; all of them were included in the evaluation.

The aggregate results of our human evaluation are shown in Table 6.7. Each annotator prefers base+soft in over 60% of the test sentences, and each prefers base in less than 30% of the test sentences. This presents very strong evidence that we are having a consistent, positive effect on formerly non-cohesive translations. A complete confusion matrix indicating agreement between the two annotators is given in Table 6.8. There are a few more off-diagonal points than one might expect, but it is clear that the two annotators are mostly in agreement with respect to base+soft's improvements. A combination annotator, which selects base or base+soft only when both human annotators agree and equal otherwise, finds only 6 sentences where base is preferred, compared to 35 for base+soft.

Piece-wise Evaluation

In our human evaluation, to make best use of our annotators' time, we sampled only from the space of sentences we expected to improve: sentences whose baseline translations con-

[2]Annotators were both native English speakers who speak French as a second language. Neither were familiar with the nature of how our system attempted to improve translation before the experiment began.

	Annotator #2		
Annotator #1	base	+soft	equal
base	6	7	1
+soft	8	35	4
equal	7	4	3

Table 6.8: Confusion matrix from human evaluation.

Cohesive Dev (1170)

	¬ soft	soft
¬ lex	33.80	33.82 (+0.02)
lex	33.91 (+0.11)	34.12 (+0.32)

Non-cohesive Dev (330)

	¬ soft	soft
¬ lex	27.46	28.04 (+0.58)
lex	27.86 (+0.40)	28.09 (+0.63)

Cohesive Test (1563)

	¬ soft	soft
¬ lex	33.78	34.03 (+0.25)
lex	33.89 (+0.11)	34.04 (+0.26)

Non-cohesive Test (437)

	¬ soft	soft
¬ lex	28.73	29.86 (+1.13)
lex	29.66 (+0.93)	29.83 (+1.10)

Table 6.9: Piece-wise BLEU evaluations for development and test sets.

tained detectable cohesion violations. However, cohesive decoding can affect sentences that were originally being translated cohesively, because the presence of the constraint can affect the log-linear weights assigned by MERT. In this section, we examine the effect of cohesive decoding on these two types of sentences. We split our development and test sets according to the sentences for which the baseline system generates a cohesive translation, generating cohesive and non-cohesive subsets for each. We then evaluate our baseline and our most promising systems: base+soft, base+lex and base+lex+soft, on each resulting set.

The results of this experiment are shown in Table 6.9. The rows and columns indicate if either the soft cohesion constraint or the lexical re-ordering model were added to the base decoder; therefore $\neg soft, \neg lex$ corresponds to our baseline. Differences in BLEU score with respect to the baseline are indicated in parentheses.

We can immediately make some general observations. First, the cohesive:non-cohesive ratio is the same for both development and test sets, roughly 4:1 in each case. Second, it appears that the cohesive set consists of easier translations for phrasal decoding. Baseline scores are 33.8 for both cohesive subsets, compared to 32.0 and 32.35 for the complete development and test sets. This is interesting, as it makes cohesion detection a sort of confidence estimate for phrasal SMT output: the lack of a violation should increase our confidence in translation quality.

Most importantly, we can note that both the cohesion feature and lexical re-ordering

System	Di	Co	LM	TM1	TM2	TM3	TM4	Ph	Wo
Base	0.898	–	1.570	0.472	1.008	1.131	0.472	0.162	-3.000
Base + soft	0.193	2.759	0.518	0.115	0.502	0.561	0.116	0.113	-0.942

Table 6.10: Weights resulting from minimum error rate training.

models have a much larger impact on the non-cohesive subset. In the cohesive subset, adding syntactic or lexical ordering information produces only modest gains in BLEU, up to an absolute improvement of about 0.3. Conversely, the non-cohesive subset shows improvements ranging from 0.4 to 1.1. Also note that the cohesion feature consistently results in more improvement on non-cohesive translations than lexical re-ordering. Finally, we can observe that, contrary to our preliminary observations, BLEU score can detect the effects of our cohesive decoder, but those differences in score appear to be washed out by the smaller improvement on the majority cohesive class. In fact, it appears that the test-set BLEU scores observed in our original comparison were so much more impressive than the development-set scores because, on the test set, the soft cohesion feature produces not only a large improvement on the non-cohesive subset, but also a modest improvement on the much larger, cohesive subset.

6.3.4 Error Analysis

In this section we perform some error analysis, examining the weights output by the minimum error rate training, and examining how the decoder uses the freedom provided by the soft cohesion constraint.

Log-linear Weight Analysis

The weights produced by minimum error rate training for our English to French task, with and without a soft cohesion constraint, are shown in Table 6.10. As one can see, the addition of the interruption counting feature *Co* lowers the weights on all other features, so that the presence of an interruption can have a substantial impact. Note that, of all the original features, the distortion penalty *Di* receives the largest relative reduction, being reduced to nearly a fifth of its previous weight. This indicates that the cohesion feature is taking pressure off the distortion model, effectively doing part of its job, by ruling out non-cohesive movement.

The above phenomenon, coupled with our BLEU results using the lexical re-ordering model, lead us to believe that the cohesion constraint can enhance the effect of other, non-

	Violation	
	True	False
Parse Correct	8	2
Parse Incorrect	1	10

Table 6.11: Analysis of cases where the soft cohesion constraint detects a violation, but is over-ridden by the decoder.

syntactic movement models. By handling certain syntactic movement errors, the constraint allows the tuning system to adjust the other movement models to account for only the remaining types of movement errors. This could potentially result in the decoder allowing more movement over-all, so long as it is accounted for the in the parse tree.

Soft Constraint Analysis

Recall that, in our initial filtering experiments on our 1500-sentence English-French development set, 330 of our baseline translations displayed cohesion violations. As we can see in Table 6.10, when we provide the interruption count as a feature during tuning, it is weighted heavily to help optimize BLEU score. The resulting decoder (base+soft) exhibits only 21 cohesion violations on our development set. In this section, we examine those 21 violations to characterize the sentences in which the soft cohesion constraint is over-ridden. It is our hope that this will help us better understand the soft constraint's relative stability with respect to the hard constraint.

We wish to determine if the flexibility provided by the soft constraint is used systematically, and if so, is it done to overcome inherent problems with syntactic cohesion, or is its primary use is to bypass parse errors? To this end, we check the tree fragments that cause the violations to see if the tree structure itself is correct. We also classify each offending movement decision according to whether or not the observed movement should actually be prohibited. The results are shown in Table 6.11.

The numbers show that our broad cases are distributed fairly evenly: 11 violations result from correct parses and 10 are from incorrect parses.[3] Meanwhile, 9 violations display truly illegal movement, and 12 are false violations; that is, a cohesion violation is detected but the offending movement is deemed acceptable by the author. In these cases, the decoder was right to over-ride the soft constraint. As we look at the cross-product of these two categories, we see that there is potential for further improvement within our framework by

[3]Our definition of a correct parse concerns only the tree fragment that is directly involved in generating the cohesion violation. All of the parse trees we examined contained at least one error.

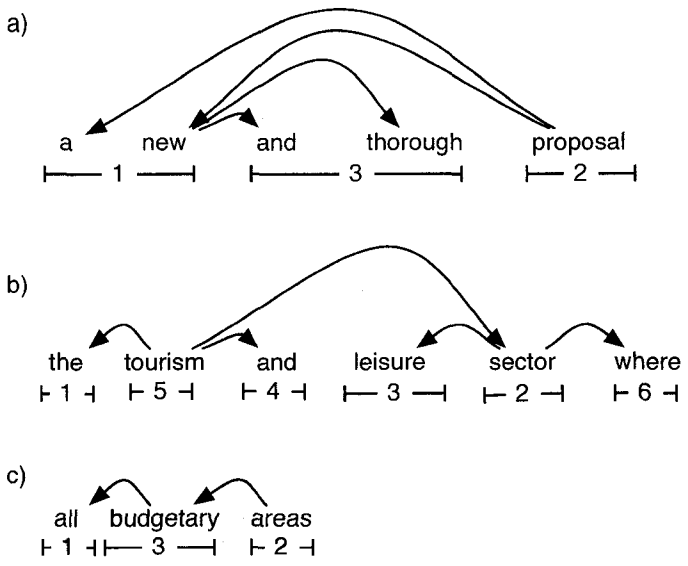

Figure 6.4: Three false violations, and the cohesion violating order in which their nodes are translated. (a) has a correct parse tree, while the trees for (b) and (c) are incorrect.

improving the parser. Most of our cases are along the diagonal, that is, either both the parse and violation are correct, or the parse has errors that lead to a false violation. If we could eliminate these parse errors, then the false violations would disappear. This would allow further weight to be placed on the interruption penalty, preventing the decoder from bypassing the cohesion constraint for true violations.

Fortunately, the false violations are easily characterized. Both cases where the parse was deemed correct involve conjunction. Minipar creates conjunction dependency chains; later conjuncts are descendants of earlier conjuncts. This can lead to unnecessary restrictions on the relative re-ordering of conjuncts. An example is shown in Figure 6.4a. Switching to a representation where conjuncts are siblings would alleviate this problem. Furthermore, 7 of the 10 false violations with incorrect parses also involve conjunction. In these cases, the parser stumbles over the longer conjunctions found in Europarl's dialogue-like text, often leading to a reversed dependency relationship. An example of this phenomenon is shown in Figure 6.4b, where the parser makes an attachment error; *tourism* should modify *sector*. Most of the remaining false violations are caused by misparsed noun phrases that create incorrect chains of modifiers, as illustrated in Figure 6.4c.

6.3.5 Related Results

This section discusses our results in the context of other attempts to integrate syntax into left-to-right phrasal decoding. In general, one can hope for three possible types of improvements from integrating source-side syntax into a phrasal decoder:

1. Improved lexical choice through syntactically informed phrases.
2. Improved movement decisions through syntactic generalization.
3. Improved movement decisions through respect for syntactic boundaries.

In enforcing syntactic cohesion, we are limiting ourselves to only the final item on that list: the movement observed during translation must respect the boundaries of syntactic phrases in the source sentence. One can think of our approach as a method to isolate that single factor, while leaving the rest of the standard phrase-based translation infrastructure untouched. Pre-processing methods, on the other hand, can potentially tap into all three sources of improvement, but they do so at the cost of having to select a single re-ordering before translation begins.

Xia and McCord (2004) apply automatically-learned syntactic transformations to both their training and test data. This allows them to capture new lexical relationships by moving related words closer together, so they can be captured by a contiguous phrase. It also allows their concept movement to benefit from broad linguistic generalizations such as "*Adj Noun* $\rightarrow$ *Noun Adj*". However, to achieve their performance gains, they have to completely disable the decoder's distortion ability, committing themselves to only the movement observed in their one-best re-ordering. Furthermore, they only achieve significant improvements when testing on out-of-domain source text.

Collins et al. (2005) also take a one-best syntactic re-ordering approach that re-orders both training and test data as a pre-processing step. They specifically target German-to-English translation. Their small set of hand-designed rules are linguistically motivated to improve the locality of German clauses with respect to English. Thus, the approach inherently generalizes linguistic movement. Furthermore, some modifiers are moved closer to their head verbs, allowing phrases to capture their agreement. However, since they allow their phrase-based decoder to perform distortions as well, they lose any guarantees that the output will respect input syntactic boundaries as it processes re-ordered German text.

Since we do not alter our training data, we are unable to take advantage of improved lexical locality. Our constraint forces us to respect syntactic boundaries, and since it is incorporated into the decoder, rather than a pre-processing step, it allows the decoder to try

a much larger range of movements. Our current constraint is only aware of the structure of the tree, and does not account for part of speech, dependency labels, or even node height. Therefore, it does not benefit from syntactic movement generalizations. As such, top level clauses are as likely to be re-ordered as noun-adjective pairs. This is why it is currently necessary to leave the distortion limit intact, even though our constraint drastically limits the possible re-ordering space. Top-level clause re-orderings, though syntactically legal, rarely occur in English-to-French translation.

It could be possible to incorporate some level of syntactic generalization into a phrase-based decoder as future work. Our interruption model of cohesion is based on the notion of the decoder leaving one subtree to enter another. We could create a model that uses this same notion to actually grade various legal movement decisions. We are currently calculating the lowest common ancestor between two adjacent phrases in order to check for interruptions; one simple approach would be to use the height of this common ancestor as a feature to grade the observed movement.

6.4 Summary

We have provided a meaningful definition of syntactic cohesion that is applicable to the output of phrasal SMT. We have used this definition to develop a linear-time algorithm to detect cohesion violations in partial phrase-based-decoding hypotheses. This algorithm was used to implement both hard and soft cohesion constraints for the Moses decoder, based on a source-side dependency tree.

Our experiments have shown that roughly 1/5 of our baseline translations contain detectable cohesion violations, and these translations tend to receive lower BLEU scores. Addressing these violations with a soft cohesion constraint has been shown to improve the BLEU scores of formerly non-cohesive translations considerably, with observed absolute improvements ranging between 0.6 and 1.1 points. However, when we examine the net effect on all translation outputs, BLEU score improves only modestly, as the pool of sentences that receive a large benefit is fairly small. However, the soft cohesion constraint never adversely affects over-all BLEU score, and can still create absolute improvements of up to 0.5 BLEU points. Syntactic cohesion also appears to produce BLEU score improvements that are orthogonal and complimentary to those of lexical re-ordering models. A human evaluation showed that translations created using a soft cohesion constraint are preferred over non-cohesive translations in the majority of cases.

Chapter 7

Conclusion

We have presented three distinct contributions to machine translation, covering the sub-problems of translation modeling, word alignment, and decoding. These projects, though separate, are connected through their use of syntactic constraints on concept movement. We begin this chapter by briefly summarizing each of these main contributions, before discussing our results in aggregate and suggesting some future work.

Contributions

With our phrasal ITG, we have shown how a small modification to the canonical binary bracketing ITG can create a model that is very similar to Marcu and Wong's (2002) joint phrasal translation model. The ITG allows us to perform perfect search and expectation, at the cost of introducing the ITG constraints on concept movement. Enabled by a novel fixed-link constraint on the inside-outside algorithm, we have tested this model both as a method for phrase-table generation, and as a phrasal word-aligner. Our phrasal aligner, once augmented with our novel non-compositional constraint, has been shown to produce high recall alignments, achieving an F-measure that is comparable to that of GIZA++. Furthermore, we have shown that the EM-learned phrase table, when used for phrasal decoding, produces a 1-point improvement in BLEU score over the C-JPTM, demonstrating the value of perfect expectations. The same model scores within 0.4 BLEU points of a heuristic phrase-extraction baseline, but with a 4-times smaller phrase table.

In our study of syntactic constraints for word alignment, we investigated the effects of various levels of agreement with an input dependency tree on one-to-one word alignment. This project provides two primary algorithmic contributions: a number of methods to combine cohesion and ITG constraints, and a method to discriminatively train an ITG bitext parser. The former resulted in the creation of two new alignment spaces: C-ITG, which

intersects ITG and cohesion space, and H-ITG, which adds a head-relationship constraint. Our experiments showed that the ITG constraints alone could produce a 10% reduction in error rate for a weak aligner, while the C-ITG constraints produced a 31% reduction. Expressiveness tests showed that there are few disadvantages to ITG constraints in a one-to-one aligner, while the C-ITG did produce a noticeable reduction in achievable recall. We also demonstrated that the H-ITG produced systematic errors, making it unattractive for future development. In our experiments with discriminative training, we used the flexible feature representation made available by SVM training to reproduce a strong bipartite matching baseline, which we then compared to an SVM-ITG with soft cohesion constraints. These experiments showed that, while a strong discriminative aligner does not receive much benefit from a hard cohesion constraint, our soft constraint was able to produce a 22% relative reduction in error rate.

The final project investigated syntactic cohesion in the context of left-to-right phrasal decoding. We developed a definition of cohesion that is relevant to phrasal decoder output, and presented two ways to use it in translation. The first, a K-best cohesion filter, was shown to produce no improvement in BLEU score. However, a deep cohesion constraint, implemented as a linear-time check on each hypothesis extension, was shown to perform much better. In particular, a soft cohesion penalty trained using minimum error rate training was shown to never harm over-all BLEU score, and to produce at times up to a 0.5-point absolute improvement. The deep constraint was also shown to combine well with lexical reordering models. Examining only the subset of sentences where we could detect a cohesion violation in the output of the baseline decoder, we found that our cohesion feature was producing noticeable improvements in BLEU score for this subset, between 0.6 and 1.1 absolute points. In the complete test set, this improvement was masked by the large number of naturally cohesive translations, which received little benefit. Finally, a human evaluation showed that bilingual speakers prefer our soft-constrained output to non-cohesive baseline output in a clear majority of cases.

Discussion

In this thesis, we have described several ways to benefit from the constraints that come with context-free assumptions regarding concept movement. In this section, we take a moment to discuss some general lessons learned regarding our ITG and cohesion constraints.

Our translation modeling and word alignment solutions in Chapters 4 and 5 are subject to ITG constraints. When introduced in (Wu, 1997), it was suggested that violations to

these constraints do not occur in practice. Our work, along with other studies such as (Wellington et al., 2006b) have shown that violations do indeed happen in real data. In § 5.3, we found 3 violations in a 500-sentence hand-aligned bitext. In § 4.4.1, we found 3,376 violations in a 393K-sentence automatically-aligned bitext. Considering both cases, it appears that less than 1% of human, English-French translations produce ITG violations; but in a large enough corpus, that can still mean missing out on thousands of sentence pairs. Ultimately, we feel that the advantages of the ITG formalism can, in the right situation, outweigh the accompanying reduction in expressiveness. With our phrasal ITG, we collect complete expectations to greatly improve our model output, surpassing a comparable flat-string model that was free to use non-ITG sentence pairs. With our SVM-ITG, the ITG allows us to do perfect search with a well-defined cohesion violation event, creating a good fit for SVM Struct, which expects a perfect search. Our results in § 5.3 suggest that the limits imposed by an ITG are much weaker than those of a fixed-tree cohesion constraint, and that adding them to an existing cohesion constraint through a C-ITG has no negative effect on cohesion's already limited expressiveness.

In Chapters 5 and 6, we intentionally introduce cohesion constraints to our algorithms to eliminate undesirable outputs and improve accuracy. These constraints limit movement according to the syntactic phrases in a fixed English dependency tree, introducing distortion data that is orthogonal to current Markov-1 or position-based distortion models. In both decoding and alignment, the cohesion constraint does provide an accuracy boost, but it appears to be best employed as a soft constraint. We provide direct evidence of the limits of a hard cohesion constraint with our expressiveness tests in § 5.3, where a hard C-ITG constraint misses 67 more correct links than an ITG alone. With our SVM aligner in § 5.5, the soft constraint produces a larger reduction in error rate than a hard constraint. Furthermore, the soft constraint produces an increase in alignment recall, in place of a reduction. In our cohesive decoding experiments in § 6.3, the difference between hard and soft constraints is more subtle, but it does appear that the soft constraint performs more consistently throughout our tests. These cohesion violation features provide guidance to our decoders and aligners, despite the relatively small amount of movement observed in the English-French language pair, and despite errors in our automatically-derived dependency trees. We set out to explore the benefits of syntactic cohesion in isolation. There appears to be little lost, and much to gain from the inclusion of a dependency-based cohesion penalty, especially when this penalty can be trained discriminatively.

Future Work

We briefly describe three ideas for future work that have arisen as a result of our research: a Bayesian extension for our phrasal ITG, the exploration of C-ITG constraints for syntactic decoding, and the notion of a translation-driven monolingual parser. We describe each in turn below.

Joint phrasal translation models are biased toward longer phrases. Our phrasal ITG and its JPTM competitor must both apply a lexicon constraint, limiting the pool of potential phrases to those that have occurred at least 5 times monolingually. If they do not, then EM will always align entire sentences to entire sentences, as one giant phrase-pair. This pressure to always build the largest phrase-pair possible may be overwhelming other strong correlations in our training data. In § 4.4.4, we argue that one major advantage of the surface heuristic for phrase extraction is its ability to avoid these frequency-based lexicon constraints. We would like to investigate the use of Bayesian priors to provide a sort of soft lexicon constraint. In (Johnson et al., 2007), a monolingual grammar induction system, which faced a similar over-fitting problem due to fluid tokenization, was repaired through the use of Gibbs sampling. The key to their solution was a simple Dirichlet prior that prefers sparse lexicons. Extending their approach to the bitext parsing case with our phrasal ITG should be straight-forward, and could eliminate the need for lexicon constraints. If this does not work, then more complex and expressive priors, such as the Dirichlet process used in (Goldwater et al., 2006) for unsupervised text segmentation, could be explored.

We have thoroughly tested our C-ITG, which combines cohesion and ITG constraints, in the word alignment task. We have not investigated the same combination for decoding. It would be fairly straight-forward to implement an additional ITG constraint in our cohesive version of Moses, using the left-to-right ITG constraint described in (Zens et al., 2004); however, we suspect that an additional ITG constraint would do little to improve accuracy, as cohesion constraints are already very strict on their own. What would be interesting is the use of the combination inside a syntactic decoder. This could take one of two forms, depending on whether we start from an ITG or a tree-to-string method. An already polynomial-time ITG decoder, like those described in (Huang et al., 2005) or (Zhang and Gildea, 2006a), could be augmented with a cohesion constraint based on a source-side dependency tree to improve efficiency and accuracy. Similarly, a tree-to-string model, such as the dependency treelet decoder described in (Quirk et al., 2005), could be augmented with ITG constraints. These constraints would have a limited effect on the expressiveness of the

decoder, as an ITG adds very few limits on top of cohesion. However, the ITG's binary-branching assumptions can potentially improve dynamic-programming efficiency, and rein in the complexity of permuting flat tree structures. Other approaches that have limited a string-to-tree decoder through binarization have benefited greatly (Zhang et al., 2006).

From the success of our SVM-ITG aligner and our cohesive phrasal decoder, it is safe the say that it is possible to analyze the syntactic structure of a sentence in a monolingual context, and use that analysis to guide movement decisions in translation. However, it is not clear that the linguistically-motivated trees that monolingual parsers currently produce are optimal for translation tasks; in fact, our experiments in § 5.3, which show cohesion limiting achievable recall, indicate that our current dependency trees are not optimal for alignment. This may be because our parser is building sub-optimal tree structures, or it may be because the parser was optimized for news and not parliamentary data. In this thesis, to overcome cases where our trees do not match our desired translations and alignments, we use cohesion as a soft constraint. An alternative solution would be to change the dependency tree to make it a better fit for our domain and task. Our parser does not need to be trained on treebank trees. A discriminative parser, such as one trained by SVM Struct, can train according to any error criteria that can be calculated efficiently. Therefore, we could use an error signal that forces the produced tree to allow the smallest amount of concept movement possible, while still allowing the movement exhibited by a provided bilingual word-alignment. Thus, our gold standard for monolingual parsing would become bilingual word alignments, rather than treebank trees. The SVM parser would still use standard parsing features, such as non-terminal productions, lexical items and part-of-speech tags, along with general features like dependency length. The system would be evaluated according to how much the translation-oriented parser improves cohesion-constrained word alignment or translation. It would be interesting to see how closely these translation-motivated trees match trees from a treebank parser, and particularly illuminating to observe how they differ.

Bibliography

H. Alshawi, S. Bangalore, and S. Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1):45–60.

N. F. Ayan, B. J. Dorr, and C. Monz. 2005. Neuralign: Combining word alignments using neural networks. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 65–72, Vancouver, Canada, October.

A. L. Berger, P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. R. Gillett, J. D. Lafferty, R. L. Mercer, H. Printz, and L. Ureš. 1994. The Candide system for machine translation. In *Proceedings of the 1994 ARPA Workshop on Human Language Technology*.

A. Birch, C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Constraining the phrase-based, joint probability statistical translation model. In *HLT-NAACL Workshop on Statistical Machine Translation*, pages 154–157.

P. Blunsom and T. Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 65–72, Sydney, Australia, July.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

C. Callison-Burch, C. Bannard, and J. Scroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, Michigan, June.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 249–256.

C. Cherry and D. Lin. 2003. A probability model to improve word alignment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 88–95, Sapporo, Japan, July.

C. Cherry and D. Lin. 2006a. A comparison of syntactically motivated word alignment spaces. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 145–152.

C. Cherry and D. Lin. 2006b. Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 105–112, Sydney, Australia, July.

C. Cherry and D. 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.

C. Cherry. 2003. A probability model to improve word alignment. Master's thesis, University of Alberta.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270, Ann Arbor, USA, June.

M. Collins, P. Koehn, and I. Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 531–540.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Y. Deng and W. Byrne. 2005. HMM word and phrase alignment for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 169–176, Vancouver, Canada.

Y. Ding, D. Gildea, and M. Palmer. 2003. An algorithm for word-level alignment of parallel dependency trees. In *Proceedings of the MT Summit*, New Orleans.

T. E. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

J. Eisner. 2003. Learning non-ismorphic tree mappings for machine translation. In *Meeting the Association for Computational Linguistics (Companion Volume)*, Sappora, Japan.

H. J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 304–311.

A. Fraser and D. Marcu. 2006a. Measuring word alignment quality for statistical machine translation. Technical Report ISI-TR-616, ISI/University of Southern California, May.

A. Fraser and D. Marcu. 2006b. Semi-supervised training for statistical word alignment. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 769–776, Sydney, Australia.

H. Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8(304–337).

W. A. Gale and K. W. Church. 1991a. Identifying word correspondences in parallel texts. In *4th Speech and Natural Language Workshop*, pages 152–157. DARPA.

W. A. Gale and K. W. Church. 1991b. A program for aligning sentences in bilingual corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–184.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 273–280, Boston, USA, May.

D. Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 80–87, Sapporo, Japan.

D. Gildea. 2004. Dependencies vs. constituents for tree-based alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

S. Goldwater, T. L. Griffiths, and M. Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 673–680, Sydney, Australia, July.

C. Goutte, K. Yamada, and E. Gaussier. 2004. Aligning words using matrix factorization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain, July.

J. Graehl and K. Knight. 2004. Training tree transducers. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 105–112, Boston, USA, May.

T. Hofmann. 1999. Probabilistic latent semantic analysis. In *Uncertainty in artificial intelligence*, pages 289–296.

J. Hopcroft and J. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley.

L. Huang, H. Zhang, and D. Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 65–73, Vancouver, Canada, October.

L. Huang, K. Knight, and A. Joshi. 2006. A syntax-directed translator with extended domain of locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 1–8.

A. Ittycheriah and S. Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 89–96, Vancouver, Canada, October.

M. Johnson, T. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 139–146, Rochester, New York, April.

M. Johnson. 1985. Parsing with discontinuous constituents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

R. Kneser and H Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 181–184.

K. Knight and Y. Al-Onaizan. 1998. Translation with finite-state devices. In *Proceedings of the 4th AMTA Conference*.

K. Knight. 1999. Squibs and discussions: Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, December.

P. Koehn and C. Monz. 2006. Manual and automatic evaluation of machine translation. In *HLT-NACCL Workshop on Statistical Machine Translation*, pages 102–121.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 127–133.

P. Koehn, A. Axelrod, A. Birch Mayne, C. Callison-Burch, M. Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *International Workshop on Spoken Language Translation*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Demonstration session.

S. Lacoste-Julien, B. Taskar, D. Klein, and M. I. Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 112–119, New York City, USA, June.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

D. Lin and C. Cherry. 2003. Word alignment with cohesion constraint. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, volume Short Papers, pages 49–51, Edmonton, Canada, May.

D. Lin. 1994. Principar - an efficient, broad-coverage, principle-based parser. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 42–48, Kyoto, Japan.

D. Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of Coling 2004*, pages 625–630, Geneva, Switzerland, August.

Y. Liu, Q. Liu, and S. Lin. 2005. Log-linear models for word alignment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 459–466, Ann Arbor, USA.

A. Lopez and P. Resnik. 2005. Improved HMM alignment models for languages with scarce resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 83–86, Ann Arbor, Michigan, June.

A. Lopez, M. Nossal, R. Hwa, and P. Resnik. 2002. Word-level alignment for multilingual resource acquisition. In *Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data.*

C. D. Manning and H. Schütze. 2001. *Foundations of Statistical Natural Language Processing*. The MIT Press.

D. Marcu and W. Wong. 2002. A phrase-based, joint probability model for statistic machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139.

I. D. Melamed and W. Wang. 2006. Proteus project working paper #2: Statistical machine translation by generalized parsing. Technical report, New York University.

I. D. Melamed, G. Satta, and B. Wellington. 2004. Generalized multitext grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain, July.

I. D. Melamed. 1997. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–313, Providence, USA.

I. D. Melamed. 1998. Manual annotation of translational equivalence: The blinker project. Technical Report 98-07, Institute for Research in Cognitive Science.

I. D. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

I. D. Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 158–165.

I. D. Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain, July.

R. Mihalcea and T. Pedersen. 2003. An evaluation exercise for word alignment. In *HLT-NAACL Workshop on Building and Using Parallel Texts*, pages 1–10, Edmonton, Canada.

R. Moore, W. Yih, and A. Bode. 2006. Improved discriminative bilingual word alignment. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 513–520, Sydney, Australia, July.

R. Moore. 2004a. Improving IBM word alignment model 1. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

R. Moore. 2004b. On log-likelihood-ratios and the significance of rare events. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 333–340, Barcelona, Spain.

R. Moore. 2005a. Association-based bilingual word alignment. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 1–8, Ann Arbor, Michigan, June.

R. Moore. 2005b. A discriminative framework for bilingual word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 81–88, Vancouver, Canada, October.

F. J. Och and H. Ney. 2000a. A comparison of alignment models for statistical machine translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1086–1090, Saarbrcken, Germany, July.

F. J. Och and H. Ney. 2000b. Improved statistical alignment models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 440–447, Hong Kong, China, October.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, July.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 161–168.

F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167.

C. H. Papadimitriou and K. Steiglitz. 1998. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–279, Ann Arbor, USA, June.

B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 73–80, Vancouver, Canada.

C. Tillman. 2004. A unigram orientation model for statistical machine translation. In *HLT-NAACL 2004: Short Papers*, pages 101–104.

I. Tsochantaridis, T. Hofman, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 823–830.

A. Venugopal and S. Vogel. 2005. Considerations in maximum mutual information and minimum classification error training for statistical machine translation. In *EAMT*.

J. M. Vilar and E. Vidal. 2005. A recursive statistical translation model. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 199–207.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 836–841, Copenhagen, Denmark.

S. Vogel, Y. Zhang, F. Huang, A. Tribble, A. Venugopal, B. Zhang, and A. Waibel. 2003. The CMU statistical machine translation system. In *MT Summmit*.

B. Wellington, J. Turian, C. Pike, and I. D. Melamed. 2006a. Scalable purely-discriminative training for word and tree transducers. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

B. Wellington, S. Waxmonsky, and I. D. Melamed. 2006b. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 977–984.

D. West. 2001. *Introduction to Graph Theory*. Prentice Hall, 2nd edition.

D. Wu and X. Xia. 1995. Large-scale automatic extraction of an English-Chinese translation lexicon. *Machine Translation*, 9(3–4):285–313.

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Santa Cruz, USA, June.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

F. Xia and M. McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of Coling 2004*, pages 508–514.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 523–530.

K. Yamada and K. Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

R. Zens and H. Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 257–264, Boston, USA, May.

R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 205–211, Geneva, Switzerland, August.

H. Zhang and D. Gildea. 2004. Syntax-based alignment: Supervised or unsupervised? In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 418–424.

H. Zhang and D. Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 475–482.

H. Zhang and D. Gildea. 2006a. Efficient search for inversion transduction grammar. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 224–231.

H. Zhang and D. Gildea. 2006b. Inducing word alignments with bilexical synchronous trees. In *Proceedings of the International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 953–960.

H. Zhang, L. Huang, D. Gildea, and K. Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 256–263.

Appendix A

Glossary

Natural language processing is a large, diverse field. Putting together a working NLP solution often requires tools from many sub-areas of the field, many of which have little or nothing to do with the current focus of the research. When we make use of such a technique, or draw a comparison to one, we attempt to describe it briefly here.

Beam search :

A beam search is a heuristic search of some state space. Like A* or Dijkstra's algorithm it uses a priority queue to visit the most promising states first, in attempt to find the best solution as quickly as possible. Unlike those other options, it is an incomplete search: there is no guarantee that it will find the optimal answer. This is because, in order to increase both time and space efficiency, it uses a limited queue size (unpromising states can fall off the priority queue and not be visited) and a limited beam width (when creating a list of successor states for the current state, only the best K successors are added to the queue). The main advantage of a beam search is that, unlike dynamic programming, there are few limitations on the features used to judge states, and unlike A*, any heuristics employed need not be admissible.

Language model :

A language model is a probability model used to characterize typical text for a particular language. A language model is designed to assign some probability to any sequence of words, but to assign more probability to the sequence if it is a correct or typical sequence for the language. For example:

$$\Pr(\text{“the dog jumps”}) > \Pr(\text{“dog the jumps”})$$

Modern language models take the form of an N-gram model (often $N = 3$), which

assesses the probability of each word according to the $N-1$ words that proceed it:

$$\Pr(w_1^m) = \prod_{i=1}^{m} \Pr(w_i|w_1^{i-1})$$
where:
$$\Pr(w_i|w_1^{i-1}) \approx \Pr(w_i|w_{i-1}w_{i-2})$$

Our translation systems use language models trained using the SRILM toolkit, which were provided by the 2006 SMT Workshop (Koehn and Monz, 2006). Both decoders we use employ Knesser-Ney Smoothing to handle unseen word sequences, ensuring that all sentences receive non-zero probability (Kneser and Ney, 1995).

Part of Speech Tagging :

Part-of-speech (POS) tagging is a sequence-tagging task. A sentence is treated as a sequence of tokens, and each token is labeled with its appropriate part-of-speech, for example:

det	*noun*	*verb*
the	dog	jumps

This task differs from normal classification in that the inputs and outputs are not independently distributed. In fact, the label of the previous word can have a large impact on the choice of the next word. For this reason, the task is usually accomplished with a Hidden Markov Model, or related technology, where each label is influenced by the token it is labeling and also by the label that came before it. The algorithms that allow these models to train and label efficiently (forward-backward and Viterbi, respectively), are beyond the scope of this document, but are described in (Manning and Schütze, 2001).

Markov-1 :

A Markov-0 model is one where the current state of the model provides sufficient information to make future decisions. Any higher order model (Markov-x, where $x > 0$) indicates that the model considers the current state, as well as the x previously observed states, when making its next decision. When applied to a sequence labeling task like POS-tagging or alignment, the Markov order of a model indicates how many labels backward in the sequence are used to inform the current label.

Appendix B

Proof: Interruption $\Longleftrightarrow$ Cohesion Violation

In this appendix we show that a cohesion violation implies an interruption, and an interruption implies a cohesion violation,thus the two concepts are equivalent.

We begin by restating the formal definitions of the two concepts. A cohesion violation is defined as follows:

> Given an e and e' that are related by either a sibling or head relationship, with the appropriate spans $[x, y]$ and $[u, v]$ (*spanS* and *spanS* for siblings, and *spanS* and *spanH* for head-modifier pairs), sort the spans so that $[u, v] \leq [x, y]$. A cohesion violation occurs if $x < v$.

An interruption occurs whenever the decoder would add a phrase $\bar{f}_{h+1}$ to the hypothesis $\bar{f}_1^h$, and:

$$
\begin{array}{ll}
\exists r \text{ s.t.} & \\
\exists e \in T(r) \text{ s.t. } e \in \bar{f}_1^h & \text{(a. Started)} \\
\exists e' \notin T(r) \text{ s.t. } e' \in \bar{f}_{h+1} & \text{(b. Interrupted)} \\
\exists e'' \in T(r) \text{ s.t. } e'' \notin \bar{f}_1^{h+1} & \text{(c. Not finished)}
\end{array}
$$

Next, we provide a lemma that will narrow down the definition of a cohesion violation:

Lemma 1 *If a cohesion violation is caused by the spans* $[u, v] \leq [x, y]$ *and* $x < v$, *then* $[u, v]$ *is not a head span.*

Assume $[u, v] \leq [x, y]$ and $x < v$ and $[u, v]$ is a head span. All head spans correspond to $spanH(e_i) = [a_i, a_i]$. Therefore $u = v$ and we have $[u, u] \leq [x, y]$ and $x < u$, and therefore $u \leq x < u$, which is a contradiction. This means for every cohesion violation, $[u, v] = spanS(e)$ for some e.

Now, we are ready to prove that a cohesion violation implies an interruption. Assume we have a cohesion violation; that is, there is some e such that $spanS(e) = [u, v]$ and either:

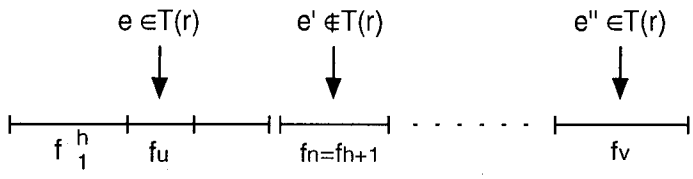

Figure B.1: A hypothesis extension $\bar{f}_1^h + \bar{f}_{h+1}$ that interrupts $T(r)$.

1. e's head e' has the head span $spanH(e') = [x, x]$ such that $[u, v] \leq [x, x]$ and $x < v$.

   or:

2. e's sibling e' has a subtree span $spanS(e') = [x, y]$ such that $[u, v] \leq [x, y]$ and $x < v$.

We prove that an interruption must occur by cases:

1. e' is e's head. The only way $[u, v] \leq [x, x]$ and $x < v$ is if $u < x < v$. By the definition of $spanH$, we know that $e' \in \bar{f}_x$. Finally, we know that $e' \notin T(e)$, because e' is e's head. Therefore, the addition of $\bar{f}_x$ to our growing hypothesis introduces an $e' \notin T(e)$ before $T(e)$, which began in $\bar{f}_u$, finishes in $\bar{f}_v$, and we have an interruption.
2. e' is e's sibling. Because e and e' are siblings, we know $T(e)$ and $T(e')$ are disjoint. Therefore, $\bar{f}_x$ contains an $\hat{e} \notin T(e)$ and $\bar{f}_v$ contains an $\bar{e} \notin T(e')$. We have three sub-cases allowed by $[u, v] \leq [x, y]$ and $x < v$:

   (a) $u < x < v$: $T(e)$ starts in $\bar{f}_u$ and is not finished when $\bar{f}_x$ is placed. $\hat{e} \in \bar{f}_x$ and $\hat{e} \notin T(e)$. $\hat{e}$ interrupts $T(e)$.

   (b) $(u = x) < v < y$: $T(e')$ starts in $\bar{f}_u$ and is not finished when $\bar{f}_v$ is placed. $\bar{e} \in \bar{f}_v$ and $\bar{e} \notin T(e')$. $\bar{e}$ interrupts $T(e')$.

   (c) $(u = x) < (v = y)$: This case is impossible. One cannot begin two siblings $T(e)$ and $T(e')$ on phrase $\bar{f}_u$, and finish them both on $\bar{f}_v, u \neq v$. $T(e)$ and $T(e')$ are disjoint, so at most one contiguous source phrase $\bar{f}_x$ can straddle both subtrees.□

Next, we prove that an interruption implies a cohesion violation. Assume we have an interruption as defined above. This definition can be visualized as shown in Figure B.1. We will refer to the interrupting phrase $\bar{f}_{h+1}$ as $\bar{f}_n$ for brevity (n stands for next). We call the phrase where $T(r)$ begins $\bar{f}_u$, and the phrase where it ends $\bar{f}_v$. We can then state that $spanS(r) = [u, v]$. Furthermore, since our tree is connected, we know e' is connected to r by some relationship, and since $e' \notin T(r)$, we know that e' is not a descendant of r.

Therefore e' is either an ancestor of r, or it is related to r by some lowest common ancestor c s.t. $c \neq r$,$c \neq e'$. Our proof that interruptions imply cohesion violations follows by cases:

1. e' is an ancestor of e:

   This leads to two sub-cases:

   (a) e' is the head of e: $spanH(e') = [n, n]$. $[u, v] \leq [n, n]$ and $n < v \rightarrow$ violation.

   (b) e' is the head of some ancestor r' of r: $spanH(e') = [n, n]$. Because $T(r')$ contains $T(r)$, we know $spanS(r') = [u', v']$ and $u' \leq u$ and $v \leq v'$. Therefore, $[u', v'] \leq [n, n]$ and $n < v' \rightarrow$ violation.

2. e' is related to r by some lowest common ancestor c s.t. $c \neq r, c \neq e'$:

   c has at least two modifiers: $m_{e'}$, the ancestor of e', and m_r the ancestor of r. These two modifiers are siblings, so if their subtree spans innersect, we have a cohesion violation. $T(m_r)$ contains $T(r)$, so $spanS(m_r) = [u', v']$ where $u' \leq u < n < v \leq v'$. $T(m_{e'})$ contains $T(e')$, so $spanS(m_{e'}) = [x, y]$ where $x \leq n \leq y$. The spans $[u', v']$ and $[x, y]$ share a point n in common, and n is not a boundary point of $[u', v']$. We can show that this always leads to an innersection by cases, depending on how the two spans are sorted:

   (a) $[u', v'] \leq [x, y] : x \leq n$ and $n < v'$. Therefore, $x < v'$, and innersection.

   (b) $[x, y] \leq [u', v'] : u' < n$ and $n \leq y$. Therefore $u' < y$ and innersection.

   Since we get an innersection in all cases, we get a violation in all cases.□

By showing that a cohesion violation implies an interruption, and that an interruption implies a cohesion violation, we have shown that the two concepts are equivalent. Thus, our search constrained to prevent interruptions will search exactly the cohesion-constrained output space.