Modeling Syntax for Parsing and Translation

Peter Venable

December 15, 2003

CMU-CS-03-216

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
John Lafferty, Chair
Daniel Sleator
Jaime Carbonell
Michael Collins (MIT)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation (NSF) under grant no. IIS-0312814, the Office of Naval Research (ONR) under grant no. N6600100C8007, SRI International under agreement no. 03-0002111, the ATR Company and IBM Corporation. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NSF, ONR, SRI, ATR, IBM, the U.S. government or any other entity.

Keywords: statistical,syntax,parsing,translation

Abstract

Syntactic structure is an important component of natural language utterances, for both form and content. Therefore, a variety of applications can benefit from the integration of syntax into their statistical models of language. In this thesis, two new syntax-based models are presented, along with their training algorithms: a monolingual generative model of sentence structure, and a model of the relationship between the structure of a sentence in one language and the structure of its translation into another language. After these models are trained and tested on the respective tasks of monolingual parsing and word-level bilingual corpus alignment, they are demonstrated in two additional applications. First, a new statistical parser is automatically induced for a language in which none was available, using a bilingual corpus. Second, a statistical translation system is augmented with syntax-based models. Thus the contributions of this thesis include: a statistical parsing system; a bilingual parsing system, which infers a structural relationship between two languages using a bilingual corpus; a method for automatically building a parser for a language where no parser is available; and a translation model that incorporates phrase structure.

Acknowledgements

The support and encouragement of many people was an important factor in the successful completion of this work. First of all, I want to thank my thesis advisor John Lafferty for his continual support, encouragement, and guidance. I'm also indebted to my committee members Danny Sleator, Mike Collins, and Jaime Carbonell for their support. Bing Zhao played an important part in the translation experiments, for providing translation hypotheses and for his invaluable aid when an understanding of the Chinese language was necessary. I'm thankful for the helpfulness of many others in putting together various pieces of this thesis, including Stephan Vogel, Alon Lavie, and David Temperley. Finally, a huge thank-you to Kelly, to Mom and Dad, to Lisa and Mindy, and to all my family and friends for your tireless encouragement and prayers. *Soli Deo Gloria!*

Contents

Chapter 1

Introduction

1.1 Motivation

Statistical methods are effective for many natural language processing tasks, including automatic translation and parsing. This thesis brings these two applications together in two ways, using translation to aid parser construction and using parsing to improve translation quality. This is made possible by a statistical model of syntactic structure and a statistical model of the relationship between the syntactic structures of two languages.

To parse a sentence is to resolve it into its component parts and describe its grammatical structure. This analysis is prerequisite to many tasks involving human language, both because a significant part of the meaning of a sentence is encoded in its grammatical structure (as opposed to the lexical selection), and because models that ignore structure are insufficient to distinguish well-formed from ungrammatical utterances. Applications that potentially benefit from syntactic parsing include corpus analysis, question answering, natural-language command execution, rule-based automatic translation, and summarization.

Many parsers are based on rules of grammar inferred through linguistic study. However, these rules are often too rigid to accommodate real-world utterances, which often bend the rules of "correct" grammar in creative ways, or include minor errors, but are still easily comprehensible by human listeners. Also, many sentences are structurally ambiguous according to grammar rules, but easily disambiguated by human listeners. In these cases, correct analysis may require lexical and distributional knowledge not found in hand-crafted grammar rules. Instead of attempting to encode this knowledge manually, which would be far too difficult, researchers have turned to corpus-based statistical techniques, in which lexical and distributional knowledge is gathered from large corpora of real human-generated sentences. In addition to increased accuracy, statistical parsers tend to exhibit greater robustness in dealing with unusual utterances, which would cause a more strictly rule-based parser to fail. They also have the advantage of being easier to build and to customize, because they do not require the work of a language expert to carefully design a grammar and patiently encode a dictionary. Instead, given an appropriate framework, all but the most basic grammar rules can be learned automatically from data, resulting in a huge savings in time and effort, especially if an existing parsing system is being ported to a new language or domain.

To train a statistical parser requires large quantities of annotated data. Typically, the training data must consist of real sentences annotated with structural information of the kind the parser will eventually generate. Unfortunately, annotating these sentences can require a huge amount of work by language experts, comparable to that required to develop a rule-based grammar. However, if a rule-based parser already exists for a given language, it can be used to automatically annotate real sentences, which can then be fed into a statistical parser. Thus we posit that a rule-based parser can be parlayed into a new parser with all the advantages of a statistical technique, without the need for a language expert.

Training a parser on the output of another parser is likely to be somewhat of a controversial idea, because some of the knowledge encoded in the handcrafted grammar will likely be lost in the transfer, which could result in a statistical parser that is less accurate than the rule-based parser from which it was trained. However, this technique has several advantages. Statistical parsing is more robust than rule-based methods, and can respond to difficult inputs with graceful degradation rather than sudden failure. This is not to say that there are no rule-based ways of increasing robustness, but that it comes naturally with statistical methods. Secondly, the use of real sentences in training actually adds information to the system, enabling it to make use of lexical and distributional knowledge ignored by the original rule-based system, which is especially useful in ambiguous cases. Even if the original parser is not always correct when generating training examples from ambiguous sentences, a large amount of training data should allow the statistical parser to overcome the noise and learn the patterns that occur most often. A third advantage of this technique is its versatility. Training data for a statistical parser could come not only from a single existing parser, but from various sources, if desired. For example, a few hand-coded examples could be added to introduce a new grammatical construct not found in the original parser, or the output of two or three parsers with different strengths and weaknesses could be combined to produce a new parser that combines their strengths.

When a parser is needed for a language in which no parser is yet available, the choice between carefully designing a rule-based parser or tediously annotating large amounts of data by hand in order to train a statistical parser is rather unappealing, since either approach requires a large amount of work by a language expert. However, perhaps bilingual resources can come to the rescue by enabling us to automatically annotate foreign sentences to use as training data for a new parser. If the relationship between the structure of the foreign language and a language for which a parser exists, such as English, can be systematically understood, then the structures generated by the English parser can be transferred over to the new language across an aligned bilingual corpus, resulting in a corpus of syntactically-analyzed sentences suitable for training a statistical parser in the new language. In order for this to work, a model is needed of the relationship between the syntactic structures of the two languages, along with an unsupervised learning technique to train the model based on a bilingual corpus where the syntactic structure is known for only one of the two languages. If a technique can be found to infer the structure of a given foreign sentence given the words and structure of its English translation, then the resulting foreign sentence structures can be used to automatically train a new parser for the foreign language in which no parser was previously available, without the need for language experts to design grammar rules or annotate data.

Since ancient times, the ability to translate from one language to another has been valued. The hope of creating an automatic system to do this translation is now closer to fruition than ever before, though for open-domain translation, the output quality of automated systems cannot yet come anywhere near that of bilingual human. Corpus-based statistical methods currently dominate machine translation research, because it is just too difficult to write rules that capture all the complexities of actual utterances, while relatively simple statistical models have brought systems a long way toward comprehensible translation. However, the models used in most statistical translation systems are much too simple to capture the syntactic structure of each sentence, often resulting in output that is ungrammatical or has the wrong meaning. Because important information is carried not only in the choice of words, but in their grammatical relationships, it is necessary to incorporate syntactic structure into translation models in order for meaning to be conveyed accurately and grammatically.

In this thesis, two new techniques are presented and then applied together in two different ways. First, a new statistical parser is presented. This parser, called Lynx, is trained from sentences annotated automatically by another parser, and achieves results comparable to the original parser. Second, a statistical model of the relationship between the syntactic structure of two languages is presented, along with an unsupervised training algorithm, titled LinkSet. Putting these two parts together, we then go on to demonstrate two applications. First, we automatically induce a parser for a foreign language across a bilingual corpus. Then we improve the output of an automatic translation system by incorporating syntactic structure into its translation and language models.

1.2 Thesis Statement

My thesis statement has 4 parts:

1. To train a new statistical parser, it is not necessary to hand-annotate a large corpus with structural information if a rule-based parser is available, because a parser trained on automatically-generated data can perform as well as or better than the original parser.

2. A statistical model of the relationship between the syntactic structures of two different languages can be effectively learned from a bilingual corpus by an unsupervised learning technique, even when syntactic annotations are available for only one of the languages.

3. Using a bilingual corpus and an existing parser in one language, a new parser can be automatically induced for the other language, without the aid of a language expert. This induced parser can then be incrementally improved.

4. Integrating syntactic structure into the statistical models used for automatic translation can increase the quality of translated output.

1.3 Overview

This section gives a high-level theoretical overview of the models developed throughout this thesis. Beginning with the structural model of English sentences used in the Lynx parser, discussed in Chapter 2, we then introduce the model of the relationship between English and a foreign language used in the LinkSet bilingual parser discussed in Chapter 3. Then the combination and application of these two models to the induction of a new foreign-language parser and to an automatic translation system, discussed in Chapters 4 and 5 (respectively) are outlined.

The Lynx parser uses a generative model $\Pr(\mathcal{E})$ of a structured English sentence $\mathcal{E}$, a syntactic tree structure including words and labels. Parsing a given English sentence string $\mathbf{e}$ is the same as finding the most likely tree $\hat{\mathcal{E}}_{\mathbf{e}}$ that yields the same surface string. Since each syntax tree $\mathcal{E}$ includes the words of the sentence, the probability of $\mathcal{E}$ given $\mathbf{e}$ is equal to the probability of $\mathcal{E}$ if the words of $\mathcal{E}$ match $\mathbf{e}$, and zero otherwise. To parse $\mathbf{e}$, we find $\hat{\mathcal{E}}_{\mathbf{e}}$ as follows:

$$\hat{\mathcal{E}}_{\mathbf{e}} = \operatorname*{argmax}_{\mathcal{E}} \Pr(\mathcal{E}|\mathbf{e})$$

Since the words $\mathbf{e}$ are contained in the structure $\mathcal{E}$, this is equivalent to limiting the range of the structure such that it must match the given string, in which case the conditioning on the probability upon $\mathbf{e}$ becomes redundant:

$$\hat{\mathcal{E}}_{\mathbf{e}} = \operatorname*{argmax}_{\mathcal{E}:\textsc{Words}(\mathcal{E})=\mathbf{e}} \Pr(\mathcal{E})$$

The model $\Pr(\mathcal{E})$, which assigns a probability to any structured English sentence, is recursively structured, breaking down the sentence structure into subtrees, and conditioning the probability of each node on the properties of its parent and immediately adjacent sibling (if any). Once it has been trained on annotated English sentences, this model can be used for parsing by searching the space of structures possible for a given sentence for the one to which the model assigns the highest probability.

When a structured English sentence is given along with its translation into a foreign language, their relationship can be modeled using the LinkSet structured translation model, which is a generative model $\Pr(\mathcal{F}|\mathcal{E})$ of the transformation of a structured English sentence into a structured foreign sentence. This structured translation model, which models the probability of a given foreign sentence tree $\mathcal{F}$ given an English sentence tree $\mathcal{E}$, is used by the LinkSet bilingual parser to find the most likely structure $\hat{\mathcal{F}}_{\mathcal{E},\mathbf{f}}$ for a given foreign sentence $\mathbf{f}$, when the words and structure of its English counterpart are known. Again, since the sentence structure $\mathcal{F}$ includes the words of the sentence, the probability of $\mathcal{F}$ given $\mathcal{E}$ and $\mathbf{f}$ is the same as the probability of $\mathcal{F}$ given $\mathcal{E}$ if the words of $\mathcal{F}$ match the string $\mathbf{f}$, and zero otherwise. Thus to find the most likely structure $\hat{\mathcal{F}}_{\mathcal{E},\mathbf{f}}$ of a a foreign sentence $\mathbf{f}$ given an English structure $\mathcal{E}$, we maximize the following expression:

$$\hat{\mathcal{F}}_{\mathcal{E},\mathbf{f}} = \operatorname*{argmax}_{\mathcal{F}} \Pr(\mathcal{F}|\mathcal{E},\mathbf{f})$$

As before, the words $\mathbf{f}$ are contained in the structure $\mathcal{F}$, so we can again simplify by stipulating the correspondence between the words $\mathbf{f}$ and the structure $\mathcal{F}$:

$$\hat{\mathcal{F}}_{\mathcal{E},\mathbf{f}} = \operatorname*{argmax}_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E})$$

In principle, these models of English sentence structure and of foreign sentence structure given English sentence structure can be composed to generate a model of foreign sentence structure $\Pr(\mathcal{F})$, which can then be used to parse foreign sentences monolingually. The probability of a foreign sentence structure can be defined in terms of the model of English structure $\Pr(\mathcal{E})$ and the model of foreign structure given English structure $\Pr(\mathcal{F}|\mathcal{E})$.

$$\Pr(\mathcal{F}) = \sum_{\mathcal{E}} \Pr(\mathcal{E}) \Pr(\mathcal{F}|\mathcal{E})$$

As it is given here, the probability of $\mathcal{F}$ is impractical to compute, since it requires summing over all (infinitely many) English structures. However, it can be approximated, as explained below in Chapter 4. Once we build a model of foreign structure, we can parse foreign sentences the same way we do English:

$$\begin{aligned}\hat{\mathcal{F}}_{\mathbf{f}} &= \operatorname*{argmax}_{\mathcal{F}} \Pr(\mathcal{F}|\mathbf{f}) \\ &= \operatorname*{argmax}_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F})\end{aligned}$$

Composed differently, the monolingual English structure model $\Pr(\mathcal{E})$ and the structured translation model $\Pr(\mathcal{F}|\mathcal{E})$ can in principle be used for translation, that is, to find the most probable English rendering of a given foreign sentence. Following the noisy-channel interpretation of statistical translation that has become standard, we pretend that the foreign sentence **f** is a transformed version of some "original" English sentence **e**. To translate, we simply "decode" the foreign signal by finding the English string **e** most likely to have generated the foreign string **f**:

$$\hat{\mathbf{e}}_{\mathbf{f}} = \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f})$$

Using Bayes' rule, we can turn this around and factor out the $\Pr(\mathbf{f})$ that would appear in the denominator, since it does not depend on **e**. Then translation from a foreign language into English requires only an English language model $\Pr(\mathbf{e})$ and a model $\Pr(\mathbf{f}|\mathbf{e})$ of translation from English to the foreign language.

$$\hat{\mathbf{e}}_{\mathbf{f}} = \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e})$$

So far we have described translation of sentence strings, ignoring syntactic structure. We can incorporate our structured models of English and of the relationship between English and a foreign language by summing over all foreign structures that are consistent with the foreign string we are translating, and over all English structures. To find the most likely English translation $\hat{\mathbf{e}}_{\mathbf{f}}$ of a foreign sentence $\mathbf{f}$, we maximize the following sum:

$$\hat{\mathbf{e}}_{\mathbf{f}} = \operatorname*{argmax}_{\mathbf{e}} \sum_{\mathcal{E}:\textsc{Words}(\mathcal{E})=\mathbf{e}} \Pr(\mathcal{E}) \sum_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E})$$

With this combined model, the best translation of a foreign sentence $\mathbf{f}$ can in principle be found. In practice, the sum over all English structures and over all foreign structures consistent with the given foreign sentence is infeasible to calculate. However, an approximation can enable translation hypotheses to be scored and reranked at a reasonable computational cost, as shown in Chapter 5.

The overview above serves as an outline of this thesis. We will first introduce a model of English sentence structure, along with a method for using it to parse English sentences. Using training data derived from an existing parser, we will train the model, and then test this new statistical parser by comparing its ability to bracket sentence constituents with that of the other parser, as measured against human-annotated bracketings. Then we will introduce a model of foreign sentence structure given English sentence structure, along with an unsupervised learning technique. Training this model with a bilingual corpus, we will test its ability to align sentences by comparing the most probable alignments according to the model, as compared to human-annotated alignments. Finally, we will use these models together in two different ways. First, we will compose them to generate a stand-alone model of foreign sentence structure, which we will then use to parse foreign sentences. We will test this new parser by comparing its accuracy in bracketing unseen foreign sentences with respect to human-annotated bracketings. Second, we will enhance an automatic translation system by using these structured models to reevaluate the hypotheses it generates, comparing the resulting translations against the original system's output using standard evaluation metrics such as the BLEU and NIST scores. Along the way, we hope to gain insight into how and why to integrate syntactic structure into statistical models of natural language.

Chapter 2

Monolingual Probabilistic Parsing

2.1 Motivation

Syntactic parsing is the analysis of a sentence for its syntactic structure. The grammar and lexicon of a language together define which structures are possible in that language and which words may go where in that structure. There are many applications of parsing, ranging from grammar checking in word processors and language modeling for tasks such as speech recognition, to natural language understanding tasks, including dialog systems, command systems, and translation systems. Parsing can also be useful for various kinds of corpus analysis and information extraction.

A common approach to parsing is to carefully encode the grammar and lexicon of a language into a computer system, allowing that system to find a valid structure (if one exists) for any given input. When ambiguous input arises, heuristics can be used to select a structure likely to be the correct one. However, encoding a grammar requires a great deal of effort by an expert in the language and in the technique of grammar encoding, expertise that may be hard to find, and this encoding must be done all over again whenever the system is ported to another language. While these rule-based systems encode the grammar rules implicitly known by speakers of the language being parsed, they typically do not make use of the semantic and distributional (statistical) knowledge speakers have, which is very important for selecting the right meaning of an ambiguous utterance. A third disadvantage of grammar-based parsing systems is that natural language often does not conform to the rules of the grammar. Unusual constructions, casual speech, innovative expressions, mistakes, noise, and interruptions can all result in sentences that are quite understandable to a human reader or listener, but utterly confusing to a rule-based parser.

Statistical parsing, on the other hand, can deal with each of these weaknesses. A statistical model is typically learned from data rather than being painstakingly constructed by a language expert. Also, a grammar learned from real data will match the actual forms seen in use, rather than an idealized grammar that may not correspond to everyday speech, or the idioms of the community for which the parser is trained. Lexical statistics allow a statistical parser to make subtle distinctions based on the specific words in the sentence, providing a more principled way to deal with ambiguity than the heuristics use in rule-based systems. Robustness is another advantage of statistical systems: since parsing is a maximization problem, a sentence that is not well-formed according to a rule-based system (resulting either in total failure or in skipping part of the sentence) will just have a lower probability for its best structure in a statistical system, resulting in graceful degradation rather than sudden failure when the going gets tough. Even better, a statistical system may be able to correctly handle unusual constructions not seen before (by the system designer or in the training data) just by choosing the most probable way to analyze a given phrase.

While a statistical parser can be trained from data, it is not necessarily easier to obtain a large quantity of hand-annotated sentences than to design a grammar by hand. However, there are various shortcuts. One approach is to use a hand-designed grammar, but then enhance it with statistics. In this case, less hand-annotated data should be necessary. This approach might give good results, but it still requires an expert to design a grammar and annotate a good deal of data by hand.

Another approach is to use an existing rule-based parser to automatically annotate data. This amounts to transferring a grammar from rules to statistics, and of course requires that a rule based parser already exist; however, it does add distributional information from real-life data, giving many of the advantages of a statistical parser. One disadvantage is that the rule-based parser will make mistakes, adding noise to the statistics of the new parser; however, one would hope that the noise would be insignificant compared to the signal. While it may not require a language expert to build a statistical parser using an existing parser, this does not yet solve the problem of building a parser for a language for which one does not yet exist. However, it is halfway there: if training data can be generated in the target language, a statistical parser for that language can be trained with little additional effort, given a general-purpose statistical parser. This process is the subject of Chapters 3 and 4. Meanwhile, we will see that in English a statistical parser can offer some advantages over a rule-based parser, even when the statistical parser's training data was all generated by the rule-based parser against which it is being compared.

How should such a statistical parser be designed? The most important consideration is the model itself. It must be able to capture the relevant linguistic phenomena, but also to generalize from a reasonable amount of training data, not requiring too much data nor taking up too much space, but dealing with sparse training data by capturing the relevant features. A secondary consideration is the search algorithm. The model must be designed so that search can be done efficiently. Fortunately, natural language has a hierarchical structure, the locality of which is a key both to efficient search and to generalization from sparse data. A recursive generative model of sentence structure, picturing a sentence as a tree, can capture this locality by conditioning the probability of each node only on nearby nodes.

2.2 Related Work

2.2.1 Background: Link Grammar

A *link grammar* is a formal grammatical system, such that a sequence of words is in its language if there is a way to draw links between words so the local requirements of each word are satisfied, the links do not cross, and the words form a connected graph. The idea was introduced by Sleator and Temperley, who also encoded English grammar as a link grammar and developed software, which will be referred to as the Link Parser, to parse sentences using this grammar [25].

In a link grammar, each word (or group of similar words) in the dictionary has local linking requirements which must be satisfied in a well-formed *linkage*. These requirements consist of lists of labeled *connectors* branching either to the left or the right, which must meet connectors with matching labels coming from other words in the sentence. Multiple *disjuncts*, as these lists of connectors are called, may be valid options for a word.

For example, Figure 2.1 shows a few words or word classes from a simplified dictionary. Each word has various connectors on either side, each of which must be joined with a matching connector in order to build a well-formed linkage. The nouns in this example can act either as objects or subjects but not both, so one and only one of the connectors **O** and **S** must be satisfied for these words, a fact not shown in the diagram, but nevertheless encoded in the dictionary. In a real dictio-

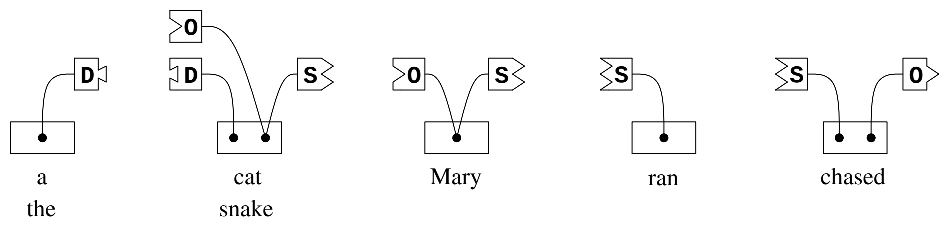

Figure 2.1: Part of a dictionary.

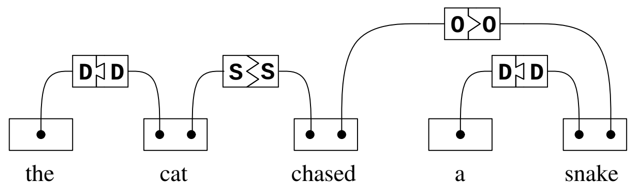

Figure 2.2: A sentence.

nary, some words may have only one possible disjunct, while others have many possible combinations of connectors. In Figure 2.2 some words are assembled into a linkage, which corresponds to the sentence "the cat chased a snake." Each connector is satisfied and no links cross, and all the words are connected, so this is a well-formed linkage.

Link grammars are very similar to dependency grammars, in which each word (except the *root word*) is linked by a directed arc to one other word. A linkage, on the other hand, is an undirected graph, and may contain cycles. Both link grammars and dependency grammars are *context-free*.

The Link Parser uses a dynamic programming technique to find a linkage that satisfies some disjunct of each word in a sentence, forming a connected graph with no crossed links. A range of words is parsed by first selecting a word within that range that connects to one of the connectors available from either the left or right end of the range, and then recursively parsing the sub-ranges on each side of the chosen word. The asymptotic running time of this algorithm is $O(n^3)$ to parse an n-word sentence. For long sentences, this is still quite slow, so some advanced pruning techniques are used to eliminate connectors that will never be satisfied, resulting in a significant speed-up in the average case.

2.2.2 Statistical Models based on Link Grammar

Several others have adapted the link grammar formalism to build statistical models of language. These models combine the advantages of lexical statistics and long-range dependencies, setting the stage for the more complex Lynx parser.

Grammatical Trigrams

Lafferty, Sleator, and Temperley, in 1992, introduced a generative grammar model, called Grammatical Trigrams, which combined the strengths of trigrams and a link grammar [17]. The key advantage of this combination is that the result is both highly lexical and able to capture long-range dependencies beyond the reach of ordinary trigrams.

The Grammatical Trigrams model generates the sub-range bordered by words L and R on the left and right, respectively, and having incoming connectors l and r, by generating a word w, a disjunct d to determine what connectors will branch from w, and an orientation o which determines whether w will be linked to l, r, or both. A disjunct is a pair of ordered lists of connector labels $d = ((l_m, l_{m-1}, ..., l_1), (r1, r2, ..., r_n))$ determining which links will branch to the left and right of the word to which it attaches. The probabilities of w, d, and o are conditioned on L, R, l, and r.

The Grammatical Trigrams model could be trained from example linkages (of the same kind used to train the Lynx parser) using the Inside–Outside (EM) algorithm. Methods of smoothing the model were also suggested. No concrete results were reported.

Inference and Estimation of a Long-Range Trigram Model

The Grammatical Trigrams model was adapted in 1994 by a group at IBM, resulting in a probabilistic link grammar, which they used as a language model to a achieve slight improvement in they output of a translation system [23]. This extended trigram model generates the next word in a sentence by first deciding whether to halt, step, or branch, conditioned on the previous two words. When branch is selected, an additional word is generated according to a long-range trigram model; this additional word will be linked to the current word, but will appear to the right of all words subsequently generated from this word. For both step and branch, a word is generated according to a regular trigram model for placement to the immediate right of the current word. Like ordinary trigrams, the grammatical trigram model is trained automatically from data. While the model does not label its links, nor is it expressive enough to generate every structure allowed by link grammar, it is a significant step forward from ordinary trigrams, and shares their advantage of not needing an explicit grammar.

Restricted Probabilistic Link Grammars

Much like the Grammatical Trigrams model, the Restricted Probabilistic Link Grammar (RPLG) of Fong and Wu [19] generates words, orientations, and disjuncts based on (L, R, l, r). Like the Long-Range Trigram Model, RPLG treats all links the same (ignoring labels), which greatly reduces the space of possible disjuncts, thus making learning more feasible but at the same time reducing the predictive power of links within the model. RPLG also adds a dependency on orientation to word probabilities, which increases the accuracy of word predictions. Orientation is restricted so that if connectors are available on both sides, either the left or both left and right may be grabbed, but not just the right, thus preventing a single sentence graph from having multiple derivations. Just like the Grammatical Trigrams model, the RPLG model was trained using the Inside–Outside EM algorithm. A parsing experiment shows improvement over the Grammatical Trigrams model.

Structure and Performance of a Dependency Language Model

Again motivated by the need for a better language model than simple trigrams, although this time for speech recognition rather than translation, a 1997 summer workshop at Johns Hopkins designed and implemented a maximum entropy dependency-grammar model [8]. As in the previous models, this one generates each word and its disjunct, with probability conditioned on a subset of the words and disjuncts to its left. In this case, maximum entropy training is used to select features from the following set: the immediately previous words (as in ordinary trigrams), any connector labels branching out from a previous word but not yet connected, and the source words for those unsatisfied connectors. This formulation gives the probability of the structure given the sentence, or of the sentence given the structure, but not of their joint probability. Using an existing dependency parser, they compared several variants of this model as the language-modeling component of a speech recognition system.

2.2.3 Other Statistical Grammar Models

Although based on phrase-structure grammar rather than link grammar, three generative parsing models proposed by Michael Collins in 1997 [9] have significant similarity to the Lynx model. As in Lynx, his Model 1 parser finds the most probable parse tree for a sentence by maximizing the joint probability of the sentence and its parse tree. Model 1 is trained on a tree bank of examples rather than using an explicit grammar, but it achieves higher precision and recall than competing parsers with which it was compared, partly because it is trained on hand-built examples rather than an errorful automatically-parsed training set. It is a generative lexicalized probabilistic context-free grammar, distinguishing internal nodes from leaves, which hold the words. In contrast, the link grammar formalism has no internal nodes. In Model 1, internal nodes are lexicalized, marked with the word that is the *head* of the phrase represented by that subtree. Collins expanded Model 1 with additional non-terminal symbols and gaps to create two more models with slightly better accuracy.

Similarly, Eisner had proposed three dependency grammar models in 1996 [11], and developed them further the following year [12]. Model C, which will be considered here, models unlabeled dependencies between words, as did the Long-Range Trigram Model and RPLG. Unlike RPLG and Grammatical Trigrams, Model C generates links with their corresponding words and subtrees one at a time rather than as a monolithic disjunct for each word. Also, each word is marked with a category tag, upon which other probabilities depend, thus reducing the effects of lexical sparsity. Model C recursively generates subranges of a sentence by generating a word and at the same time proceeding to generate the subrange rooted at that word, which consists of a sequence of links, each with its associated tag and word, and then, recursively, any subtree rooted at that word. The sequences of links generated on the left and right side of a word are mutually independent, and Markovian independence is assumed among links, making the probability for each link depend on the previous link generated but not on any other links generated by the word.

In a detailed exposition of his parsing algorithm for bilexical grammars [13], which condition grammar rules (probabilities) on pairs of words, Eisner noted a strong connection between his method and the Link Parser, both of which require time $O(n^3)$ to parse a sentence of n words.

2.3 The Lynx Model

Based on the link grammar formalism to define the types of structures it can model, the Lynx parser is a very different creature, using automatically-learned statistics rather than hand-crafted rules to define its grammar. Hence the name LYNX, which stands for **L***inks,* **Y***et* **N***ot e***X***actly*.

The core the of Lynx parser is a generative model of a sentence with syntactic structure. We notate an English sentence as **e**, and the sentence together with its structure as $\mathcal{E}$. Once this model is defined and trained, parsing a sentence is just a matter of selecting the most probable structure for that sentence.

$$\hat{\mathcal{E}}_{\mathbf{e}} = \operatorname*{argmax}_{\mathcal{E}:\textsc{Words}(\mathcal{E})=\mathbf{e}} \Pr(\mathcal{E})$$

Of course there are myriad ways the model $\Pr(\mathcal{E})$ could be formulated, some better than others. A good model must be sensitive enough to capture the relevant linguistic phenomena, but concise enough to overcome the challenges of training data sparsity, limited memory space, and limited computation time. The Lynx model is a recursive generative model of sentence structure, in which the probability of generating a node in the syntax tree is conditioned only on features of nearby nodes, in particular the parent and the next elder sibling.

In order to define the model in more detail, it will be useful to establish some notation. A structured English sentence $\mathcal{E}$ is a collection of nodes, arranged in a tree structure. A node ν (nu) is a tuple $\langle c, t, w \rangle$, where $\nu.w$ is a word, $\nu.t$ is a part-of-speech tag assigned to that word, and the link between node ν and its parent has the connector label $\nu.c$. The children of each node are ordered left to right and may fall on either side of the word associated with the node. That is, the nodes of the tree are arranged in a total ordering from left to right, which corresponds to the order of the words in the sentence and to an inorder traversal of the tree. The function $\textsc{Words}(\mathcal{E})$ returns the words of $\mathcal{E}$ in order. The root of the tree is a special symbol call the *left wall*.

To facilitate tree operations, we use the following functions: $\textsc{Parent}(\nu)$, $\textsc{Children}(\nu)$, and $\textsc{Elder}(\nu)$. Each returns a (possibly empty) set: respectively, the parent, children, or immediate elder sibling of ν. They return a set of nodes (or references to nodes). Each node except the root has a unique parent. The children of a node are all the nodes that have that node as their parent. The unique immediate elder sibling of ν, if it exists, has the same parent as ν and is farther from the parent than ν is, according to the total ordering on nodes, and there is no other

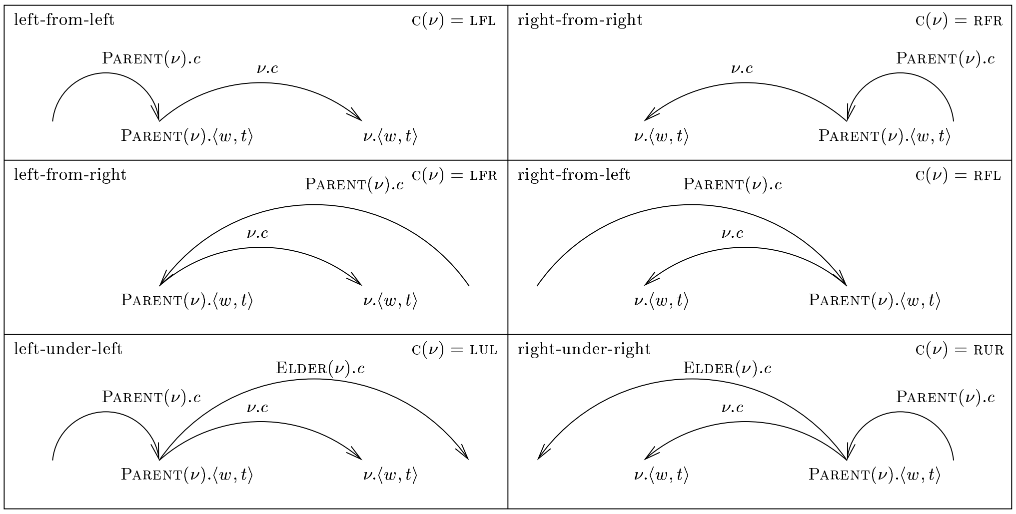

Figure 2.3: The six contexts of connectors, words, and tags.

node closer to ν that has these properties. With these definitions, it is possible to order the nodes such that each node is preceded by its parent and elder, although this ordering does not necessarily correspond to the linear order of words in the sentence.

Using this notation we can write an equation for the probability of a structured sentence, showing the main independence assumption of the model:

$$\Pr(\mathcal{E}) \quad = \quad \prod_{\nu \in \mathcal{E}} P\left(\nu | \textsc{Parent}(\nu), \textsc{Elder}(\nu)\right)$$

This equation means that in this model the probability of a sentence and its structure is the product of the probabilities of the individual nodes and words, and that the probability of each node and its corresponding word is conditioned on the parent and elder sibling of the node, and on the word associated with the parent, not on other nodes or words. That is, the model exhibits locality in the tree structure.

To break down the model further, let us distinguish six different contexts in which a node may occur, shown in Figure 2.3. The context of a node ν is a deterministic function of its parent and elder, with the range $\{$LFL, RFR, LFR, RFL, LUL, RUR$\}$ and which we will notate as $\mathrm{C}(\nu)$. To simplify the equations below, we also define two sets $\mathrm{C}_t = \{\mathrm{LFL}, \mathrm{RFR}, \mathrm{LFR}, \mathrm{RFL}\}$ and $\mathrm{C}_u = \{\mathrm{LUL}, \mathrm{RUR}\}$.

$$
\begin{aligned}
P(\nu|\ldots) &= P(\nu|\textsc{c}(\nu), \textsc{Elder}(\nu).c, \textsc{Parent}(\nu).\langle c,t,w\rangle) \\
&= P(\nu.c|\ldots)\, P(\nu.t|\nu.c,\ldots)\, P(\nu.w|\nu.t,\nu.c,\ldots) \\
P(\nu.c|\ldots) &= P(\nu.c|\textsc{c}(\nu), \textsc{Elder}(\nu).c, \textsc{Parent}(\nu).\langle c,t,w\rangle) \\
&= \begin{cases} P(\nu.c|\textsc{c}(\nu), \textsc{Parent}(\nu).\langle c,t,w\rangle) & \textsc{c}(\nu) \in \textsc{C}_t \\ P(\nu.c|\textsc{c}(\nu), \textsc{Elder}(\nu).c, \textsc{Parent}(\nu).\langle t,w\rangle) & \textsc{c}(\nu) \in \textsc{C}_u \end{cases} \\
P(\nu.t|\ldots) &= P(\nu.t|\nu.c, \textsc{Parent}(\nu).w) \\
P(\nu.w|\ldots) &= P(\nu.w|\textsc{Parent}(\nu).w, \nu.c, \nu.t)
\end{aligned}
$$

Thus the model is broken down into sub-models which are limited in what they are conditioned upon.

2.3.1 Smoothing

Several of the sub-models listed above, however, are still conditioned on enough variables that they would have problems with memory requirements and data sparsity if they were not in turn composed of smaller, simpler models. Thus several sub-models are linear combinations of two or more basic models, shown below with λ's as mixing parameters; these λ's can be hand-tuned and should sum to one within each group.

$$
\begin{aligned}
P(\nu.t|\ldots) &= \lambda_1 P(\nu.t|\textsc{c}(\nu), \nu.c) + \lambda_2 P(\nu.t|\textsc{c}(\nu), \textsc{Parent}(\nu).w) \\
P(\nu.w|\ldots) &= \lambda_1 P(\nu.w|\nu.c, \nu.t) + \lambda_2 P(\nu.w|\textsc{Parent}(\nu).w, \nu.t) + \\
&\quad \lambda_3 P(\nu.w|\nu.t) \\
P(\nu.c|\ldots) &= \begin{cases}
\begin{array}{l} \lambda_1 P(\nu.c|\textsc{c}(\nu), \textsc{Parent}(\nu).c) + \\ \lambda_2 P(\nu.c|\textsc{c}(\nu), \textsc{Parent}(\nu).t) + \\ \lambda_3 P(\nu.c|\textsc{c}(\nu), \textsc{Parent}(\nu).\langle c,t\rangle) + \\ \lambda_4 P(\nu.c|\textsc{c}(\nu), \textsc{Parent}(\nu).\langle c,w\rangle) \end{array} & \textsc{c}(\nu) \in \textsc{C}_t \\
\\
\begin{array}{l} \lambda_1 P(\nu.c|\textsc{c}(\nu), \textsc{Elder}(\nu).c) + \\ \lambda_2 P(\nu.c|\textsc{c}(\nu), \textsc{Parent}(\nu).t) + \\ \lambda_3 P(\nu.c|\textsc{c}(\nu), \textsc{Elder}(\nu).c, \textsc{Parent}(\nu).t) + \\ \lambda_4 P(\nu.c|\textsc{c}(\nu), \textsc{Elder}(\nu).c, \textsc{Parent}(\nu).w) \end{array} & \textsc{c}(\nu) \in \textsc{C}_u
\end{cases}
\end{aligned}
$$

The context $\textsc{c}(\nu)$ is really a model selector, not a parameter. Thus we have a model that is broken down into pieces of practical size. The sub-models for $\nu.t$,

$\nu.w$, and $\nu.c$ have no components that condition on more than two parameters, allowing it to avoid serious problems with data sparsity and excessive memory requirements.

2.3.2 Link Grammar and Cycles

The Lynx model is based on link grammar (described in Section 2.2.1) in that it is a generative model of linkages having the same format as those described by link grammar. Link grammar describes undirected graphs, while the Lynx model describes directed trees. To translate an undirected linkage into a directed tree, the graph is traversed starting at the *left wall* (which becomes the root of the tree) assigning the direction of traversal to each edge followed, resulting in a directed tree in which any node can be reached from the root. Trees are inherently acyclic, and since the undirected graphs described by link grammar may contain cycles, these cycles must be dealt with before converting the graph into a Lynx-style tree. The simplest solution is to cut them. We do this by removing the longer of the two links that connect to the rightmost word in the cycle. Inspection of numerous real sentences containing cycles shows that this is the cut that most often would result in the smallest loss of useful information.

At an earlier stage in the research, we attempted to model these cycles instead of cutting them. One solution was to generate the rightmost node of a cycle conditioned on both incoming links (in contrast, other nodes are each conditioned on a single parent node), and to model links that are part of a cycle separately from links that are not. This approach, while maintaining linkages in a shape exactly equivalent to that of link grammar (and thus of the Link Parser), and being basically effective at modeling sentences with cycles, nevertheless had some significant problems. Although it was still a proper generative model, the model was made more complex and confusing. Training the model took hours instead of minutes, because some parameters had to be summed over all possible pairs of connectors. This approach also made search more difficult, not only making the algorithm more complicated, but making the search less able to use the model as a guide, resulting in a choice between a slower or a less accurate search. Because of these problems and the additional problem of dealing with cycles in a bilingual parsing model, this approach was abandoned (after much suffering) in favor of the much simpler cycle-cutting approach.

2.3.3 Training

The simplest way to train the Lynx model is to simply use empirical counts from a large corpus of linkages. In theory it would be possible, and perhaps preferable, to use an EM approach to train the parameters of this model, but in practice the simple approach works reasonably well, while the small improvement EM would probably provide is likely not worth its considerable computational cost.

Large annotated corpora are available in other formalisms (notably context-free grammar), for example the Penn Treebank, but these cannot be used to train the Lynx parser because there is no well-defined way to translate from CFG to link grammar. This difficulty is due to the ambiguity inherent in the annotations used for the Treebank, resulting in more than one possible linkage consistent with each annotated Treebank sentence. Translating from link grammar to CFG is similarly difficult, due to ambiguities in link grammar with respect to the annotations used in the Treebank.

Since no large corpus of hand-compiled linkages exists in link grammar, the next best thing is to parse text using the Link Parser to automatically generate a corpus of linkages. Of course, a machine-parsed corpus will be a less reliable source of knowledge, but there is reason to hope that the noise introduced by parsing errors will not overwhelm the signal, especially if a large amount of data is used. One way to improve the odds of this is to use a corpus that has been hand-analyzed in a different formalism (since such a corpus does exist) and to either translate syntactic structures from that formalism into link grammar, or use them to constrain the Link Parser in some way. Since direct translation is difficult, we adapted the Link Parser to select the linkage that corresponds most closely in terms of bracketing structure with the hand-annotated structure, when such annotations were available.

2.4 Search

The goal of parsing is to find the most likely structure of a given sentence string. Since we have a model of sentence structure that includes the string (a generative joint model), we could in theory enumerate every structure consistent with the given string, evaluate the probability of each, and select the most probable. Unfortunately, this is obviously infeasible because of the huge variety of possible structures. Fortunately, the model has been designed with locality in terms of hierarchical structure, so that substructures that exist in multiple sentence trees can

procedure PARSESENTENCE($sentence$)
 return (CHOOSELINKANDWORD(LEFTWALL, NONE))

procedure CHOOSELINKANDWORD(L, R)
 if $L + 1 = R$
  then return (1)
 for each $word \in L..R$
  do {
   for each $tag \in tags[word]$
    do {
     $cset \leftarrow connectors[L, word] \cup connectors[R, word]$
     for each $connector \in cset$
      do $heap.\text{INSERT}\begin{pmatrix} \langle connector, tag, word \rangle, \\ \Pr(connector, tag, word|L, R) \end{pmatrix}$
    }
  }
 $prob \leftarrow 0$
 $limit \leftarrow \text{MAX}(\epsilon, \gamma \times heap.\text{PEEK}(\langle connector, tag, word \rangle))$
 while $heap.\text{POP}(\langle connector, tag, word \rangle) \geq limit$
  do {
   $subprob \leftarrow \begin{pmatrix} \Pr(connector, tag, word|L, R) \times \\ \text{CHOOSELINKANDWORD}(L, word) \times \\ \text{CHOOSELINKANDWORD}(word, R) \end{pmatrix}$
   $prob \leftarrow \text{MAX}\ (prob, subprob)$
  }
 return $(prob)$

Figure 2.4: The Lynx parsing algorithm, simplified.

share the cost of computing their probabilities. The most probable subtree of each node is the same regardless of what happens elsewhere in the tree.

Based on this recursively-structured model, a simple search algorithm would be to enumerate all values for a node that might be the root of a tree covering a particular range of words (starting with the entire sentence), and then recursively search for the best subtree given each value of the node. The best subtree of each node (and its probability) is cached, greatly reducing the computational burden compared to a naive exhaustive search.

However, the above search is still both exhaustive and expensive. While a complete search is good in that it guarantees that the best structure (according to the model) will be found, it is too slow for many applications. Fortunately, the model itself holds the key to directing the search intelligently. When a search path is being selected, the model can be used to search the most promising path first, that is, the path with the most probable first step. Any subtree less probable than a known alternative need not be searched any further, so finding good paths early on will speed up search even without sacrificing completeness.

At each juncture, it is common for there to be a wide range of possibilities, but only a few with reasonable probability. A lot of time could be saved without introducing many search errors by not considering those options that are far less likely. For example, the search could ignore any step having a probability below some global threshold. This is known as a beam search. Unfortunately, setting such a threshold does not work in this case because at some points in the search the probability distribution is quite smooth (requiring a low cutoff) and at others it is quite sharp (requiring a higher cutoff). The answer to this is to use a cutoff that is proportional to the most probable option at each juncture rather than a fixed threshold. This is known as a *dynamic beam search*, a variant of the beam search method popular in speech recognition [18, 21]. The ratio between the highest probability option and the lowest probability that will be considered at that juncture is known as the *beam ratio*, which we notate here as γ, such that $p_{min} = \gamma \cdot p_{max}$. Adjusting the beam ratio adjusts a trade-off between speed and completeness: $\gamma = 1$ yields a greedy search, and $\gamma = 0$ a complete search.

Figure 2.4 shows the search algorithm used in the Lynx parser, simplified for readability. Given a range of words, it returns the probability of the most probable structure that produces those words. It has two main steps. First, it gathers all the possible directions the search could take from this point, and finds the probability according to the model of the first step of each. A step in this case is the selection of a word from within the given range to be generated next, a POS (part of speech) tag for that word, and a connector from among those that might possibly connect this word to the word on its right or left. Second, it goes through the gathered possibilities from the most likely to the least (until the dynamic beam cutoff is reached), recursively searching the sub-ranges to the left and right of the selected word. In the real implementation, the best structure is of course returned along with its probability, and there are some additional optimizations.

2.5 Results

The experiments below show that the Lynx parser performs about as well as the Link parser, which is an interesting result because all the training data used as input to the Lynx parser was output from the Link parser. The two systems score quite differently on a sentence-by-sentence basis, indicating that the Lynx parser is not just learning to imitate the Link parser, but to outdo it on many sentences.

Two data sets were used in the following experiments, both taken from the *Wall Street Journal*. Long sentences and sentences containing conjunctions and were removed. Also, some other problematic sentences were removed, such as those containing strange characters. The *WSJ-1989* data set consists of 14817 sentences from 1989. The *WSJ-1987* data set is much larger, consisting of 282879 sentences from 1987.

2.5.1 Quantitatively comparing Link vs. Lynx

To quantitatively evaluate the Lynx parser, we took text that had been annotated for constituent structure from the *Penn Treebank*, and computed the precision and recall of constituent structures found by Lynx with respect to the hand-annotated structures. For comparison, we also evaluated the output of the Link parser in the same way on the same sentences.

The comparison was performed using 10-fold cross-validation. To test each fold, the Lynx parser was trained using sentences from the other 9 folds, so it was never tested on sentences on which its current model had been trained. To train each model for the Lynx parser, training sentences were parsed using the Link parser. Annotations were available for these sentences as part of the *Treebank*, so it would have been nice to be able to use those directly instead of having to rely on an automatic parser to generate training data. However, the formalism used in the *Treebank* can not be translated simply and directly into link grammar, so instead we used the annotations to select the best among the various hypotheses generated

	recall	precision	F_1-score
Lynx	$68.43\% \pm 0.23\%$	$73.29\% \pm 0.15\%$	$70.77\% \pm 0.19\%$
Link	$67.60\% \pm 0.18\%$	$74.97\% \pm 0.12\%$	$71.09\% \pm 0.15\%$
Nulls	$68.89\% \pm 0.19\%$	$74.72\% \pm 0.14\%$	$71.68\% \pm 0.17\%$
Oracle*	$72.38\% \pm 0.19\%$	$78.90\% \pm 0.13\%$	$75.50\% \pm 0.16\%$
Link-opt*	$72.17\% \pm 0.18\%$	$79.53\% \pm 0.16\%$	$75.67\% \pm 0.15\%$

Table 2.1: Comparison of the Link and Lynx parsers on the *WSJ-1987* data set with simple and optimal hybrid approaches, plus the filtered system used to generate training data. Means and standard deviations are shown for recall, precision, and F_1-score. Systems that "cheat" are marked with an asterisk.

by the Link parser when parsing each sentence, and then trained the Lynx parser on these selected parse structures.

Table 2.1 shows the aggregate precision and recall of the Link and Lynx parsers on both data sets, along with two hybrid approaches, which simply select the output of either parser on a sentence-by-sentence basis. The first hybrid approach, *Nulls*, uses a very simple heuristic, selecting the Link parser for sentences which it is able to analyze completely, and selecting the Lynx parser whenever the Link parser requires **null** links to complete its analysis. The second hybrid approach is a cheating oracle, which always knows which parser will do better; it is included to show an upper bound on the performance of sentence-level hybrids. Also shown is a cheating version of the Link parser, *Link-opt*, which looks at the correct answer when selecting the best among its hypotheses. The output of this version was used as training data for the Lynx parser.

For all the percentages in Table 2.1, the standard deviations (over 10 trials) are under a quarter of a percentage point. All pairwise comparisons that can be made from this table are statistically significant at $p = 0.001$, with one exception: the comparison of Link vs. Nulls on precision is significant at $p = 0.01$.

As these results show, the Lynx parser performs about as well as the Link parser, yet the two systems score quite differently on a sentence-by-sentence basis. On the large *WSJ-1987* dataset, the Lynx parser achieves a slightly higher recall than the Link parser, at only slightly less precision. This is particularly interesting because all the training data used by the Lynx parser came from the Link parser.

Figure 2.5 shows comparisons of the Lynx parser vs. the Link parser on the

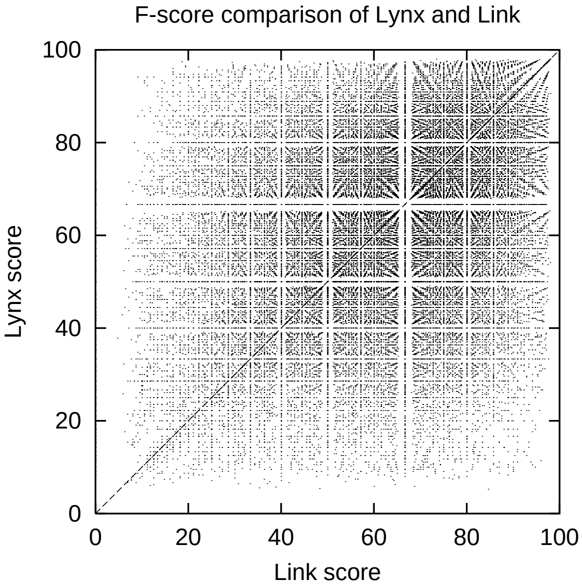

Figure 2.5: Sentence-by-sentence comparsion of the Link parser and the Lynx parser by F_1-score on the *WSJ-1987* dataset.

respective data sets. For each sentence, they compare the F_1-score[1] of the Link parser (on the X axis) against the F_1-score of the Lynx parser (on the Y axis). Any points on the diagonal ($x = y$) indicate sentences in which both parsers got the same score, while sentences above the diagonal were analyzed more correctly by the Lynx parser, and sentences below the diagonal were analyzed more correctly by the Link parser. The grid-like structure of the scatter plots appears because each sentence must have a whole number of brackets, limiting the possible scores to a range of fractions where both the numerator and denominator are integers less than twice the number of words in the sentence.

Figure 2.6 shows the F_1-score of each of the parsers and hybrids for each of the 10 folds in the cross-validation experiment, with 10 four-way comparisons in an arbitrary order.

While the plot in Figure 2.5 shows that there are many sentences for which one parser does better, and many where the other does better, the way these are distributed can be seen quantitatively in Table 2.2, which shows the number and percentage of sentences for which each system out-scores each other system. Each

[1] $f_1 = 2pr/(p + r)$

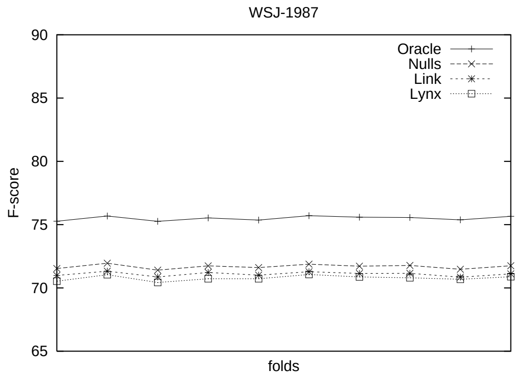

Figure 2.6: Comparsion of the Link and Lynx parsers and two hybrids by F_1-score on the *WSJ-1987* dataset, broken down into 10 trials.

	Link		*Lynx*		*Nulls*		*Oracle*	
Link			84967	30%	28066	10%	84967	30%
Lynx	85078	30%			65284	23%	85078	30%
Nulls	19794	7%	56901	20%			76695	27%

Table 2.2: The number and percentage of sentences for which the column method beats the row method on the *WSJ-1987* dataset.

	WSJ-1989				*WSJ-1987*		
	zero		perfect		zero		perfect
Link	791	5.3%	2467	16.6%	6874	2.4%	0
Lynx	718	4.8%	1885	12.7%	4765	1.7%	0

Table 2.3: The number of sentences for which each parser scored zero or 100%.

cell represents the number of sentences on which the system at the head of its column scores higher than the system at the start of its row. No system can outscore the *Oracle* (by definition), so it appears only among the columns. On 40% of all sentences, the parsers scored equally well.

Points on the borders of the scatter plots represent sentences for which at least one system got either a perfect score or a zero. Quite a few points fall into these categories, as quantified in Table 2.3. As this table shows, the Link parser is both more likely to get a zero score and more likely to get a perfect score than the Lynx parser. On the more difficult *WSJ-1987* corpus, neither parser bracketed any sentence perfectly, and the Lynx parser had just over half the failure rate of the Link parser. One of the keys to Lynx's success is its increased coverage due to the graceful degradation of a statistical parser, in contrast to the rigidity of a knowledge-based parser. Because of its smoother performance curve, the Lynx parser is less likely to fail completely when faced with a difficult sentence.

As shown in Figure 2.5 and Table 2.2, there are many sentences on which one parser does better, and many on which the other does better, with the data almost equally divided between the two. This shows that each parser is getting different things right (and different things wrong), motivating the use of a hybrid approach, which selects the more accurate parser for each sentence in order to give better results than either parser alone. As shown in Table 2.1, the trivial hybrid *Nulls*, which uses the Link parser for each sentence which it can analyze without skipping any words, and uses the Lynx parser otherwise, does slightly better than either parser alone. The *Oracle* hybrid, which is not a real system but shows the best that could be done by selecting between Link and Lynx on a sentence-by-sentence basis, outperforms *Nulls* by several percentage points, showing that a better selection scheme could offer significant improvement over either of the two parsers. Again, this shows that the Lynx parser is accurately parsing sentences that were missed by the Link parser, even if it is also missing some sentences correctly parsed by the Link parser.

source	recall	precision
unfiltered	76.0%	74.5%
filtered	77.3%	75.4%

Table 2.4: Recall and precision of the Lynx parser trained with filtered and unfiltered data, on *WSJ-1989*

2.5.2 Quality of training data

In the above experiment, the training data for Lynx was all generated using *Link-opt*, a version of the Link parser that selects from among the possible linkages of a given sentence the structure most consistent with a given constituent structure. Hand-annotated constituent structures were taken from the *Penn Treebank* for this purpose, and were also used to evaluate the results of the parsers and hybrids. Using this optimized version of Link to train Lynx is a kind of semi-supervised learning, since hand-annotated data was used, but could not be converted directly into training data. In order to measure the effect of this hand-annotated data, the *WSJ-1989* data set was parsed twice, using training data generated with and without the benefit of hand-annotations. Ten-fold cross-validation was used, with each tenth being evaluated on a model trained on the other nine tenths. The results are shown in Table 2.4. Filtering the training data based on hand-annotations gives an improvement of about one percentage point. This comparison is statistically significant at $p = 0.05$, a somewhat weak significance result but enough to indicate that filtering is worthwhile given appropriate data.

2.5.3 Training Size

To see how much training data the Lynx model needs for best results, we built a series of models from differing amounts of training data, testing each model for constituent bracketing based on the same held-out set of 552 hand-parsed *WSJ* sentences from the *Penn Treebank* to produce the recall curve in Figure 2.7. The corresponding precision curve has approximately the same shape.

Once training size reaches about 30,000 sentences, the learning curve levels off and additional training data does not give any additional increase in accuracy. Since the data added later is from articles farther away (temporally) from the test set, it is likely that it is less similar to the test data and therefore less useful for parsing it. It seems reasonable that having training data that is similar in style

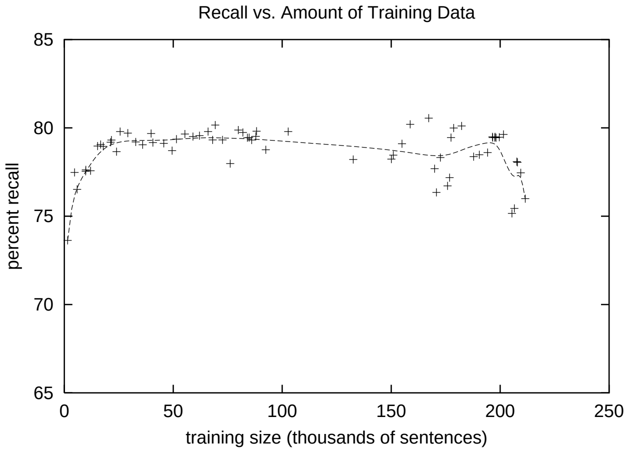

Figure 2.7: Lynx recall vs. training size

and vocabulary to the test data is more important than having large quantities of training data.

2.5.4 Beam Ratio

As mentioned above (Section 2.4), the beam ratio is a search parameter that selects a trade-off between speed and accuracy. A beam ratio of 1 would give a greedy search, and a beam ratio of 0 would give a complete search. Figure 2.8 shows this trade-off at several different beam ratios, with the F_1-score of 552 sentences evaluated for constituent bracketing compared with the time required to parse the entire test set (using a relatively slow computer). As expected, smaller beam ratios resulted in both longer parsing times and higher accuracy.

2.5.5 Examples

Figures 2.9 and 2.10 show the same five sentences parsed by the Lynx and Link parsers, respectively. These sentences were selected because they demonstrate typical constructions on which the Lynx parser does better than the Link parser.

- *This time around, they're moving even faster.*
  The Link parser doesn't know the idiom "this time around," and it does not allow the proposition *around* to appear without an object, so it resorts to a

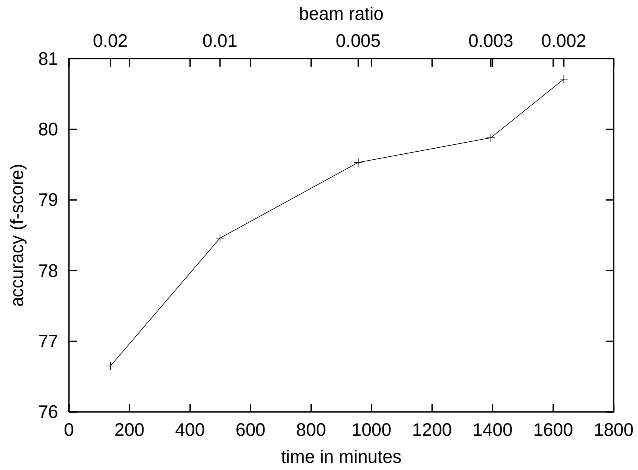

Figure 2.8: F_1-score vs. time to parse 552 sentences using various beam ratios

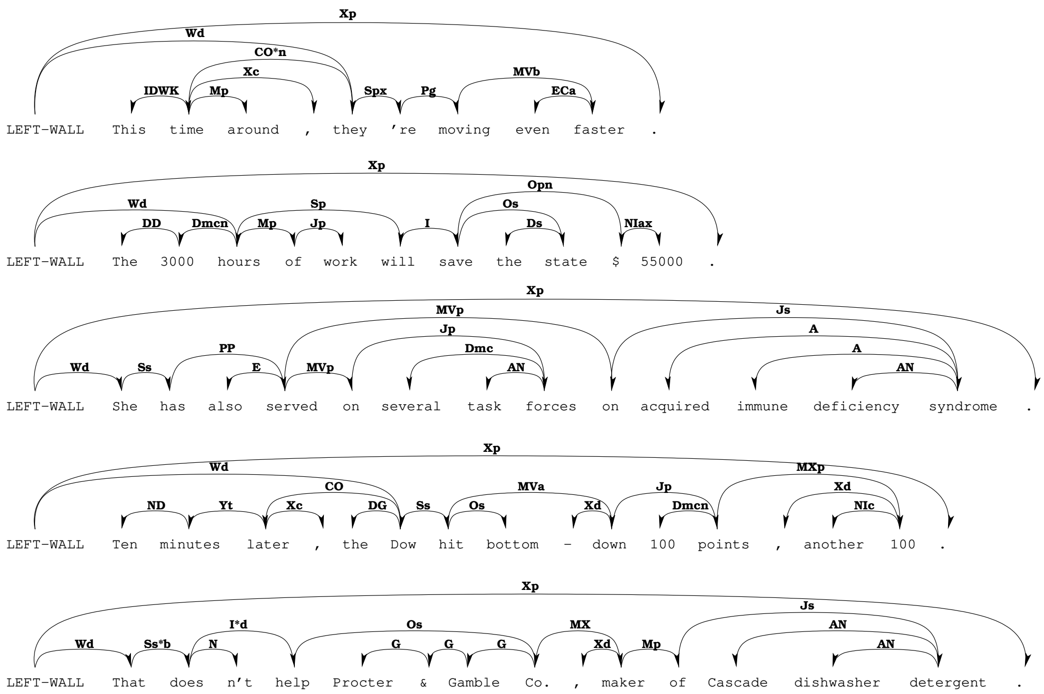

Figure 2.9: Selected sentences parsed by the Lynx parser

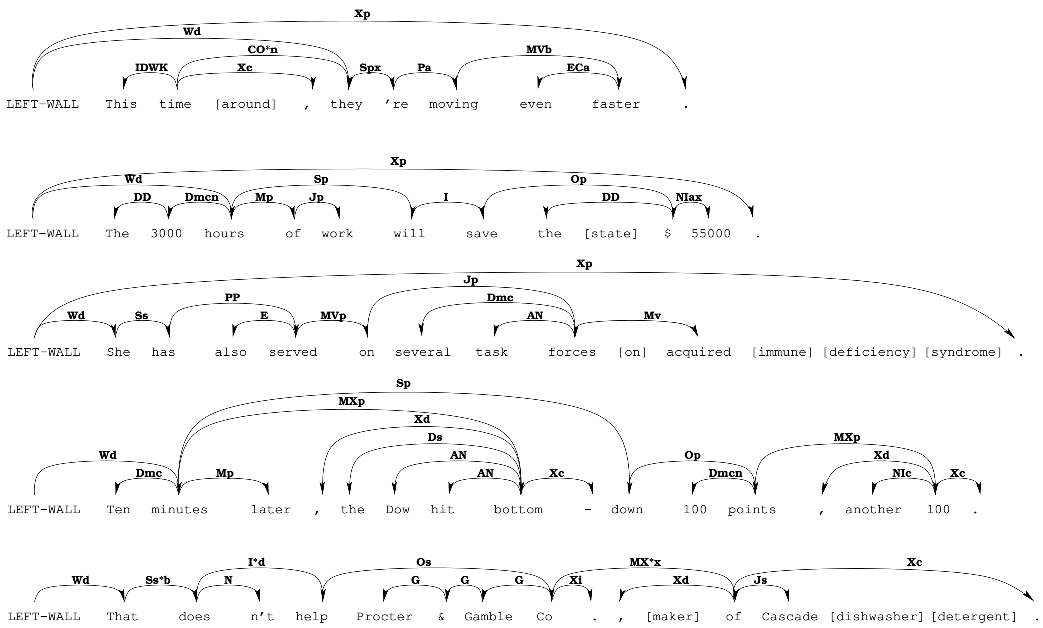

Figure 2.10: Selected sentences parsed by the Link parser

null link, while the Lynx parser analyzes *around* as a prepositional phrase modifying *time*.

- *3000 hours of work will save the state $55000.*
  The Link parser doesn't know that *save* can be ditransitive (take two objects) but the Lynx parser allows it.

- *She has also served on several task forces on acquired immune deficiency syndrome.*
  The Link parser chokes on the phrase *acquired immune deficiency syndrome* because it has no determiner, but the Lynx parser easily adapts.

- *Ten minutes later, the Dow hit bottom — down 100 points, another 100.*
  The Link parser chokes on the phrase *hit bottom* because it expects *bottom* to be preceded by a determiner, and tries a totally different analysis, where *minutes* is the subject and *down* is the verb. The Lynx parser doesn't have this problem.

- *That doesn't help Procter & Gamble Co., maker of Cascade dishwasher detergent.*

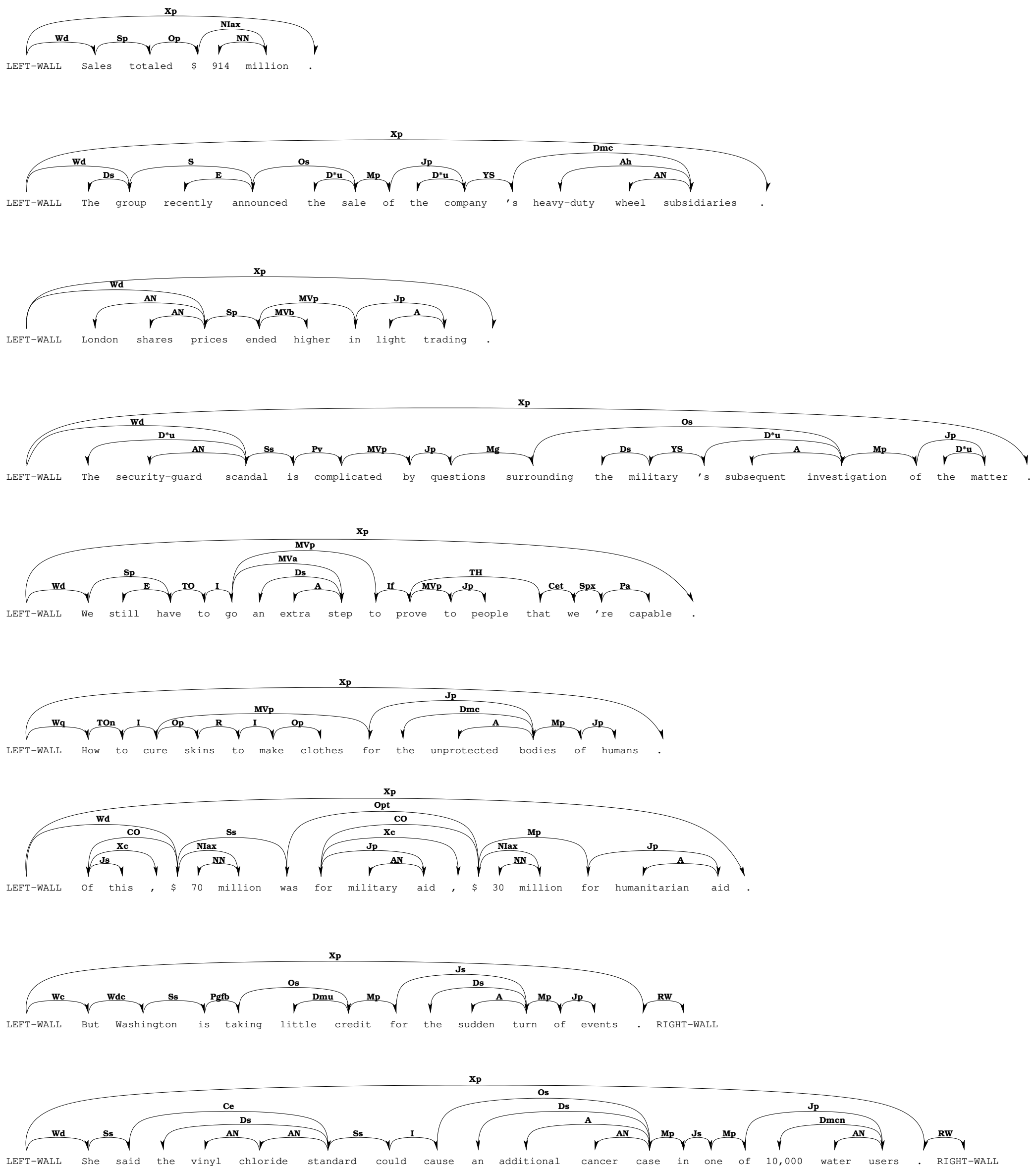

Figure 2.11: Randomly selected sentences parsed by the Lynx parser

Again, the Link parser expects *maker* to have a determiner, and is unable to come up with a reasonable parse. The Lynx parser, again, deals gracefully with missing items.

For fairness, balancing the selected sentences above, nine sentences were also selected at random from the *WSJ-1987* corpus parsed by Lynx; these are shown in Figure 2.11. Six of the nine are fully correct; we will look in detail at the other three.

- *The group recently announced the sale of the company's heavy-duty wheel subsidiaries.*
  The analysis given by Lynx incorrectly connects *of* to *company* with a **Jp** link, meaning that *company* is the object of the preposition *of*. Instead, this link should have connected *of* with *subsidiaries*. The Link parser analyzes this sentence correctly. The Lynx parser can also analyze this sentence correctly when a wider beam is used, which indicates that the error is due to a search deficiency rather than a problem with the model.

- *How to cure skins to make clothes for the unprotected bodies of humans.*
  While this is not strictly a well-formed English sentence, the Lynx parser does reasonably well at analyzing the entire fragment; the Link parser, on the other hand, skips the first two words and interprets the verb as an imperative. The parse shown here, however, has two incorrect attachments. First, the connection between *skins* and *to* is wrong; it should be replaced by a link labeled **MVi** from *cure* to *to*. Second, the attachment of the prepositional phrase starting with *for* should be to *clothes* rather than *cure*. Both of these are parsed correctly by the Link parser, and both are also parsed correctly if the Lynx parser uses a wider beam.

- *Of this, $70 million was for military aid, $30 million for humanitarian aid.*
  The structure shown means that *$70 million* was *$30 million for humanitarian aid*, for the purpose of *military aid*. Instead, it should mean that *$70 million was for military aid* and that *$30 million* was *for humanitarian aid*. Since link grammar does not handle deletion (of the verb in the second phrase) there is no fully correct linkage possible, so the Link parser cannot solve this one either.

Finally, the 25 worst sentences from the *WSJ-1987* corpus on which the Link parser did fine but the Lynx parser scored lowest were found and categorized. Of

1. But Jerry isn't quite the stereotype he appears to be.
2. In Montreal, Dominion declined to comment except to say, "We're reviewing the situation."
3. His conception did not include petit dejeuner tete-a-tete.
4. Don't say that something is within a statute's reach; say it's within its 'ambit.'
5. The mafrag is designed for afternoon gatherings called "qat" chews.
6. Of his East-bloc brushes, Geoff says, "We don't mingle.
7. We genuinely feel our proposed plan was a good-faith effort to resolve the issue.
8. Mrs. Walton worries that Lafeyette has become unusually withdrawn: "He says talking isn't going to help him.
9. "But most people who are confronted with this pay."
10. The economy is growing moderately despite the stock market crash, a Journal survey of economists says.
11. Viewed through these eyes, Fresno wins hands down.
12. A spokeswoman for British & Commonwealth, a U.K. financial-services concern, declined to discuss the purported talks.
13. Dealers are reluctant to talk for the record because the usually secretive IBM has been extraordinarily so on this go-round.
14. She says the commercial is the only one she doesn't fast-forward through her video recorder.
15. "Oversights were made in these instances which I genuinely regret," Rep. Coelho said in a brief statement.

Table 2.5: Some sentences on which Lynx had difficulty

these worst offenders, 60% failed completely, due either to search errors, model errors, bogus or strange sentences, or software bugs. However, on a second attempt, with a more generous beam ratio, better-trained model, and a more careful eye for software bugs, 32% were quite successful, including 40% of those that had failed completely in the original experiment. Some of the worst sentences are shown in Table 2.5. Many of these problem sentences contain quotation marks, which do not cause problems directly, but often indicate a sentences that will be difficult to parse. The improvement due to minor tweaking of the model and search parameters shows that additional training and computational resources can be of significant help in dealing with difficult sentences.

2.6 Discussion

The Lynx parser analyzes sentences for syntactic structure by finding the most probable structure consistent with the given text, according to a statistical model of structured sentences. The model is trained from examples. Because it is expensive to annotate examples by hand, we used an existing parser to analyze sentences collected from news articles, and used its output as training data. Although this automatically-generated training set has significantly more errors than one would expect from a hand-labeled training set, the Lynx parser is able to score as well as the Link parser at constituent bracketing. In the experiment above (see Table 2.1), the Lynx parser achieved an F_1-score better than or equal to the score achieved by the Link parser on 70% of the sentences, and the mean scores are very close. This shows that the Lynx parser compares favorably with an established parser, which is good. However, this result is made more meaningful by the observation that the Link parser is the source of all the training data for the Lynx parser. That is, everything the Lynx parser knows about English grammar, it learned from the Link parser, with the exception of the very general principles implicit in the model's framework.

Admittedly, the Lynx parser does not score better than its source of training data, only about the same. However, on 30% of the sentences evaluated, the Lynx parser scores better than the Link parser, showing that it actually does have a significant improvement in the ability to parse some sentences, offset by inferior performance on some other sentences. Another way of looking at the difference between the two parsers is to predict the score that would result if both cooperated. If a highly accurate selector could be trained to choose which parser has a better analysis of each sentence, a gain of almost 5 percentage points is possible.

Why is the Lynx parser able to score better than its source of training data on so many sentences? There are several reasons, but the most obvious is that statistical parsing offers a significant advantage in robustness over rule-based parsing. In the rule-based Link parser, any constructions not anticipated by the designers of the grammar can only be covered through **null** links. There is a sharp dividing line between what is considered grammatical and what is not, which can cause sentences with a missing word to be analyzed strangely or not at all. A statistical parser such as Lynx, on the other hand, does not have a division between grammatical and ungrammatical sentences, but only a probability distribution. Thus when confronted with a sentence that would be considered ungrammatical by a more rigid grammar, a stochastic syntax model simply assigns a lower score, nevertheless selecting the structure most probable under the circumstances. This generalization yields a great increase in robustness, allowing performance to degrade gracefully on more difficult sentences.

In the task of modeling the grammar of a language, as in many machine-learning tasks, there's a tension between under-generalizing and over-generalizing. If a rule-based grammar model is too general, it will over-generate, coming up with ungrammatical parses, but if it's too narrow, it will fail to parse some grammatical sentences. A statistical parser, on the other hand, does not need to sharply distinguish valid from invalid parses, but only to assign a probability to each in a way that gives the most correct analysis the highest probability. If the model is too flat, or does not take crucial dependencies into account, it will be unable to distinguish good from bad structures. A model that is too sharp, however, will assign zero probability to unfamiliar sentences, failing in much the same way as a rule-based parser might. In fact, this problem can be even worse on an unsmoothed stochastic model than a rule-based model because the lexical dependencies in the model could be trained on too sparse a training set to obtain reasonable probabilities. Striking the right balance between over-generalizing and under-generalizing is important in the design of a statistical syntax model.

The robustness gained through statistical modeling clearly depends on the framework in which that model is learned, and on how the model is smoothed. It would be possible to design a statistical model that captures exactly the grammar used by the Link parser and does not generalize beyond it. Such a model could still have an advantage over the original Link parser because it would use statistics rather than heuristics to select the best from among all parses consistent with the grammar for a given sentence. The easiest way to implement such a system would be to simply add a statistical component on top of the existing Link parser, as a post-processing step. Such a system could be useful, but it would not be any more robust than the original Link parser. In contrast, the model used in the Lynx parser is smoothed to assign part of its probability mass to structures that were never seen in the training data. The probability of each link and each word is considered separately (rather than a disjunct of connectors being treated as a unit), so while a missing, misplaced, or unknown word may make a structure as a whole less probable, the parts that are recognized will continue to assign it their share of probability, making partial analyses possible. In order for this to work well, the framework of the Lynx model had to be designed well, with good decisions made as to which probabilities should be conditioned on which, and which should be considered independent. As shown in Section 2.3 above, the main assumption made is that syntax has a hierarchical structure and that dependencies are local within that structure.

Because of the independence assumptions and smoothing of the Lynx model, it can easily handle inserted or deleted words and phrases. In the examples discussed above (Section 2.5.5), the Lynx parser's analysis of the sentence "*3000 hours of work will save the state $55000*" (Figure 2.9) shows the verb *save* taking two objects, which was not allowed by the Link parser. The three following sentences each demonstrate the Lynx parser's ability to correctly parse a noun phrase that is missing a determiner, considered ungrammatical by the Link parser, which is thus not able to analyze it correctly. A noun phrase that always has a determiner in the training data will score a lower probability (according to the Lynx model) in the absence of one; however, the probability of the noun in the presence of the determiner is smoothed with the probability of the noun disregarding context, thus allowing the phrase to be correctly analyzed. In contrast, the Link parser's **null** link mechanism allows it to gloss over a inserted word, but not to connect an inserted phrase in a meaningful way. It has no mechanism to deal with missing words. Thus the design of the Lynx model framework helps it to parse robustly.

Although all the data used to train the Lynx model was generated by the Link parser, the sentences which the Link parser parsed for this purpose were real sentences, constituting new input to the system. One of the reasons the Lynx parser is able to do so well is that it gains the benefit of the grammatical and distributional knowledge implicit in the sentences used for training, even though the analysis of these sentences by the Link parser is fallible. Most of the errors made by the Link parser when generating training data are compensated for by the large amount of data. While this training is a fully automated process, given a human-generated but unannotated text corpus (a plentiful resource), it is a *semi-supervised* learning process because the explicit grammar knowledge exercised by the Link parser acts as a kind of supervisor. The combination of this explicit grammar knowledge with real data contains more information than either alone.

While this semi-supervised learning technique brings reasonably good results, there are still many sentences which the Lynx parser does not analyze as accurately as the Link parser can. However, many of these errors are only search errors, which can be avoided at the expense of a little more computation time. As described above, a significant portion of sentences on which the Lynx parser failed in the *WSJ-1987* experiment can be handled correctly with just a little coaxing. Therefore, while the Lynx parser's results are currently comparable with those of the Link parser, with a mix between improvements and deterioration, much of the deterioration could be addressed with a moderate additional investment, resulting in a parser that significantly out-scores the Link parser overall. However, it is also likely that a similar investment in the Link parser could yield comparable returns. This agenda was not pursued because the goal of the Lynx parser is not primarily to provide a new parser that does the same task as the Link parser does, only better. Rather, the purpose of the Lynx parser is to provide a system for quickly constructing a parser from data. Two applications of this capability are explored in Chapters 4 and 5 of this thesis: inducing a monolingual parser through bilingual parsing, and reranking translation hypotheses with a bilingual parsing model.

Chapter 3

Bilingual Parsing

3.1 Motivation

In order to create a parser for a language in which no parser is available, without requiring extensive labor by a language expert to either design a grammar or annotate a corpus, it would be very useful to be able to automatically infer the syntactic structure of foreign sentences. When given a bilingual corpus where a parser is already available for the opposite language (such as English), this could be done in principle by word-aligning the corpus and "copying" the structure across the alignments. Of course, figuring out the details of how this "copying" would take place could be quite tricky, or even impossible, since alignments are not always one-to-one.

In order to build a statistical translation system that accurately models the hierarchical structure of natural language, it would be very useful to be able to train a model of a structured foreign sentence given a structured English sentence. This would require designing a model that captures the relevant features of the relationship between structures in two languages. Even when given a bilingual corpus in which structures are available for both languages, training such a transformational model might not be straightforward, because the relationship between the structures would have to be defined in a way that could be modeled. Such corpora are not usually available anyway, especially if the two languages must be annotated using the same formalism.

Putting these two problems together, we need a way to infer the structure of a given foreign sentence (with no previous knowledge of the foreign language), given an English translation and its structure; and we need a way to train a transformational model that generates the foreign structure given the English structure. Solving these problems would allow us to build both a parser for the foreign language and a statistical translation system between that language and English that takes into account the syntactic structure of each sentence.

The LinkSet bilingual parser solves these two problems together by simultaneously inferring the structure of foreign sentences, based on their English counterparts, and training a model of how the English sentences' structure must be transformed to produce the inferred foreign structures. Inferring the structure of a sentence is usually a parsing problem, but in this case the grammar of the foreign language is unknown. However, we assume that the structure of each foreign sentence is related to the known structure of its English counterpart, according to the transformations allowed by our model. We call this approach *bilingual parsing* because we use the structure of an English sentence to generate a specialized grammar in order to parse its foreign counterpart. Once our model has been trained, it should be reasonably straightforward to find the best structure (according to the model) for any foreign sentence in our sentence-aligned bilingual corpus. But how can we train the model in the first place? The answer is essentially an EM approach: starting with a flat model for each structural transformation, we use a word-translation model trained using IBM Model 2 (or some equivalent) to assign a probability to each possible foreign structure. Each valid foreign structure is equivalent to a particular way of transforming the given English structure, which in turn is equivalent to a particular word-to-word alignment between the English and foreign sentences. We use the probability of this alignment to update the parameters of the transformation model for the particular transformations necessary to generate the corresponding foreign structure. Performing this update for every possible alignment of every sentence pair in the corpus constitutes one round of training. As a byproduct of each round of training, the most probable structure of each foreign sentence can trivially be found, since each possible structure is assigned a score. These structures are used in Chapter 4 to automatically induce a foreign-language parser. In addition to improving the quality of foreign structures inferred, the learned transformational model can then be used in a translation system, which is the subject of Chapter 5.

3.2 Related Work

3.2.1 Basic Statistical Alignment

As mentioned in the first chapter, in the early 1990's an IBM research group developed a foundational series of generative translation models [4, 5]. These models all worked in terms of an alignment between the words of the English and foreign versions of each sentence, such that each foreign word was either associated with one English word or with a special null word. Some models allowed more than one French word to align with a single English word; others required one-to-one alignment. While they do not account for syntactic structure, these simple models work reasonably well at aligning the words of parallel sentences, and their parameters can be used as a stochastic translation dictionary. The word-translation model for the LinkSet bilingual parser was trained using IBM Model 2, which includes distortion probabilities to model the position of each word, along with the word-translation probabilities; however, the distortion model was not explicitly used by LinkSet. Many others have adopted or adapted the five IBM Models, and many different alignment models have been proposed, too many to enumerate here. We will focus instead on those models that attempt to account for syntactic structure.

3.2.2 Structural Alignment

Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora

Inversion transduction grammar (ITG), introduced in 1997 by Dekai Wu [27], is a formalism capable of representing bilingual context-free grammars, adding to the power of ordinary transduction grammars, which rigidly require the two languages to have the same structure, by allowing some subtrees to have opposite orderings in the two languages. Stochastic inversion transduction grammars associate a probability with each rewrite rule. Wu uses the term *bilingual parsing* to refer to his unsupervised method for learning an ITG from a bilingual corpus, which is similar to but not the same as our use of the term to refer to our method for learning a bilingual structural mapping from a bilingual corpus where one side has been annotated by a parser. From a Chinese-English bilingual corpus and a stochastic translation dictionary, Wu trained an ITG with only one non-terminal symbol and evaluated its bracketing accuracy at 80% compared to human bracketings. As future work, Wu suggested using known English parse trees to constrain the bilingual parsing, in order to transfer grammatical knowledge from English to Chinese, an idea that is similar to what the LinkSet parser does. One key similarity of ITG to LinkSet is that both consider all possible alignments between each sentence pair, rather than selecting a single alignment first from which to constrain other learning. The two techniques have different points of view in that ITG simultaneously describes the structure of two sentences, while the LinkSet model describes the relationship between two distinct parse trees. While ITG can model any alignments the LinkSet model can, it is much more restrictive in the shape of the two syntax trees, and particularly in the relationship between them. Because of this, a bilingual grammar learned in an ITG framework may not be able to conform to a standard grammar of either language, but settles for a task-driven decomposition, while the LinkSet model begins with a standard grammar in one language and maps to a similar grammar in the other.

Learning Dependency Translation Models as Collections of Finite State Head Transducers

Similar to ITG are weighted head transducers, introduced in 2000 by Alshawi, Bangalore, and Douglas [1], which, like the LinkSet bilingual parsing model, have a less restricted grammar because they allow arbitrary branching factors and arbitrary reordering within parse trees. Their method is to search first for hierarchical alignments based on correlation statistics, and then construct head transducers consistent with these alignments. The transducers can then be used for translation, and successful translation was demonstrated from English to Spanish in the air travel domain, and to Japanese in the telephone-operator domain. These experiments were in task-oriented domains with small vocabularies, in contrast with the harder general domain problem at which the LinkSet system is aimed. This method first selects an alignment and then builds a transducer, rather than integrating the full range of potential alignments into the model as do ITG and LinkSet.

3.2.3 Alignment Using Known Structure

Grammar Inference and Statistical Machine Translation

In his 1998 thesis, Ye-Yi Wang [26] developed a bilingual word-clustering technique based on mutual information, which improves the quality of monolingual word classes by using bilingual data. These classes were then used to find bilingual phrases, common word-class sequences that often have a corresponding phrase as a translation. These phrases were then used in a statistical machine translation system to model sentence structure more accurately than the distortion models of the original statistical machine translation work. Wang's Structure-based Alignment Model works by first parsing an English sentence into phrases, and then generating a translation based on those phrases. Thus it projects simple structural information from English to the foreign language when generating a translation.

A Syntax-based Statistical Translation Model

The previous work most closely related to the LinkSet model is the syntax-based statistical translation model designed by Yamada and Knight [28], which accepts an English parse tree as an input and generates a Japanese parse tree by applying three channel operators: reordering of the children of any node, insertion of a word adjacent to any node, and translation of each leaf node. While the expressiveness of this mapping between parse trees is very similar to the LinkSet bilingual parsing model, the LinkSet model gains expressiveness with the additional operations of subtree-deletion, head-deletion, and null-linking. Yamada and Knight trained their model efficiently using the Inside–Outside algorithm, and used it to produce alignments judged by humans to be more accurate than alignments produced using IBM Model 5. Because structure of this model is so similar to the LinkSet model, we use a variant of their training algorithm to efficiently train the LinkSet model.

More recently (2003), Knight and Yamada teamed up with Charniak to build a translation system based on their model [6]. Although the model was trained using English sentences parsed with Collins' parser, they adapted Charniak's parser to act as a decoder, formulating translation as parsing of a foreign sentence using special grammar rules. To generate these rules, English CFG rules were supplemented with all variants that can be made by reordering the right-hand side, plus an insertion rule, and, finally, a translation rule for each equivalent English/foreign word pair. They showed an increase in high-quality translations (especially in syntactic well-formedness) compared to two other systems.

The insight that translation can be viewed as a parsing problem with a special grammar underlies the LinkSet system as well: LinkSet specializes parsing rules for each sentence during training in order to efficiently search for the most probable structured alignment between an English and a foreign training sentence.

Loosely Tree-Based Alignment for Machine Translation

When structural information is available for both languages in an alignment task, tree-to-tree alignment is possible, as recently demonstrated by Gildea [16]. Expanding on the work of Yamada and Knight, Gildea's tree-to-tree alignment system does not achieve a significant improvement in alignment quality over tree-to-string alignment, but does achieve greatly increased efficiency due to the much more constrained nature of tree-to-tree alignment. However, the additional constraints imposed by a pair of trees can often be too tight to allow a correct alignment between languages with different structure, so Gildea relaxes the model by introducing a subtree-cloning operation, which by allowing a subtree to be cloned and inserted at any point (along the linear word order) turns the model into a sort of hybrid between tree-based and unstructured, IBM-style alignment models.

A Best-first Alignment Algorithm for Automatic Extraction of Transfer Mappings from Bilingual Corpora

Like our LinkSet method, the methods of Wang, and of Yamada, Knight, and Charniak assume that the structure of English is known, while that of the foreign language is not. When structure is known for both languages, however, a higher-quality mapping should be possible. Menezes and Richardson [20] present an alignment algorithm that uses 18 rules and a translation dictionary to align the logical form of two structured sentences, one being a translation of the other. Using this alignment, they generate more general transfer mappings, which can then be used in a translation system. Evaluation of the resulting system, trained for the same domain as the test set, showed it to be comparable to an unspecialized commercial translation engine.

3.3 The LinkSet Model

The LinkSet bilingual parser simultaneously trains and uses a structural translation model to find the most likely structure of a foreign sentence, when the sentence is given along with its English structure and text. The most likely structure $\hat{\mathcal{F}}_{\mathbf{f}}$ for a foreign sentence $\mathbf{f}$ can be found by maximizing the probability of the structure (given its English counterpart $\mathcal{E}$) over all structures consistent with the foreign sentence:

$$\hat{\mathcal{F}}_{\mathbf{f}} = \operatorname*{argmax}_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E})$$

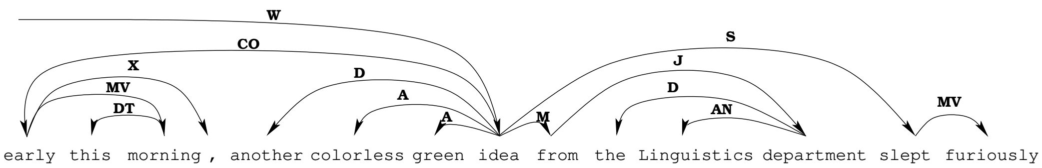

Figure 3.1: An example sentence structure

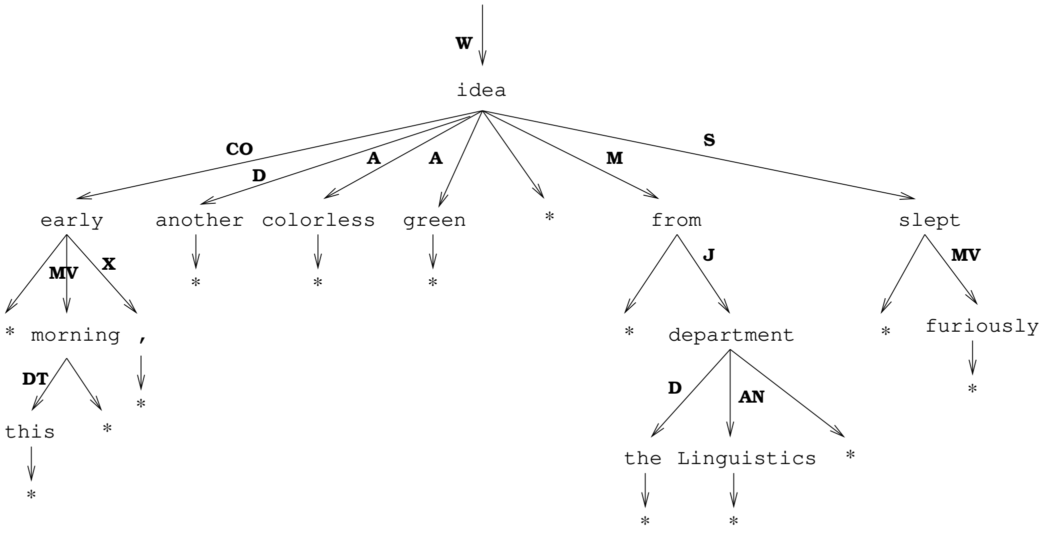

Figure 3.2: The same structure as a syntax tree

This structural translation model is a transformational model of a foreign sentence and its structure, given an English sentence and its structure. Thus the foreign sentence structure and text are generated with the probability of generating each node in the syntax tree conditioned on the corresponding English node.

The syntactic tree structures introduced in the previous chapter are based on a link grammar, from which we've picked a special node to be the root, and imposed direction on all the dependencies based on that choice. Our view, in which each node has children to its left and right, is equivalent, as a comparison of Figures 3.1 and 3.2 will show. In Figure 3.2, the asterisks represent the position of the current node relative to its children, dividing the left from the right. In Figure 3.3, we again see the same tree, but this time with each node holding ordered lists of children (divided between left and right), with the local node label preceding these lists. Here the positions of the nodes and the lines connecting the nodes give no information not already shown inside the nodes.

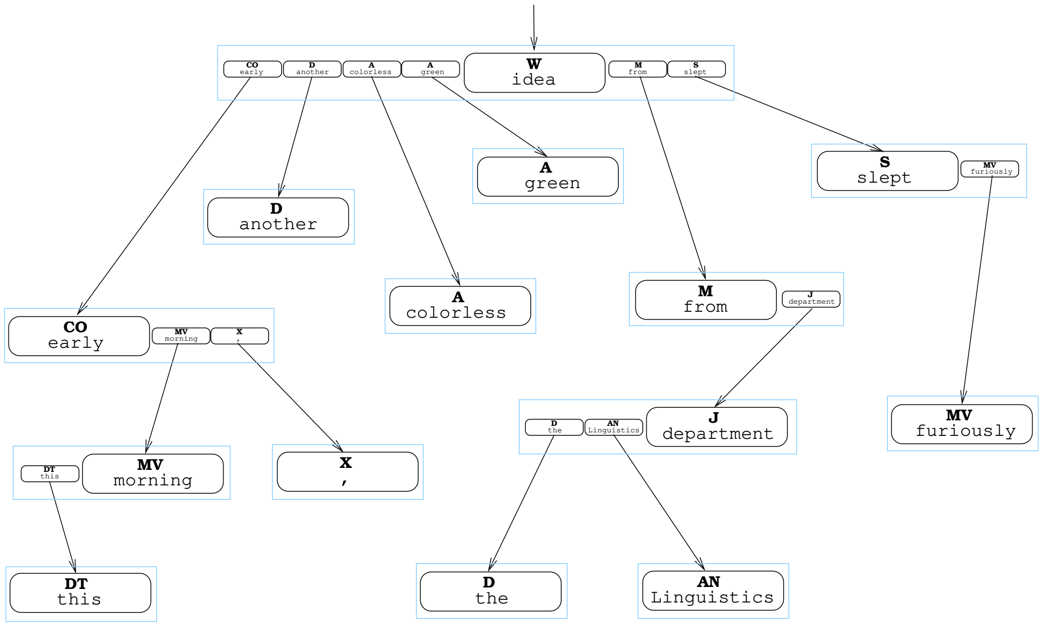

Figure 3.3: The same structure as a syntax tree, showing children as lists

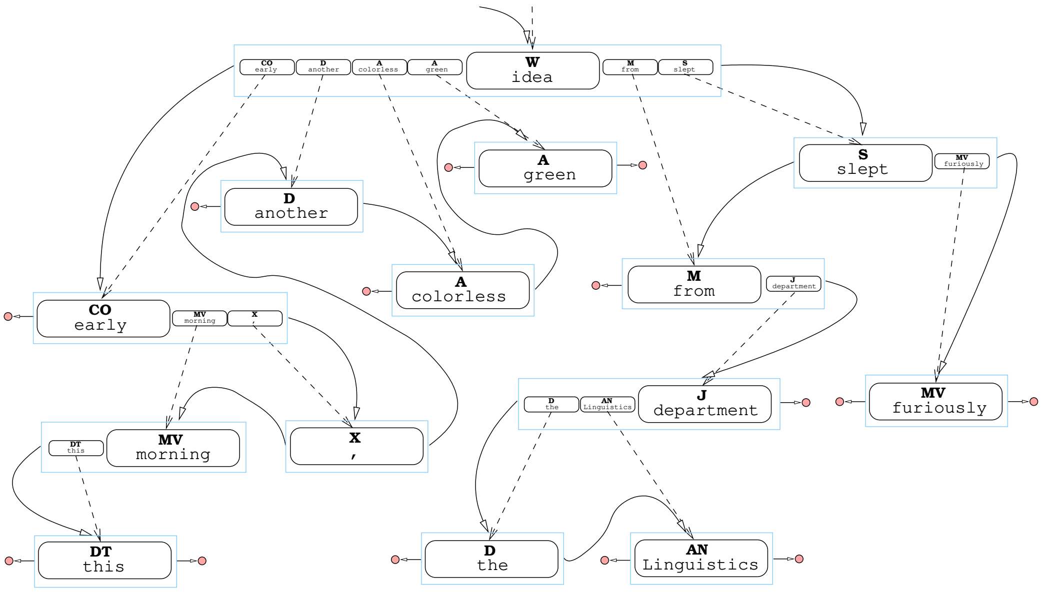

Figure 3.4: The same structure with binary links added

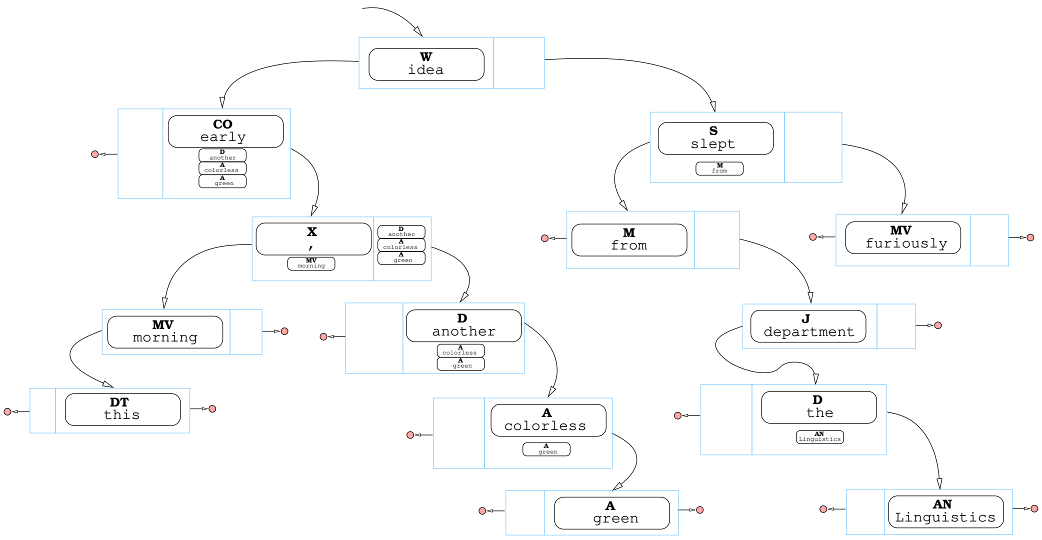

Figure 3.5: The same structure as a binary syntax tree

procedure CONVERTTREE(L, R)
if $L \neq \emptyset$
then $Z \leftarrow L.$POP$()$
else $Z \leftarrow R.$POP$()$
$\mathcal{L} \leftarrow$ CONVERTTREE$(L, Z.L)$
$\mathcal{R} \leftarrow$ CONVERTTREE$(Z.R, R)$
return $(\mathcal{L}, \mathcal{R})$

Figure 3.6: CONVERTTREE algorithm

The LinkSet search algorithm uses a different tree structure to represent the syntax of each sentence, one that more closely fits the operation of the algorithm, which is related to the algorithm used in the Link parser. There is a one-to-one mapping between this new tree representation and the original one, but unlike the original representation, it is always a binary tree. Although the information content is essentially the same, the relationships between nodes are managed differently. The parsing algorithms used by the Link, Lynx, and LinkSet parsers divide sentences recursively into two parts, where each part is characterized by the range of words it covers and by the labels of the links connecting it to the rest of the sentence. A range of words with its associated incoming links corresponds to a node in the binary tree used to represent the sentence.

To show the relationship between a tree of the more intuitive type and the binary trees used for parsing, Figure 3.4 is an intermediate stage, showing the links of both kinds of trees. The dotted lines are the old connections carried over from Figure 3.3, and the curving solid links are the connections of the new binary structure. The unique left and right children of a node are the outermost left and right children (respectively) of the corresponding N-ary node, if any. Nodes that originally had no children on a side will adopt their nearest sibling or ancestor's sibling in that direction. Thus the order of nodes, according to an inorder traversal, is the same between the two representations, preserving the word order of the sentence.

Figure 3.5 shows the fully-transformed binary tree. The dotted links being removed, it has the binary links added in the previous figure, and the contents of the nodes have been revised. Each major node (shown as a double rectangle) consists of two lists of minor nodes, which are words with their incoming connectors. The two lists correspond to the incoming connectors available to the range of words covered by the major node. The minor node that heads a major node is the topmost connector on its left side, if any, or otherwise the topmost connector on the right side. Once the head is removed, the left and right children are the farthest children (in the original tree) of the head, if they exist, or the top nodes in the respective lists of incoming connectors. Any remaining incoming connectors are inherited by the major node's children on their respective sides. Since the lists of minor nodes in each major node represent inherited incoming connectors but not true children of the current head, the links between major nodes contain important information, rather than being redundant with respect to the node contents, as in Figure 3.3.

A more formal description follows. Each major node is a tuple $\langle \mathcal{L}, \mathcal{R} \rangle$ of vectors of minor nodes for the left and right, respectively, where each minor node in those vectors is a tuple $\langle c, t, w, L, R\rangle$, where $\langle c, t, w\rangle$ has the same interpretation as in Section 2.3 above (connector, tag, and word) and $\langle L, R\rangle$ gives the minor node's immediate left and right children, each of which is a major node. Each major node $\langle \mathcal{L}, \mathcal{R}\rangle$ is associated with a single minor node, which is the topmost minor node in $\mathcal{L}$, unless $\mathcal{L} = \emptyset$, in which case it is the topmost minor node in $\mathcal{R}$. For convenience, we define the simple function $\textsc{Top}(\nu.\mathcal{L}, \nu.\mathcal{R})$ to give the minor node associated with the major node ν. The CONVERTTREE algorithm, shown in Figure 3.6, converts from the N-ary tree representation (see Figure 3.3) to the binary representation (see Figure 3.5), when started with the parameters $\langle \boxed{\text{W:idea}}, \emptyset\rangle$, where $\boxed{\text{W:idea}}$ represents the top minor node. The binary representation will be assumed for the rest of this chapter.

The probability $\Pr(\mathcal{F}|\mathcal{E})$ of generating a structured foreign sentence given a structured English sentence can be expressed as the probability of passing the English sentence tree through certain transformations and finding the foreign sentence tree as a result. Each major node of the tree passes through the following transformations recursively, starting with the root.

1. A minor node is selected (from among those listed at the current major node) to head the current major node.

2. The selected node may be replaced with one of its descendants. (This is called *head-deletion*.)

3. The word at the selected node (or its replacement) is translated from English to the foreign language.

4. Any remaining minor node may be deleted.

5. The remaining minor nodes are repartitioned into left and right sets.

6. On either side, a new minor node may be inserted. Inserted nodes correspond to *null* links.

The probability of a foreign sentence tree given an English sentence tree is the product over all nodes in the English tree of the probabilities of applying these transformations in a way that results in the foreign sentence tree.

1. The first step in the transformation of the English major node $\langle \mathcal{L}_{\mathcal{E}}, \mathcal{R}_{\mathcal{E}}\rangle$ into the foreign major node $\langle \mathcal{L}_{\mathcal{F}}, \mathcal{R}_{\mathcal{F}}\rangle$ is the selection of a minor node ν_s from among its members to be the head:

$$\Pr(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \mathcal{L}_\mathcal{E}, \mathcal{R}_\mathcal{E}) = \sum_{\nu_s} P_C(\nu_s | \mathcal{L}_\mathcal{E}, \mathcal{R}_\mathcal{E}) \; P_2(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \nu_s, \mathcal{L}_\mathcal{E}, \mathcal{R}_\mathcal{E})$$

2. Once a head is selected, the second step allows ν_s to be replaced by one of its descendants:

$$P_2(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \nu_s, \mathcal{L}_\mathcal{E}, \mathcal{R}_\mathcal{E}) = \sum_{\nu_h} P_H(\nu_h | \nu_s) \; P_3(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \nu_h, \mathcal{L}_\mathcal{E}, \mathcal{R}_\mathcal{E})$$

3. Then the selected minor node is translated, which includes translation of its word and copying of its POS tag:

$$P_3(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \nu_h, \mathcal{L}_\mathcal{E}, \mathcal{R}_\mathcal{E}) = P_T(\textsc{Top}(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F}) | \nu_h) \; P_4(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \nu_h)$$

4. After the translation step, remaining minor nodes (candidates to be children of this node) may be deleted:

$$\begin{aligned} P_4(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \nu_h) &= \sum_{\mathcal{L}', \mathcal{R}'} P_D(\mathcal{L}' | \nu_h.L.\mathcal{R}) \; P_D(\mathcal{R}' | \nu_h.R.\mathcal{L}) \times \\ &\quad P_5(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \mathcal{L}', \mathcal{R}', \nu_h) \end{aligned}$$

5. After deletion, the left and right sets are repartitioned:

$$\begin{aligned} P_5(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \mathcal{L}', \mathcal{R}', \nu_h) &= \sum_{\mathcal{L}''} P_F(\mathcal{L}'' | \mathcal{L}', \mathcal{R}') \times \\ &\quad P_6(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \mathcal{L}'', ((\mathcal{L}' \cup \mathcal{R}') - \mathcal{L}''), \nu_h) \end{aligned}$$

6. Finally, *null* links may be inserted, and the left and right major-node children are transformed recursively.

$$\begin{aligned} P_6(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F} | \mathcal{L}'', \mathcal{R}'', \nu_h) &= \sum_{\mathcal{L}''', \mathcal{R}'''} P_I(\mathcal{L}''' | \mathcal{L}'') \; P_I(\mathcal{R}''' | \mathcal{R}'') \times \\ &\quad \Pr(\mathcal{L}_\mathcal{F} - \textsc{Top}(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F}), \mathcal{L}''' | \nu_h.L) \times \\ &\quad \Pr(\mathcal{R}''', \mathcal{R}_\mathcal{F} - \textsc{Top}(\mathcal{L}_\mathcal{F}, \mathcal{R}_\mathcal{F}) | \nu_h.R) \end{aligned}$$

Each step in the sequence above refers to a sub-models relevant to its respective transformation step. These models are P_C, P_H, P_T, P_D, P_F, and P_I; they will now be described in more detail:

1. $P_C(\nu|\mathcal{L},\mathcal{R})$ models the probability of selecting a particular minor node to be the top of the transformed major node. In practice, an approximation $P_C(\nu.c)$ depending only on the label of the selected connector is used.

2. $P_H(\nu'|\nu)$ models the head-deletion operation, which may replace any node with one of its descendants. The probability is zero unless $(\nu' = \nu)$ or $(\nu' \in \textsc{Descendants}(\nu))$. If $\nu = \nu'$, then the probability is just the probability of not head-deleting that node:

$$P_H(\nu|\nu) \quad = \quad 1 - \Pr(\textsc{HeadDel}|\nu.c)$$

   Otherwise, we recursively apply the probability of head-deleting this node until a path to the original is found:

$$P_H(\nu'|\nu) \quad = \quad \Pr(\textsc{HeadDel}|\nu.c) \sum_{\nu'' \in \textsc{Children}(\nu)} P_H(\nu'|\nu'')$$

3. $P_T(\nu_{\mathcal{F}}|\nu_{\mathcal{E}})$ models the translation operation. It gives the probability of translating the English word e (with POS tag t) into the foreign word f. The tag is copied, so the probability must be zero if $\nu_{\mathcal{F}}.t \neq \nu_{\mathcal{E}}.t$.

$$P_T(\nu_{\mathcal{F}}|\nu_{\mathcal{E}}) \quad = \quad P_T(\nu_{\mathcal{F}}.w|\nu_{\mathcal{E}}.w, \nu_{\mathcal{E}}.t)$$

4. $P_D(\mathcal{S}'|\mathcal{S})$ models the deletion operation, giving the probability of $\mathcal{S}'$ being left after a (possibly empty) subset of $\mathcal{S}$ is deleted. The probability is zero unless $\mathcal{S}' \subseteq \mathcal{S}$. The probability of each deletion is considered to be independent.

$$P_D(\mathcal{S}'|\mathcal{S}) \quad = \quad \prod_{\nu \in S} [1 - P(\textsc{Del}|\nu)]^{\delta(\nu \in S')} \, P(\textsc{Del}|\nu)^{\delta(\nu \notin S')}$$

5. $P_F(\mathcal{L}'|\mathcal{L},\mathcal{R})$ models the reordering operation. It gives the probability of the nodes in $\mathcal{L}'$ being on the left side after reordering, when the nodes in $\mathcal{L}$ and $\mathcal{R}$ originated on the left and right sides, respectively. It models which nodes "flip" between left and right, which is why it we label it P_F. The

probability is zero unless $\mathcal{L}' \subseteq (\mathcal{L} \cup \mathcal{R})$, since new nodes cannot appear through flipping.

$$\begin{aligned} P_F(\mathcal{L}'|\mathcal{L},\mathcal{R}) &= \left(\prod_{\nu\in\mathcal{L}} \Pr(\textsc{Flip}|\nu.c)^{\delta(\nu\notin\mathcal{L}')}\,[1-\Pr(\textsc{Flip}|\nu.c)]^{\delta(\nu\in\mathcal{L}')}\right) \times \\ &\quad \left(\prod_{\nu\in\mathcal{R}} [1-\Pr(\textsc{Flip}|\nu.c)]^{\delta(\nu\notin\mathcal{L}')}\,\Pr(\textsc{Flip}|\nu.c)^{\delta(\nu\in\mathcal{L}')}\right) \end{aligned}$$

6. $P_I(\mathcal{S}'|\mathcal{S})$ models the insertion operation. It gives the probability of extra *null* nodes being inserted. The probability is zero unless $\mathcal{S}' \supseteq \mathcal{S}$, since insertion never makes nodes disappear. Also, every inserted node must have the connector label *null* and not be aligned with any node in the original English tree, or the probability will be zero. Since all inserted nodes are internally identical, they all have the same probability.

$$\begin{aligned} P_I(\mathcal{S}'|\mathcal{S}) &= \prod_{\nu\in\mathcal{S}'} \Pr(\textsc{Ins}|\nu)^{\delta(\nu\notin\mathcal{S})} \\ &= \Pr(\textsc{Ins})^{\sum_{\nu\in\mathcal{S}'}\delta(\nu\notin\mathcal{S})} \end{aligned}$$

In summary, the basic building blocks of LinkSet's structured translation model are: $P_C(c)$, $\Pr(\textsc{HeadDel}|c)$, $P_T(f|e,t)$, $\Pr(\textsc{Del}|c)$, $\Pr(\textsc{Flip}|c)$, and $\Pr(\textsc{Ins})$. Once trained, these sub-models each have about 100-500 parameters. In the simplest version of LinkSet, all these sub-models except the translation model $P_T(f|e,t)$ are uniform at the lowest level shown here. Reasonable results can be obtained even with this simplified model. Then, the other sub-models can be trained using EM, which should give even better results.

3.4 EM Training

While the LinkSet bilingual parser works reasonably well using a structured translation model in which all sub-models are flat except for the word translation probabilities, it is reasonable to expect improved modeling if the parameters are trained using an EM approach.

The Inside–Outside Algorithm is a dynamic programming EM technique commonly used to parse context-free grammars. It involves calculating two kinds of probabilities with respect to each pairing of a non-terminal symbol with a particular span of words. The inside probability $\mathfrak{i}$ is the probability that the words in the current span are generated given the current non-terminal symbol. The outside probability $\mathfrak{o}$ is the probability of generating the words to the left of the current span, then the current non-terminal, and then the words to the right of the current span. Thus the product of the inside and outside probabilities associated with any particular non-terminal and span is the probability that the sentence is generated and contains that non-terminal in that position.

To adapt this algorithm for the LinkSet model, we use a major node $\langle \mathcal{L}, \mathcal{R} \rangle$ to fulfill the role of a non-terminal symbol. Thus the inside probability $\mathfrak{i}(\mathcal{L}, \mathcal{R}; \mathbf{f}_l^r)$ is the probability that the words $(f_l \ldots f_r)$ in the current span are generated, given the pair of sets of incoming connectors $\langle \mathcal{L}, \mathcal{R} \rangle$. The outside probability $\mathfrak{o}(\mathcal{L}, \mathcal{R}; \mathbf{f}_l^r)$ is the probability of generating the words $(f_1 \ldots f_{l-1})$ to the left of the current span, then the current pair $\langle \mathcal{L}, \mathcal{R} \rangle$, and then the words $(f_{l+1} \ldots f_n)$ to the right of the current span. The formulae below assume that if $l \geq r$ then $\mathfrak{i}(\mathcal{L}, \mathcal{R}; \mathbf{f}_l^r) = 0$ and $\mathfrak{o}(\mathcal{L}, \mathcal{R}; \mathbf{f}_l^r) = 0$. For brevity, we combine the contributions of the deletion, flipping (reordering), and insertion steps in a single probability distribution P_{DFI}:

$$P_{DFI}(\mathcal{L}, \mathcal{R}|\nu) \quad = \quad \sum_{\mathcal{L}', \mathcal{R}', \mathcal{L}''} \left[\begin{array}{c} P_D(\mathcal{L}''|\nu.L.\mathcal{R}) \; P_D(\mathcal{R}''|\nu.R.\mathcal{L}) \times \\ P_F(\mathcal{L}''|\mathcal{L}', \mathcal{R}') \times \\ P_I(\mathcal{L}|\mathcal{L}'') \; P_I(\mathcal{R}|(\mathcal{L}' \cup \mathcal{R}') - \mathcal{L}) \end{array} \right]$$

Figure 3.7 shows a node and its inside, dividing a range of words into two sub-ranges, and both left- and right-side variants of a node and its outside. These contexts should be helpful in interpreting the following formulae.

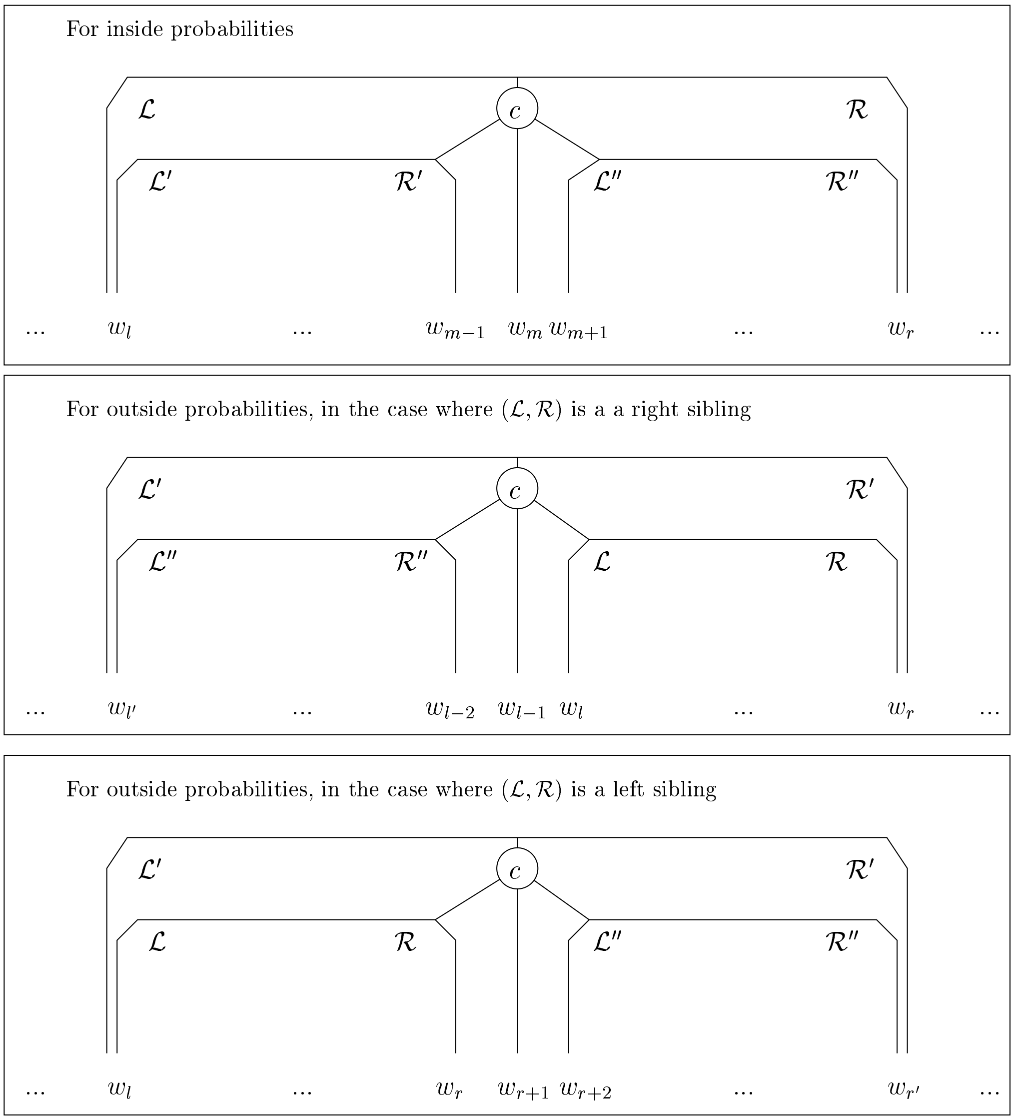

Figure 3.7: The recursive structure of $\langle \mathcal{L}, \mathcal{R} \rangle_{l,r}$

$$
\begin{aligned}
\mathfrak{i}(\mathcal{L},\mathcal{R};\mathbf{f}_l^r) &= \sum_{\nu_0\in\textsc{Top}(\mathcal{L},\mathcal{R})} P_C(\nu_0.c)\sum_{h=0}^{3}\sum_{\nu_1^h}\Pr(\neg\textsc{HeadDel}|\nu_h.c)\times \\
&\left[\begin{array}{c}\prod_{g=0}^{h-1}\Pr(\textsc{HeadDel}|\nu_g.c)P_C(\nu_{g+1}.c)\times \\ \delta(\nu_{g+1}.a\in\textsc{Children}(\nu_g.a))\end{array}\right]\times \\
&\sum_{l\le m\le r} P_T(f_m|\nu_h.a.\langle w,t\rangle)\sum_{\mathcal{R}',\mathcal{L}''} P_{DFI}(\mathcal{R}',\mathcal{L}''|\nu_h.a)\times \\
&\mathfrak{i}(\mathcal{L}-\{\nu_h\},\mathcal{R}';\mathbf{f}_l^{m-1})\ \mathfrak{i}(\mathcal{L}'',\mathcal{R}-\{\nu_h\};\mathbf{f}_{m+1}^r) \\
\mathfrak{o}(\mathcal{L},\mathcal{R};\mathbf{f}_l^r) &= \sum_{\mathcal{L}',\mathcal{R}'}\ \sum_{\nu_0\in\textsc{Top}(\mathcal{L}',\mathcal{R}')} P_C(\nu_0.c)\sum_{h=0}^{3}\sum_{\nu_1^h}\Pr(\neg\textsc{HeadDel}|\nu_h.c)\times \\
&\left[\begin{array}{c}\prod_{g=0}^{h-1}\Pr(\textsc{HeadDel}|\nu_g.c)P_C(\nu_{g+1}.c)\times \\ \delta(\nu_{g+1}.a\in\textsc{Children}(\nu_g.a))\end{array}\right]\times \\
&\left(\begin{array}{l}\left[\begin{array}{l}\sum_{1\le l'<l} P_T(f_{l-1}|\nu_h.a.\langle w,t\rangle)\ \mathfrak{o}(\mathcal{L}',\mathcal{R}';\mathbf{f}_{l'}^r)\times \\ \sum_{\mathcal{R}''} P_{DFI}(\mathcal{R}'',\mathcal{L}|\nu_h.a)\times \\ \mathfrak{i}(\mathcal{L}'-\{\nu_h\},\mathcal{R}'';\mathbf{f}_l^{m-1})\end{array}\right] \\ + \\ \left[\begin{array}{l}\sum_{r+1<r'\le n} P_T(f_{r+1}|\nu_h.a.\langle w,t\rangle)\ \mathfrak{o}(\mathcal{L}',\mathcal{R}';\mathbf{f}_l^{r'})\times \\ \sum_{\mathcal{L}''} P_{DFI}(\mathcal{R},\mathcal{L}''|\nu_h.a)\times \\ \mathfrak{i}(\mathcal{L}'',\mathcal{R}'-\{\nu_h\};\mathbf{f}_{m+1}^r)\end{array}\right]\end{array}\right)
\end{aligned}
$$

The formulae below follow this basic outline: For an item X, $\mathfrak{i}_x(x,C;\mathbf{f}_l^r)$ is the probability of generating the words $(f_l\ldots f_r)$ given that x occurs in context C, and $\mathfrak{o}_x(C;\mathbf{f}_l^r)$ is the probability of generating the context C (covering positions l through r), generating an X within that context, and generating the words outside that range.

$$
\begin{aligned}
\mathfrak{i}(\mathcal{L},\mathcal{R};\mathbf{f}_l^r) &= \sum_{\nu_0\in\textsc{Top}(\mathcal{L},\mathcal{R})} P_C(\nu_0.c)\ \mathfrak{i}_c(\mathcal{L},\mathcal{R},\nu;\mathbf{f}_l^r) \\
\mathfrak{i}_c(\mathcal{L},\mathcal{R},\nu;\mathbf{f}_l^r) &= \sum_{h=0}^{3}\sum_{\nu_1^h}\Pr(\neg\textsc{HeadDel}|\nu_h.c)\times \\
&\left[\begin{array}{c}\prod_{g=0}^{h-1}\Pr(\textsc{HeadDel}|\nu_g.c)P_C(\nu_{g+1}.c)\times \\ \delta(\nu_{g+1}.a\in\textsc{Children}(\nu_g.a))\end{array}\right]\times
\end{aligned}
$$

$$
\begin{aligned}
\mathfrak{i}_h(\mathcal{L},\mathcal{R},\nu_h;\mathbf{f}_l^r) &= \sum_{l\le m\le r} P_T(f_m|\nu_h.a.\langle w,t\rangle)\ \mathfrak{i}_t(\mathcal{L},\mathcal{R},\nu_h,m;\mathbf{f}_l^r) \\
\mathfrak{i}_t(\mathcal{L},\mathcal{R},\nu_h,m;\mathbf{f}_l^r) &= \sum_{\mathcal{R}',\mathcal{L}''} P_{DFI}(\mathcal{R}',\mathcal{L}''|\nu_h.a)\times \\
&\quad \mathfrak{i}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r) \\
\mathfrak{i}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r) &= \mathfrak{i}(\mathcal{L}-\{\nu_h\},\mathcal{R}';\mathbf{f}_l^{m-1})\ \mathfrak{i}(\mathcal{L}'',\mathcal{R}-\{\nu_h\};\mathbf{f}_{m+1}^r) \\
\mathfrak{o}_c(\mathcal{L},\mathcal{R},\nu;\mathbf{f}_l^r) &= \delta(\nu=\textsc{Top}(\mathcal{L},\mathcal{R}))\ P_C(\nu.c)\ \mathfrak{o}(\mathcal{L},\mathcal{R};\mathbf{f}_l^r) \\
\mathfrak{o}_h(\mathcal{L},\mathcal{R},h,\nu_0^h;\mathbf{f}_l^r) &= \left[\begin{array}{c}\prod_{g=0}^{h-1}\Pr(\textsc{HeadDel}|\nu_g.c)P_C(\nu_{g+1}.c)\times \\ \delta(\nu_{g+1}.a\in\textsc{Children}(\nu_g.a))\end{array}\right]\times \\
&\quad \mathfrak{o}_c(\mathcal{L},\mathcal{R},\nu;\mathbf{f}_l^r) \\
\mathfrak{o}_t(\mathcal{L},\mathcal{R},h,\nu_0^h,m;\mathbf{f}_l^r) &= P_T(f_m|\nu_h.a.\langle w,t\rangle)\ \mathfrak{o}_h(\mathcal{L},\mathcal{R},h,\nu_0^h;\mathbf{f}_l^r) \\
\mathfrak{o}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r) &= P_{DFI}(\mathcal{R}',\mathcal{L}''|\nu_h.a)\sum_{\nu_0^{h-1}}\mathfrak{o}_t(\mathcal{L},\mathcal{R},h,\nu_0^h,m;\mathbf{f}_l^r)
\end{aligned}
$$

We can then use these detailed inside and outside probabilities to update the counts C_X for each parameter $X\in\{C,H,T,D,F\}$. The formulae below show the contribution of a single sentence pair to the counts for each parameter. They are normalized by the probability of the sentence pair regardless of alignment.

$$
\begin{aligned}
\gamma &= \frac{1}{\mathfrak{i}(\mathrm{root};\mathbf{f})} \\
C_C(c) &= \gamma\sum_{\mathcal{L},\mathcal{R},l,r}\sum_{\nu}\delta(\nu.c=c)\ \mathfrak{o}_c(\mathcal{L},\mathcal{R},\nu;\mathbf{f}_l^r)\ \mathfrak{i}_c(\mathcal{L},\mathcal{R},\nu;\mathbf{f}_l^r) \\
C_H(\textsc{HeadDel};c) &= \gamma\sum_{\mathcal{L},\mathcal{R},l,r}\sum_{h=0}^{3}\sum_{\nu_0^h}\left(\sum_{g=0}^{h-1}\delta(c=\nu_g.c)\right)\times \\
&\quad \mathfrak{o}_h(\mathcal{L},\mathcal{R},h,\nu_0^h;\mathbf{f}_l^r)\ \mathfrak{i}_h(\mathcal{L},\mathcal{R},\nu_h;\mathbf{f}_l^r) \\
C_H(\neg\textsc{HeadDel};c) &= \gamma\sum_{\mathcal{L},\mathcal{R},l,r}\sum_{h=0}^{3}\sum_{\nu_0^h}\delta(c=\nu_h.c)\times \\
&\quad \mathfrak{o}_h(\mathcal{L},\mathcal{R},h,\nu_0^h;\mathbf{f}_l^r)\ \mathfrak{i}_h(\mathcal{L},\mathcal{R},\nu_h;\mathbf{f}_l^r) \\
C_T(f;e) &= \gamma\sum_{\mathcal{L},\mathcal{R},l,r}\sum_{\nu}\delta(\nu.a.w=e)\sum_{l\le m\le r}\delta(f_m=f)\times
\end{aligned}
$$

$$
\begin{aligned}
&\mathfrak{o}_t(\mathcal{L},\mathcal{R},c,m;\mathbf{f}_l^r)\ \mathfrak{i}_t(\mathcal{L},\mathcal{R},c,m;\mathbf{f}_l^r) \\
C_D(\text{DEL};c) &= \gamma \sum_{\mathcal{L},\mathcal{R},l,r} \sum_{\nu_h,m,\mathcal{R}',\mathcal{L}''} \sum_{\nu} \delta(\nu.c=c)\ \times \\
&\quad \delta(\nu.a \in (\nu_h.a.L \cup \nu_h.a.R) \wedge \nu \notin (\mathcal{R}' \cup \mathcal{L}'')) \times \\
&\quad \mathfrak{o}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r)\ \mathfrak{i}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r) \\
C_D(\neg\text{DEL};c) &= \gamma \sum_{\mathcal{L},\mathcal{R},l,r} \sum_{\nu_h,m,\mathcal{R}',\mathcal{L}''} \sum_{\nu} \delta(\nu.c=c)\ \times \\
&\quad \delta(\nu.a \in (\nu_h.a.L \cup \nu_h.a.R) \wedge \nu \in (\mathcal{R}' \cup \mathcal{L}'')) \times \\
&\quad \mathfrak{o}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r)\ \mathfrak{i}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r) \\
C_F(\text{FLIP};c) &= \gamma \sum_{\mathcal{L},\mathcal{R},l,r} \sum_{\nu_h,m,\mathcal{R}',\mathcal{L}''} \sum_{\nu} \delta(\nu.c=c)\ \times \\
&\quad \delta((\nu.a \in \nu_h.a.L \wedge \nu \in \mathcal{L}'') \vee (\nu.a \in \nu_h.a.R \wedge \nu \in \mathcal{R}')) \times \\
&\quad \mathfrak{o}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r)\ \mathfrak{i}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r) \\
C_F(\neg\text{FLIP};c) &= \gamma \sum_{\mathcal{L},\mathcal{R},l,r} \sum_{\nu_h,m,\mathcal{R}',\mathcal{L}''} \sum_{\nu} \delta(\nu.c=c)\ \times \\
&\quad \delta((\nu.a \in \nu_h.a.L \wedge \nu \in \mathcal{R}') \vee (\nu.a \in \nu_h.a.R \wedge \nu \in \mathcal{L}'')) \times \\
&\quad \mathfrak{o}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r)\ \mathfrak{i}_{dfi}(\mathcal{L},\mathcal{R},\nu_h,m,\mathcal{R}',\mathcal{L}'';\mathbf{f}_l^r)
\end{aligned}
$$

Once the counts are collected for all sentence pairs in the whole training set, we can reestimate the probabilities, as follows:

$$
\begin{aligned}
P_C(c) &= C_C(c) \ / \ \sum_{c'} C_C(c') \\
P_T(f|e) &= C_T(f;e) \ / \ \sum_{f'} C_T(f';e) \\
P_H(\text{HEADDEL}|c) &= \frac{C_H(\text{HEADDEL};c)}{C_H(\neg\text{HEADDEL};c) + C_H(\text{HEADDEL};c)} \\
P_D(\text{DEL}|c) &= \frac{C_D(\text{DEL};c)}{C_D(\neg\text{DEL};c) + C_D(\text{DEL};c)} \\
P_F(\text{FLIP}|c) &= \frac{C_F(\text{FLIP};c)}{C_F(\neg\text{FLIP};c) + C_F(\text{FLIP};c)}
\end{aligned}
$$

3.5 Search

The task of finding the most probable structure of a given foreign sentence given its structured English translation can be framed as a parsing problem. We use the known English structure to define the grammar rules that apply to each foreign sentence, and then parse the foreign sentence to determine how those rules apply, which is equivalent to determining an alignment between the English and foreign sentences. Recursive parsing with memoization allows the implicit enumeration of all possible parses, while a probabilistic model can simultaneously be used to weight all possible parses for the EM training of the model and to select the most probable parse. The *LinkSet Bilingual Parsing Algorithm* is an efficient way to deterministically enumerate all the grammar structures allowed by a bilingual structure model. It was designed with our syntax-tree transformations in mind, but it is also quite similar to the efficient algorithm used by the rule-based Link Parser. One insight that led to this algorithm is that it would be very wasteful to maintain a distinction between subtree deletions or reorderings that occur in different orders but result in the same tree. Thus in recursive parsing we want to delay decisions until they become relevant, so that we don't miss out on opportunities to reuse work through memoization because we picked a different value for some variable that has no effect on the result until a later stage. Therefore, while the Link Parser keeps track of stacks of connectors, the LinkSet Parser commits only to unordered sets of connectors.

Figures 3.8 and 3.9 show a rule-based version of the LinkSet parser that simply counts the possible parses. Probabilistic modeling and some optimizations have been abstracted away. At each recursive step in parsing, a range of the foreign sentence is parsed given a set $\mathcal{L}$ of connectors available on the left and a set $\mathcal{R}$ of connectors available on the right.

$$\mathcal{L} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathcal{R}$$

$$f_1 \quad f_2 \quad f_3 \quad f_4 \quad f_5 \quad f_6 \quad f_7 \quad f_8 \quad f_9$$

For each word in the range, an attempt is made to connect to that word using one of the incoming connectors. Each valid disjunct (there may be one or more for each word and each incoming connector ν) defines a set of outgoing connectors. The parser tries all subsets of that outgoing connector set, and then for each subset, tries all partitions of that set into left and right halves ($\mathcal{R}'$ and $\mathcal{L}''$ respectively).

procedure $\textsc{Parse}(\mathcal{L}, \mathcal{R}, \mathbf{f}_l^r)$
 if $n \leftarrow cache[\mathcal{L}, \mathcal{R}, l, r]$
  then return (n)
  else $n \leftarrow 0$
 if $L + 1 = R$
  then $\begin{cases} \textbf{if } \mathcal{L} = \emptyset \textbf{ and } \mathcal{R} = \emptyset \\ \quad \textbf{then return } (1) \\ \quad \textbf{else return } (0) \end{cases}$
 if $\mathcal{L} = \emptyset$ **and** $\mathcal{R} = \emptyset$
  then $\begin{cases} n \leftarrow \textsc{Parse}(\mathcal{L}, \mathcal{R}, \mathbf{f}_{l+1}^r) \\ cache[\mathcal{L}, \mathcal{R}, l, r] \leftarrow n \\ \textbf{return } (n) \end{cases}$
 if $\mathcal{L} \neq \emptyset$
  then $\begin{cases} \textbf{for each } \nu \in \mathcal{L} \\ \quad \textbf{do } \{n \leftarrow n + \textsc{Connect}(\mathcal{L} - \{\nu.c\}, \mathcal{R}, \mathbf{f}_l^r, \nu) \\ cache[\mathcal{L}, \mathcal{R}, l, r] \leftarrow n \\ \textbf{return } (n) \end{cases}$
 if $\mathcal{R} \neq \emptyset$
  then $\begin{cases} \textbf{for each } \nu \in \mathcal{R} \\ \quad \textbf{do } \{n \leftarrow n + \textsc{Connect}(\mathcal{L}, (\mathcal{R} - \{\nu.c\}), \mathbf{f}_l^r, \nu) \\ cache[\mathcal{L}, \mathcal{R}, l, r] \leftarrow n \\ \textbf{return } (n) \end{cases}$
 return (0)

Figure 3.8: The LinkSet bilingual parsing algorithm, simplified.

procedure $\text{C}\textsc{onnect}(\mathcal{L}, \mathcal{R}, \mathbf{f}_l^r, \nu)$

$$
\begin{array}{l}
n \leftarrow 0 \\
\mathbf{c} \leftarrow (\nu.L.\mathcal{R} \cup \nu.R.\mathcal{L}) \\
\textbf{for each } \mathrm{s} \subset \mathbf{c} \\
\quad \textbf{do} \begin{cases}
\textbf{for each } \mathcal{R}' \subset \mathrm{s} \\
\quad \textbf{do} \begin{cases}
\mathcal{L}'' \leftarrow (\mathrm{s} - \mathcal{R}') \\
\textbf{for each } l < m < r \\
\quad \textbf{do} \begin{cases}
n_L \leftarrow \text{P}\textsc{arse}(\mathcal{L}, \mathcal{R}', \mathbf{f}_l^m) \\
n_R \leftarrow \text{P}\textsc{arse}(\mathcal{L}'', \mathcal{R}, \mathbf{f}_m^r) \\
n \leftarrow n + n_L \times n_R
\end{cases}
\end{cases}
\end{cases} \\
\textbf{return } (n)
\end{array}
$$

Figure 3.9: The LinkSet bilingual parsing algorithm, continued.

Then the left and right sub-ranges are recursively parsed, using the appropriate connector sets, as shown in Figure 3.10. If there is no connector available on the left, then a connection comes from the right instead, as shown in Figure 3.11.

If a range has zero width, it is parsed successfully if and only if both incoming sets are empty. On the other hand, a range that contains words may be parsed even with empty connector sets through the use of *null* links. This allowance makes the parser robust, able to continue when there is no known way to connect part of a sentence, as shown in Figure 3.12.

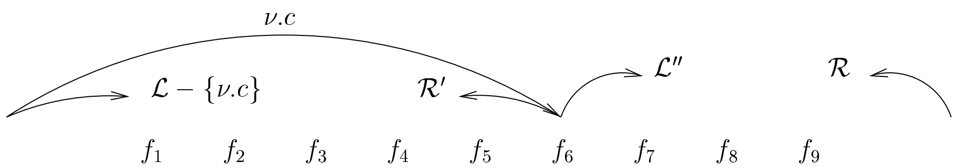

Figure 3.10: The left and right sub-ranges are recursively parsed.

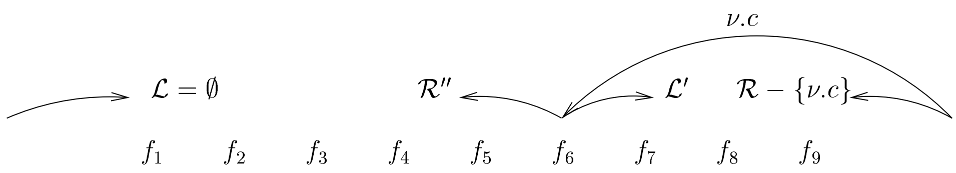

Figure 3.11: A connection comes from the right instead.

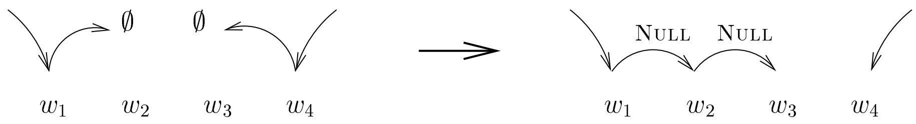

Figure 3.12: Robust parsing with *null* links.

3.6 Results

It is difficult to evaluate the results of bilingual parsing directly, because it is hard to define the "correct" answers, since the relationship between two languages is complex and messy. The following two chapters, which use these results to induce a monolingual parser and to rerank translation hypotheses, will provide better opportunities to evaluate its effectiveness. However, since bilingual parsing implicitly aligns the words of each bilingual sentence pair, we can evaluate its performance by selecting the highest-scoring alignment and comparing it to a hand-aligned standard.

3.6.1 The Corpora

In the following experiments we used several different corpora, each bilingual between English and one of French, Romanian, or Chinese. Each corpus is aligned sentence-by-sentence. Perhaps the most well-known corpus in the statistical translation community is the Canadian Hansard corpus, which comprises over a million sentences transcribed from the Canadian parliament, which is conducted in both English and French. We used two different collections of Hansard data, with different sizes and different preprocessing, obtained from two different intermediate sources. For Romanian and English, we used a corpus consisting of news, the

Romanian Constitution, and text from Orwell's novel *1984*. For Chinese and English, we used a combination of news sources, including Hong Kong News and Xinhua News. Table 3.1 shows the amount of data used from each corpus, along with vocabulary sizes.

3.6.2 Training a Word-Translation Model

The first step in bilingual parsing, after gathering a bilingual corpus, is to train a word-translation model, which will form the basis for word-to-word alignment of the corpus, and for the structured translation model of which word-translation is a key component. To train the word-translation parameters of the bilingual structure model, we used GIZA++, an extension by Franz Josef Och of a toolkit developed by by the Statistical Machine Translation team during the summer workshop in 1999 at the Center for Language and Speech Processing at Johns-Hopkins University [22].

To reduce the perplexity of the word-translation model, a few easily-translated types of words were replaced with tokens, such as NUMBER or DATE. Most punctuation was separated from adjacent words.

In order to provide some additional smoothing, we trained an additional model that translated from an English part-of-speech tag to a foreign word. We modified GIZA++ to train these models jointly, maximizing the probability of the training data given a linear combination of the word-to-word and tag-to-word models. The same linear combination of these models was then used for bilingual parsing. This smoother model is especially helpful for less-common words, for which the words-only model may be less accurate.

In order to build a model based on POS tags, of course, the English side of the corpus must be tagged. We used Eric Brill's rule-based tagger [3].

	training pairs used		vocabulary size	
Corpus	words	structure	English	foreign
English–French(A)	500k	250k	48k	65k
English–French(B)	1130k	250k	44k	62k
English–Chinese	282k	90k	49k	72k
English–Romanian	50k		26k	44k

Table 3.1: Sentence-pairs used in various corpora to train a word-translation model and a structured translation model; vocabulary sizes. (k=1000)

A second application of POS tags to translation is word-sense disambiguation. In order to distinguish various senses of the same English word, we built a translation model in which each English word was appended with a POS tag. Thus word-forms that represent multiple parts of speech have separate translation models for each type, which makes sense linguistically because in a different language they are likely to have separate forms. We tested this model only on the Chinese–English language pair.

3.6.3 Evaluation of Alignment

One way to test how well LinkSet works is to compare the alignments it produces with "correct" alignments specified by a language expert. While alignments are somewhat subjective, they do provide a quantifiable measure that allows comparison between different techniques on the same test data. To provide context for LinkSet's alignment scores, a baseline and oracle are also scored. The baseline is an alignment using the basic word-translation model, but no syntactic structure, to select the most probable English counterpart for each foreign word. More specifically, the baseline was computed by selecting the most probable English word to align to each foreign word, according to the same word-translation model used as a component of the LinkSet model, which is equivalent to IBM Model 1. This word-translation model was trained using IBM Model 2. The oracle gives the best alignment possible under the constraints that the alignment must be one-to-one and that it must be possible to derive the foreign sentence structure from the English structure by means of the transformations described above (Section 3.3). The best alignment, for the oracle, is defined as the best recall possible at 100% precision. Showing the degree of mismatch between the assumptions of the alignment framework and the particular evaluation set, the oracle provides an upper bound on the possible performance of LinkSet, thus putting the LinkSet alignment scores in perspective.

For French–English bilingual parsing, two different training sets and two different test sets were available. For our first experiment, we used training set A, which consists of 500 thousand sentence pairs, and has relatively clean preprocessing. We had 65 sentences aligned by hand, and compared these alignments to those found by LinkSet. We compared two different mixtures of the word-translation model, either using POS tags or not. We also trained a structural translation model on 250 thousand sentence pairs, of which the English side was parsed with the Link parser, in two rounds of EM, comparing the results after 0, 1, and 2 rounds. Table 3.2 shows the results. The best result was achieved using the simple

Model	EM	recall	precision	F_1-score
baseline		64.5%	47.3%	54.6%
$\lambda \Pr(f\|e) + (1-\lambda) \Pr(f\|t)$	0	68.2%	54.5%	60.6%
	1	69.8%	56.3%	62.3%
	2	67.9%	54.3%	60.4%
$\Pr(f\|e)$	0	74.4%	61.1%	67.1%
	1	73.2%	60.5%	66.2%
	2	**74.7%**	**61.7%**	**67.6%**
oracle		83.6%	100.0%	91.1%

Table 3.2: Alignment results for **French**, using word-translation models trained on 500k sentence pairs (training set A) and a structure model trained on 250k sentence pairs (using 0-2 rounds of EM), tested on 65 hand-aligned sentence pairs.

word translation model (no tags) and 2 rounds of EM, exceeding the score of the baseline by over 12 percentage points.

A second French–English experiment was evaluated on a larger test set provided for the HLT-NAACL 2003 Workshop Shared Task. In addition to comparing translation models with and without POS tags and comparing amounts of EM training for the structural translation model, we also used two different training sets: set A, used in the previous experiment, and set B, which was also provided for the HLT-NAACL workshop. Table 3.3 shows the results. The best result was achieved using simple word translation model (no tags) and 1 round of EM. However, the best recall was achieved using the word-translation model with tags. Training set A consistently gave better results than training set B, probably because the preprocessing was better for this set, although it is possible that the larger set (B) was disadvantageous for some other reason.

To test Chinese–English alignment accuracy, we used three different combinations of word-translation model: one using words smoothed with tags, one using tags to disambiguate words, and one just using words. We also built a structural translation model, and compared results without this model, and using the versions after one or two rounds of EM. We tested these combinations on 183 hand-aligned sentence pairs from Xinhua news. The results, visible in Table 3.4, show that the simple words-only translation model and the structural translation model after one round of EM combine to achieve the best score.

In this experiment, the second round of EM caused a slight degradation in

Model	Training Set	EM	recall	precision	F_1-score
baseline			60.2%	35.0%	44.3%
$\Pr(f\|e)$	B	0	58.3%	37.3%	44.3%
		1	60.0%	38.5%	45.5%
		2	59.9%	38.4%	46.8%
	A	0	62.1%	40.1%	48.7%
		1	62.7%	**40.5%**	**49.2%**
		2	62.3%	40.4%	49.1%
$\lambda \Pr(f\|e)+$ $(1-\lambda)\Pr(f\|t)$	B	0	60.1%	38.1%	46.8%
		1	60.6%	38.1%	46.8%
		2	60.4%	38.0%	46.6%
	A	0	62.4%	39.3%	48.2%
		1	**63.2%**	39.9%	48.9%
		2	62.6%	39.5%	48.4%
oracle			70.4%	100.0%	82.6%

Table 3.3: Alignment results for **French**, using a structure model trained on 250k sentence pairs (using 0-2 rounds of EM), tested on 484 hand-aligned sentence pairs.

Model	EM	recall	precision	F_1-score
baseline		60.5%	58.9%	59.7%
$\lambda \Pr(f\|e) + (1-\lambda)\Pr(f\|t)$	0	62.2%	55.3%	58.5%
	1	62.5%	55.6%	58.9%
	2	61.7%	54.8%	58.0%
$\Pr(f\|e,t)$	0	66.7%	55.3%	63.4%
	1	66.6%	55.6%	63.5%
	2	65.9%	54.8%	62.8%
$\Pr(f\|e)$	0	66.8%	63.4%	65.1%
	1	**67.0%**	**64.0%**	**65.4%**
	2	66.4%	63.6%	65.0%
oracle		78.5%	100.0%	88.0%

Table 3.4: Alignment results for **Chinese**, using word-translation models trained on 282k sentence pairs, and a structure model trained on 90k sentence pairs (using 0-2 rounds of EM), and evaluated on 183 hand-aligned sentences.

	recall	precision	F_1-score
baseline	39.7%	54.3%	45.8%
$\Pr(f\|e)$	49.9%	69.0%	57.9%
$\lambda \Pr(f\|e) + (1-\lambda) \Pr(f\|t)$	**50.4%**	**69.5%**	**58.4%**
oracle	56.7%	100.0%	72.3%

Table 3.5: Alignment results for **Romanian**, using a word-translation model trained on 50k sentence pairs, without a structure model, and evaluated on 13 hand-aligned sentences.

	Romanian	Chinese	French	French
sentences	13	183	65	484
baseline	45.8%	59.7%	54.6%	44.3%
LinkSet	50.4%	65.4%	67.6%	49.2%
oracle	72.3%	88.0%	91.1%	82.6%
improvement	17.4%	20.1%	35.6%	12.8%

Table 3.6: Cross-linguistic comparison of alignment results, showing F_1-scores for the baseline, the best LinkSet configuration, and the oracle. The portion of the space between the baseline and oracle claimed by LinkSet is shown as "improvement."

performance. This was also true in the French–English experiments, in 5 out of 6 cases. This is probably caused by overfitting. While using a structural translation model still improves results in only 6 of the 9 EM comparisons, these include the best results for each test set. Thus the EM-trained structural translation model improves alignment accuracy.

Finally, since we had the data, we did a quick test of alignment between Romanian and English. A small Romanian–English corpus, along with test data, was provided for the 2003 HLT-NAACL workshop. Minimal effort was expended to train a word translation model using 50k sentence pairs. LinkSet's alignment results were evaluated, along with a baseline and oracle, on hand-aligned test data, and the results are shown in Table 3.5.

A summary of the four experiments is shown in Table 3.6. In all four cases, the LinkSet bilingual parser was able to align the words of bilingual sentence pairs significantly better than a baseline technique which took the most probable alignment according to the word-translation model without any structural information.

3.6.4 Examples

In order to show more qualitatively what the results of bilingual parsing with LinkSet look like, representative sentence pairs were selected from among the results of the experiment above. Above-average pairs were chosen, since mostly-correct answers are more interesting to examine than incorrect answers. Figures 3.13 and 3.14 show selected pairs of French and English sentences, and Figure 3.15 shows selected pairs of Chinese and English sentences. Each of the English sentences was parsed using the Link parser, and the resulting structure is shown above the sentence. The lines connecting English words to foreign words show the most probable alignment found by LinkSet. Beneath the foreign sentence is the structure induced through bilingual parsing. Dashed lines indicate incorrect alignments made by LinkSet, and dotted lines indicate true alignments incorrectly omitted by LinkSet. Solid lines show correct alignments found by LinkSet. The following paragraphs offer some comments on the sentences shown.

- *The problem he raised is indeed a real one.*
  The parse and the alignment are correct. Some French words do not have English counterparts and are left out, but the induced French structure is otherwise correct.

- *Mr. Speaker, may I direct a question to the Minister of Labour.*
  The English parse is slightly incorrect, leaving out the word *may*, which results in the corresponding omission of *may* from the alignment and the French structure.

- *As this is what is required, it is the target the government has in mind.*
  While three of the alignments shown here (in dashed lines) are considered incorrect by the test set, not having the same meanings word-for-word, it could be argued that they are actually correctly aligned: *as* and *et*, while they don't have exactly the same meaning, both serve structurally to join two statements; the phrases *is required* and *il faut* mean the same thing, even though their individual words do not. Since this system allows only one-to-one alignments, it is not possible to align *what* to both *ce* and *que*.

- *Federal government carpenters get \$6.42 in Toronto and \$5.23 in Halifax and Moncton.*

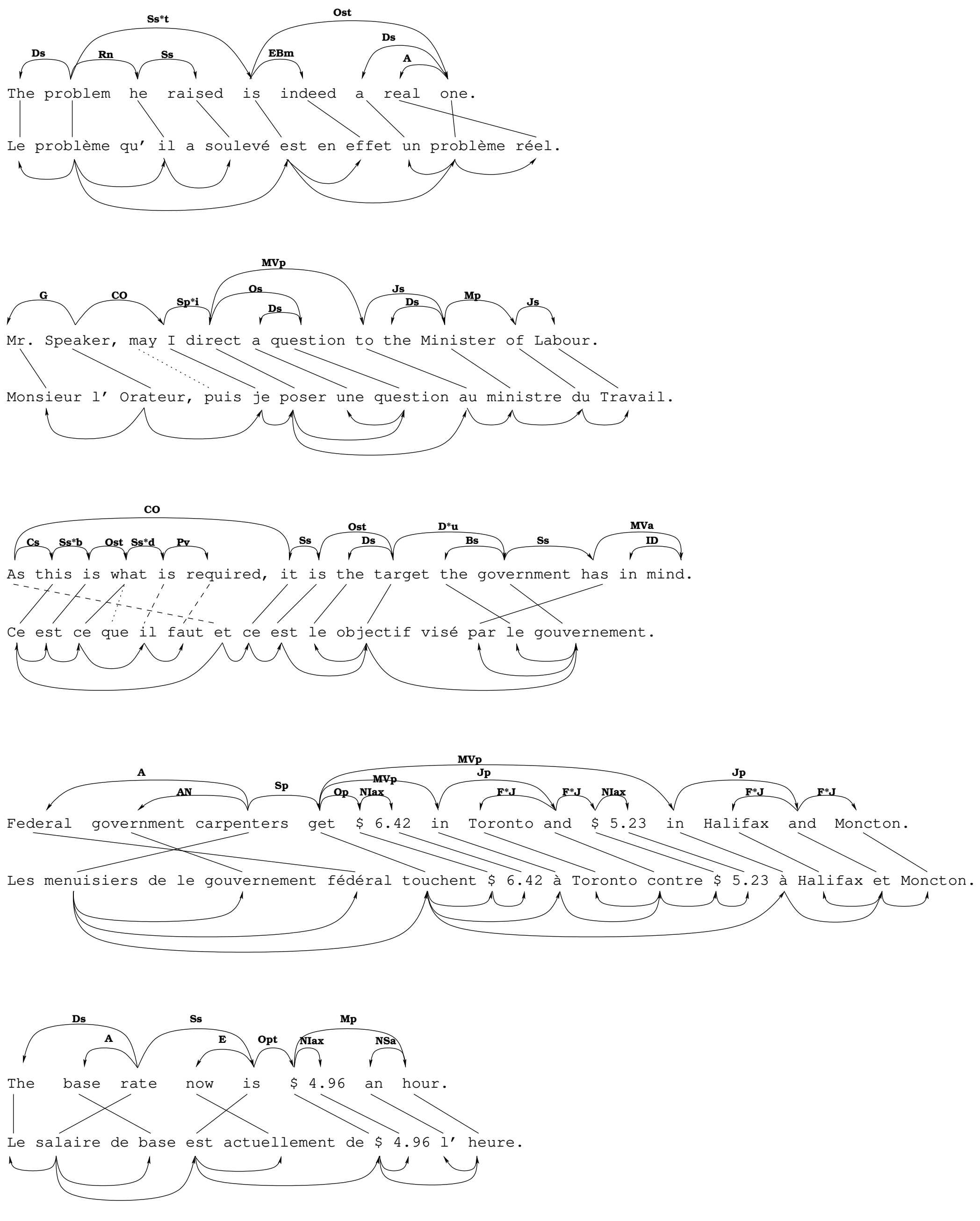

Figure 3.13: Selected French–English sentence pairs aligned with LinkSet.

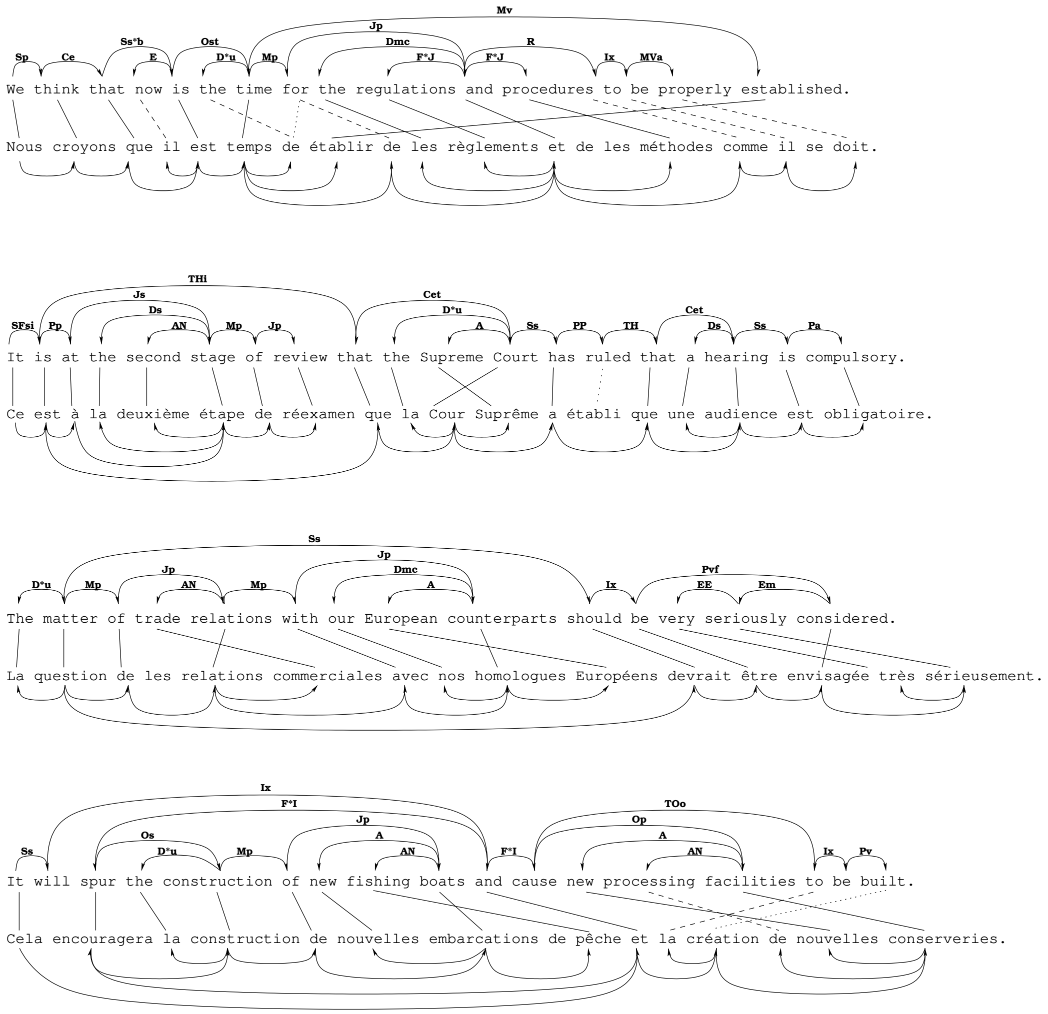

Figure 3.14: Selected French–English sentence pairs aligned with LinkSet.

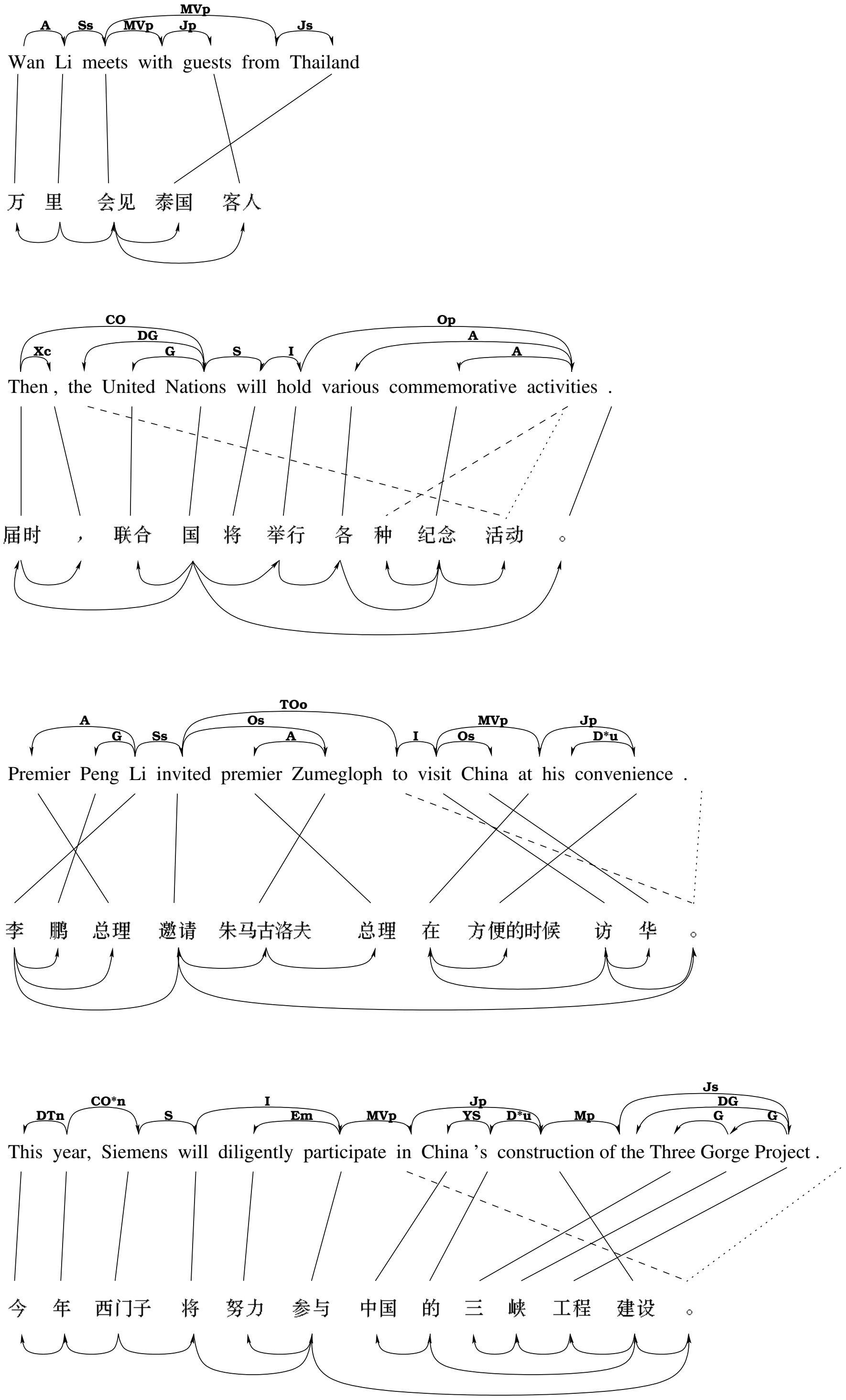

Figure 3.15: Selected Chinese–English sentence pairs aligned with LinkSet.

The analysis shown is correct. The short French words that are not aligned to English are correctly connected to the induced French structure using *null* links (not shown).

- *The base rate now is \$4.96 an hour.*
  The analysis shown is correct.

- *We think that now is the time for the regulations and procedures to be properly established.*
  The English parse is incorrect, particularly in connecting *established* to *time* rather than *be* and *properly*. This, combined with the fact that the French sentence has a slightly different structure, causes problems for the alignment and the induced structure.

- *It is at the second stage of review that the Supreme Court has ruled that a hearing is compulsory.*
  The analysis shown is correct, except for the missing alignment between *ruled* and *tabli*.

- *The matter of trade relations with our European counterparts should be very seriously considered.*
  The analysis shown is correct.

- *It will spur the construction of new fishing boats and cause new processing facilities to be built.*
  The analysis shown is mostly correct, but two spurious alignments (shown in dashed lines) connect common words that have no correct alignment available. Common words tend to have flatter translation models, allowing them to connect to almost any word when under pressure from the structure. The desired alignment of *built* to *cration* is not possible (in the configuration tested) because it would require too many consecutive head-deletion operations.

- *Wan Li meets with guests from Thailand*
  The analysis shown is correct. The two English prepositions have no equivalent Chinese words.

- *Then, the United Nations will hold various commemorative activities.*
  The analysis shown has two misaligned words, shown by dashed lines, with the dotted line showing a better alignment that was not selected by LinkSet.

Since the system is biased in favor of aligning more words rather than leaving them unaligned, the correct alignment would get a lower score due to its decrease in the number of aligned words.

- *Premier Peng Li invited premier Zumegloph to visit China at his convenience.*
  The analysis shown is mostly correct, but aligns one pair of words (shown by dashed lines) that should each be left unaligned. These are both very common words, and thus have flat translation models, making it easy to erroneously align them to each other.

- *This year, Siemens will diligently participate in China's construction of the Three Gorge Project.*
  Just as in the previous sentence, the analysis shown is mostly correct, but aligned a common English word to a common Chinese punctuation mark.

3.7 Discussion

Because the LinkSet Bilingual Parser is an intermediate step leading to monolingual parsers for other languages, and to structured transformation models for translation, evaluating it directly in terms of word-to-word alignment fails to fully describe its effectiveness. However, it seems reasonable to expect that the usefulness of the models trained using LinkSet would be limited by its ability to assign meaningful correspondences between the English and foreign versions of a training sentence. The results shown above, while they fall short of what oracles show may be possible with better modeling, are good enough to inspire hope that the noise introduced by incorrect alignments will not overwhelm the useful content in the models derived from the alignment process for the purposes of translation and monolingual parsing. The following two chapters will demonstrate to what degree models trained with LinkSet achieve practical results in spite of noisy alignments.

As the experiments above show, EM training of the bilingual parsing model was helpful in improving the quality of alignments found. In most cases, a single round of EM offered a small boost in alignment quality, while successive rounds did not. Overall, the gain offered by EM was fairly small.

In this section we will discuss both the model we used to align sentences within this framework, and the framework itself. Here we refer to the structure of the syntax trees and the valid transformations on those trees as the *framework*, and we refer to the way the probabilities of specific transformations are calculated as the *model*.

Our goals in the design of the LinkSet model include the following. The model must match the framework; that is, it must provide a probability distribution covering the same transformations allowed by the framework. Each part of the model should be detailed enough to capture the relevant features that enable the system to distinguish and prefer the appropriate transformation in each case, but at the same time be general enough to avoid the twin pitfalls of computational intractability and data sparsity. Also, for the sake of robustness, it is desirable that the model never assign zero probability to a sentence pair, however poorly it fits the framework, but allows some best guess (however weak) to be selected.

When designing the LinkSet model, we defined a sub-model for each transformation defined in the framework, and simplified the parameterization of each sub-model to avoid sparsity and to keep the search algorithm relatively efficient. This was apparently successful. However, it may be that some of these simplifications went too far, such that increasing the parameterization of one or two models in a strategic way might pay off in terms of improved discrimination and better alignment results, without a large cost in terms of increased data sparsity, model size, and computational complexity.

One candidate for this reparameterization is the probability $P_C(\nu|\mathcal{L},\mathcal{R}) \approx P_C(\nu.c)$ of selecting a node ν as the next connector. It seems reasonable that the probability of a particular connector appearing at a syntactic node be conditioned on the set of alternatives, but this is problematic since there are too many possible sets of alternatives. Perhaps some representation could be devised to partition the space of sets of connectors in a meaningful way, which might then allow this connector-selection probability to be more effective.

A second candidate for reparameterization is the reordering operation. Because sparsity sets in if we try to parameterize on a set of connectors, we model reordering as a series of independent binary decisions, whether or not to flip a connector to the opposite side from its original position, relative to the parent node. This solves the sparsity and efficiency problems, but perhaps it leaves something lacking in discriminative power. One problem with the current approach is that connectors may have to "decide" whether to flip each time they are passed down to a lower node, which happens when a sibling is chosen first to head the next major node. It seems that the probability of flipping should depend not only on the label of the connector that might flip, but also on its parent connector.

These two candidates work together to determine the new ordering of connectors in a transformed syntax tree. They do so in a somewhat unintuitive way due to the inside-out structure of the efficient LinkSet parsing algorithm, a structure inherited from the Link parser, in which it was less confusing. It would be nice to be able to model the reordering of nodes in a more intuitive way, which probably means directly transforming the order of a set of connectors which would be siblings in the original, non-binary, tree representation, rather than pulling the binary tree structure of the LinkSet algorithm into this part of the model. However, this breach between the structure of the model and the algorithm, while technically not breaking the LinkSet framework, would probably have a similar effect on the efficiency of the algorithm, or at least on the comprehensibility of the modified algorithm.

A basic feature of this framework is that the children of a node may have a different ordering across languages. Combined with the fact that two ostensible translations may in fact have considerable mismatch, and that the translation probabilities of many common words are fairly flat, this occasionally yields bizarre results. In one relatively common case, the inferred structure of the foreign sentence has a link jumping from the first word to the end of the sentence, and then back again to the second word. In such cases, the word near the end of the sentence is a misaligned common word that has no direct complement in the English sentence. A better analysis would be to perform head-deletion, replacing the pair of links that connect the first and second word via the last word with a direct link, and attaching the last word to a neighbor with a null link. However, the current system is biased against null links for the obvious reason that a preference for null links would result in many sentence pairs with no alignment at all. What is needed is better modeling of the probability of using a null link. The easiest way to approach this would be to reparameterize the probability $\Pr(\textsc{Ins})$ of inserting a new subtree using a null link so that it depends on the word being connected by the null link, and perhaps also on its parent node.

The binary tree structure used in this framework lends itself quite well to efficient computation, which is a very important factor when dealing with experiments that may require hundreds of hours of CPU time. However, as we observed in our analysis of possible shortcomings in the model, it would be more intuitive to have an N-ary tree, allowing each node to connect directly to the nodes with which it has a close linguistic relationship. The value of an "intuitive" model is more than just that of being easy to understand. A framework that more closely follows linguistic intuition is more likely to capture the relevant distinctions in the parameterization of its model. However, such a model will be susceptible to computational inefficiency, parameter bloat, and data sparsity, unless great care is taken to design the framework, model, and algorithm well.

The LinkSet framework in some sense assumes that cross-lingual word alignments are one-to-one, an obvious untruth that varies in severity depending on the particular corpus. However, tree structure gives this assumption some wiggle-room. For example, if a two-word phrase in English matches a single word in another language, the head of the two-word phrase could be aligned with the single-word translation. Conversely, if a single English word matches a two-word foreign phrase, the head of the foreign phrase could be aligned to the English word, and the other foreign word could be null-linked to its head. Thus the structure deduced for the foreign phrase would still be correct. Variants of this analysis exist for other numbers of words on each side. A better analysis, however, might be to align the head of an opaque multi-word phrase with the head of its translation, and not try to force the words inside the phrases to align, when they clearly do not have a direct correspondence. In this case, the members of the foreign phrase would be connected to its head by null links. This is what LinkSet does, given a well-trained model. However, in this case it might make more sense to assign the foreign phrase a flat structure, with each word linked to the head, rather than putting null links in series (a right-branching structure); perhaps this decision is language dependent, or should be conditioned on the tag of the head (distinguishing noun phrases and verb phrases). In order to do this well, the model would need to have a better way of determining the boundaries of an opaque phrase. One way to do this would be to augment the word-translation model with phrase-to-phrase translations. Another possibility would be to use monolingual models to estimate the probability that a particular phrase is opaque to translation. While the LinkSet framework does not explicitly account for multi-word phrases, it is flexible enough to align them in a reasonable way. However, a framework and model that explicitly handle such alignments should be able to learn their correct mappings more readily.

In summary, the design of the LinkSet framework and model has been directed toward the goal of effectively modeling the data-driven mapping of sentence structure across languages along a pathway that avoids the swamp-lands of overly simplistic modeling on one side, and the twin pitfalls of computational complexity and data sparsity on the other. Our compass for this journey has been the incorporation of linguistic insight into the structure of statistical models. That is, we used linguistic intuition to guide the design of our model so it would capture the relevant phenomena. This approach not only gives a model the right discriminative power, but also helps to make it tractable and trainable, by removing dependencies that linguistic intuition tells us are less relevant. The right simplifications are crucial if a model is to avoid the ill effects of data sparsity and overly-taxing computational and storage requirements. When the need to discriminate and the need to simplify collide, we can take comfort in the fact that computers are still getting faster, and today's overnight job could be real-time in a decade or two.

One of the key linguistic insights guiding this framework is that the recursive structure of a sentence can be much the same in two different languages even when different ordering of that syntax tree makes the observed strings completely different. This simple idea has taken us a long way, but it has also revealed some limitations. The deepest of these is that a good translation of certain thoughts into a given language might have a completely different structure from its English equivalent. In these cases, the correspondence must simply be "memorized," though perhaps even a completely idiomatic translation would still be amenable to some generalization through modeling. It remains an open question how best to integrate this kind of mapping into a generative model based on deep structural similarity.

The LinkSet bilingual parser demonstrates a data-driven statistical translation model that brings statistical modeling closer to capturing the structural syntactic relationships between two languages. It is reasonably effective at aligning the words of bilingual sentence pairs in an unsupervised way, and in it is laid the groundwork for the two applications presented in the next two chapters: inducing a monolingual parser through bilingual parsing, and reranking translation hypotheses with a bilingual parsing model.

Chapter 4

Inducing a Monolingual Parser through Bilingual Parsing

4.1 Motivation

Natural language parsers, which analyze human-language sentences, are useful for a wide variety of tasks, including (but not limited to) dialog, translation, classification, and corpus adaptation. Unfortunately, parsing is a hard problem, and constructing a new parsing system that works reasonably well typically requires a large effort by language experts, either in designing a grammar or in annotating training data. However, by training the Lynx parser (described in Chapter 2) using data annotated automatically by the LinkSet bilingual parser (described in Chapter 3), a quick-and-dirty parser can be constructed with minimal effort and without much need for language expertise. Incremental improvements can then be made by addressing the various deficiencies of the resulting monolingual parser.

4.2 Related Work

Unsupervised methods for inducing a part-of-speech tagger and a noun-phrase bracketer across a bilingual corpus were put forward in 2001 by Yarowsky and Ngai [29]. Soon thereafter, Yarowsky, Ngai and Wicentowski added morphological analysis to their suite of induced text analysis tools [30]. Starting with an English POS tagger and standard IBM Model 3 alignments, they induced POS tags in the other language by copying tags across alignments, and trained a POS tagger on the other side. Similarly, they induced a NP bracketer by copying NP identifiers across alignments. Focusing on robust learning methods for noisy alignment data, they showed that robust projection outperforms simple projection even on hand-aligned data. This work is similar to ours in two key ways. First, alignments are used to project grammar knowledge from English to another language. Second, both depend on noise-robust learning techniques, since both the alignments and the grammar knowledge, not to mention the translations themselves, tend to be too noisy for traditional supervised-learning techniques.

4.3 Model

Combining the model of monolingual sentence structure $\Pr(\mathcal{E})$ described in Chapter 2, and the model $\Pr(\mathcal{F}|\mathcal{E})$ of a structured foreign sentence given a structured English sentence described in Chapter 3, we can in principle derive a model of monolingual sentence structure $\Pr(\mathcal{F})$ for the foreign language.

$$\Pr(\mathcal{F}) = \sum_{\mathcal{E}} \Pr(\mathcal{E}) \Pr(\mathcal{F}|\mathcal{E}) \tag{4.1}$$

That model can then be used to parse foreign sentences, by finding the most probable structure consistent with the given foreign sentence string **f**.

$$\hat{\mathcal{F}}_{\mathbf{f}} = \operatorname*{argmax}_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}) \tag{4.2}$$

However, in practice the model composition shown in Equation 4.1 is infeasible, since it requires a sum over all (infinitely many) possible English sentence structures for each French structure. We can estimate this sum using a large bilingual corpus, so that only attested sentence pairs contribute to a smoothed model of French structure. That is, we restrict the sum over $\mathcal{E}$ in Equation 4.1 to English structures that actually co-occur with the French structure $\mathcal{F}$ in our training corpus.

Without some kind of smoothing, the resulting model would assign probability mass only to those foreign sentence structures that had actually been seen in our training corpus. However, since both the translation model $\Pr(\mathcal{F}|\mathcal{E})$ and the monolingual structure model $\Pr(\mathcal{F})$ have simplifications and smoothing built in, the gross simplification of using only observed $\langle\mathcal{E},\mathcal{F}\rangle$ pairs does not result in an overly rigid model. Another reason this works is that the probability of the vast

	Penn Treebank		Hand Bracketed	
system	recall	precision	recall	precision
Lynx	75.1%	80.1%	91.6%	93.0%
Link	78.8%	86.6%	92.0%	93.5%
baseline	7.9%	7.8%	68.2%	68.2%

Table 4.1: Recall and precision of 100 English sentences on Lynx, Link and a right-branching baseline, according to two evaluation metrics.

majority of English sentences, given any particular foreign sentence, will be very close to zero, as these will not be valid translations, so zero is a reasonable estimate. Only those English structures that form possible translations of the given foreign sentence need be considered. Limiting this to a single translation is less desirable, but not unreasonable, considering the limitations of available corpora.

Using a large parallel corpus, we first perform bilingual parsing using LinkSet, selecting the most probable sentence structure $\mathcal{F}$ for each foreign sentence. Then this set of structured foreign sentences is used to train the Lynx parser, building a model $\Pr(\mathcal{F})$ in exactly the same way we built the model of English structure $\Pr(\mathcal{E})$ in Chapter 2. Once this model is built, we can use it to parse foreign sentences.

4.4 Results

To test this ability to quickly build a working parser for a foreign language, we used two bilingual corpora (French–English and Chinese–English) to automatically build monolingual parsers for French and Chinese. We then tested these parsers by parsing test sets in each language, and comparing the structures assigned by the new parsers against hand-bracketings of the same test sentences.

4.4.1 Comparing Linkages and Bracketings

The evaluation of a parser is somewhat of a subjective matter, since even human experts may disagree on what is the correct structure of a given sentence. However, a commonly-used metric for syntactic parsing is constituent bracketing, because it allows a given structure to be compared against a hand-bracketed canonical structure, yielding easily-interpreted recall and precision percentages.

1. Locate all constituents according to the canonical bracketing.
2. Find all subtrees of the linkage, excluding single leaf nodes.
3. For each subtree:
   (a) If this subtree is coextensive with a constituent, consider them matched.
   (b) Otherwise, for every constituent inside the subtree:
      i. If this subtree and constituent are consistent (no crossing), consider them matched. This may result in multiple constituents matching the same subtree.
      ii. Otherwise, consider this constituent to have failed to match.
4. The recall is the number of matches divided by the total number of constituents.
5. The precision is the number of matches divided by the number of constituents that either matched or failed to match a subtree.

Figure 4.1: An algorithm for link grammar bracket matching

The difficulty of evaluating parsing results is compounded when a formalism such as link grammar is used rather than the more traditional phrase structure grammar, which corresponds directly to a constituent bracketing. Link grammar structures can not be translated unambiguously to constituent bracketings, so instead of translating each linkage into a bracketing for comparison against the corresponding canonical bracketing, we compared linkages against canonical bracketings in a way that accounts for the ambiguity present both in linkages and in incomplete bracketings. An algorithm for computing the new metric is shown in Figure 4.1. To verify the validity of this evaluation metric, we used it to score some English structures that had already been evaluated using the old method (from Chapter 2) in which each structure was translated into a constituent bracketing using language-specific heuristics, and then compared with a bracketing for the same sentence from the Penn Treebank. Each of the two evaluation metrics works with different bracketing formats, so the sentences had to bracketed again by hand. The results of this comparison are shown in Table 4.1. While the numbers are not the same for the two methods, they are somewhat similar. Our goal in any case is not just to duplicate the results of the Penn Treebank metric in a language independent way, but also to avoid its problem with arbitrary interpretation of ambiguous structure. This evalution metric, then, allows linkages to be compared with bracketings in languages other than English, without arbitrarily forcing ambiguous structures into a particular bracketing, and without requiring hard-to-build and unreliable language-specific software to convert linkages to bracketings for each language.

4.4.2 Refining the Model

The training data given to the Lynx parser in these experiments, which was generated using the LinkSet bilingual parser, was quite noisy, a result of both word-level translation ambiguity and of mismatches in structure between many sentences and their translations. In an attempt to reduce the negative effects of this noise, we used two approaches: EM and attenuation.

Since the model used by the Lynx parser is too complex to allow proper EM training without significant redevelopment of the system, we used a crude approximation of EM by simply re-weighting each sentence structure in the training data based on the score the previous model assigned to that structure. One attempt, which used these scores directly as weights, was rapidly overtrained and gave very poor results. Therefore we reduced the learning rate by weighting each training round based on a function of the weights given by the previous round, plus

	EM	recall	precision
Lynx	0	**76.4%**	**80.5%**
Lynx	1	71.9%	76.7%
Lynx	2	73.0%	77.8%
Lynx	3	71.3%	75.6%
baseline		76.0%	76.0%

Table 4.2: Recall and precision of the Lynx parser in French

smoothing for any zero-weighted sentences, yielding much better results. Various weighting functions were tried, including multiplication by a factor that serves as a learning rate (which did not work well) and multiplication of the log score by a learning rate factor (which worked better), the latter being used for the reported results.

Having observed that unreasonably long tails on the distributions of many parameters of the model were having an adverse effect, not only on results, but also on the speed of parsing, we attenuated the model using a combination of maximum limits on the number of values for an attribute and maximum ratios between the most and least probable values. This reduction in model size had the immediate effect of speeding up parsing, thanks to reduced search space, and also improved parsing results by several percentage points.

In the experiments reported here, EM was applied first (with no attenuation) and then the resulting model was attenuated before being used to parse the test set. While the EM did yield a small improvement before attenuation, that gain was hidden by the larger improvement given by model attenuation, which yielded bigger gains in the no-EM case than after one or two rounds of EM. Thus, the best models in these experiments were those that did not use EM.

4.4.3 The Experiments

As a point of comparison for all these experiments, we generated a baseline score by constructing a right-branching parse of each sentence, taking punctuation into account. This simple method gives quite good results for many languages, thus providing a good basis of comparison for other methods. We also compared the reported results against a few other potential baselines, such as random bracketing or left-branching bracketing, but these were consistently outperformed by the right-branching baseline. Since we assume that natural-language syntax is struc-

	EM	recall	precision
Lynx	0	**40.6%**	**47.8%**
Lynx	1	38.6%	47.1%
Lynx	2	37.5%	45.2%
Lynx	3	37.6%	45.3%
baseline		34.2%	38.9%

Table 4.3: Recall and precision of the Lynx parser in Chinese

tured like a tree, we would expect, for most languages, that either a left-branching or right-branching a baseline would do reasonably well, depending on how flexible the word order is for that language. For the languages we tested (English, French, and Chinese) the right-branching method worked best as a baseline.

For our first experiment, we used 29 hand-bracketed French sentences from section 100 of the Canadian Hansard corpus to test a French parser built automatically from sections 0 through 99 of the same corpus. The training data consisted of bilingual sentence pairs, where the English version of each sentence was parsed using the Link parser. The resulting French parser achieved a recall of 76.4% and precision of 80.5%, slightly better than the right-branching baseline. We later did another French experiment with 344 sentences from the Paris 7 treebank (produced by the LLF laboratory of the University of Paris), and found lower scores, but a much more significant difference between the baseline and our automatically-induced parser. In this case, recall was 53.6% and precision was 55.8%.

For our second experiment, we used 265 Chinese sentences from the Xinhua Treebank to test a Chinese parser built automatically from other sentences from Xinhua News. The training data consisted of bilingual sentence pairs, where the English version of each sentence was parsed using the Link parser. The resulting Chinese parser achieved a recall of 40.6% and precision of 47.8%, significantly better than the right-branching baseline.

A summary of the results above is shown in Table 4.4, which compares the Lynx bracketing score in each language against the corresponding right-branching baseline score. Unsurprisingly, the English version of the parser is most able to exceed the performance of its baseline. Although the Chinese version does not achieve a very impressive score, it does make a significant improvement over its baseline. The first French test (number 1 in the table), on the other hand, is almost the reverse of the Chinese: While failing to improve very much over its baseline,

	baseline	Lynx	improvement
English	68.2%	92.3%	36%
Chinese	36.4%	43.9%	21%
French (1)	76.0%	78.4%	3%
French (2)	40.3%	54.7%	36%

Table 4.4: A cross-linguistic comparison of the Lynx parser vs. the right-branching baseline, by bracketing F_1 score, with the margin by which Lynx does better than the baseline.

it does quite well overall, with high precision and recall. The French baseline scores higher than the English baseline, perhaps because the French sentences are more consistently right-branching than English or Chinese, or perhaps because the hand-annotator favored right-branching interpretations of the sentences. However, a second test of the French Lynx parser on the more difficult Paris 7 test set (number 2 in the table) shows a much more significant improvement over the right-branching baseline.

4.4.4 Improving the French Parser

Upon closer examination of the French Lynx parser's output, we discovered several common constructions that were not handled correctly. In order to show how these problems can be addressed, we focus here on just one common construction in French, *ne ... pas*, which corresponds to the English *not*. This is tricky for LinkSet to handle because a non-contiguous two-word French phrase needs to be aligned to a single English word, which is not possible in the LinkSet model as presented above. The best we can hope for in the standard configuration of LinkSet is to have *not* align to either *ne* or *pas*, thus connecting it to its antecedent with an **N** link, and a null link connect the other. If this happens throughout its training data, and both *ne* and *pas* are seen with **N** links, the French Lynx parser will then be equipped to correctly link both of them with **N** links to the verb they modify. However, in practice we found that this was not happening.

Looking back at the training data, we found that while *not* was often aligned with *ne* (46%) or *pas* (23%), the other word was often not null linked, but aligned to some common word, such as *the*. This is not too surprising, since the automatically trained word-translation model tends to assign a substantial probability to the alignment of any common English word to and common French word.

Having diagnosed the problem, we proceeded to attempt a quick fix. Using a list of 47 English words with meaning similar to *not*, we purged the word-translation model of any probability of aligning *ne* or *pas* to any word not in the list. Then we ran the LinkSet bilingual parser on the original bilingual corpus, using this restricted model, and generated training data for an improved French Lynx parser.

This simple change had the desired effect of forcing every instance of *ne* or *pas* in the training data to either align to *not* or a synonym, or ne null linked. As we had hoped, the resulting French Lynx parser was now able to correctly use the link **N** when connecting both *ne* and *pas*. However, a problem still remained: *ne* was very often linked to the word on its left, rather than the verb on its right, to which it should have been linked. This can be easily explained. By default, null links generated in LinkSet connect an unaligned word to the word on its left.

While they work quite well when a contiguous French phrase is aligned to a single English word, null links are not really will-equipped to handle discontiguous French phrases. A better approach would be to allow both French words to align to the same English word. An equivalent notion would be to allow the English word to "duplicate" itself, so that each copy could align to one of the corresponding French words. When the English word being duplicated is connected to only one other word in its English structure (provided as input to LinkSet), this operation carries no risk of causing a cycle or violating the no-crossing constraint on links. Thus we propose dealing with this problem by adding optional duplication of leaf nodes to the transformations carried out by the LinkSet parser. Viewing the LinkSet model as an alignment model, we can conceptualize this cloning operation as a fertility model, where the probability of duplicating an English word is the probability of increasing its fertility, allowing it to align to more than one foreign word. To test this idea without a complete redesign of LinkSet, we can simulate the operation by duplicating the word *not* in the training data for every English sentence in which it's French counterpart contains *ne...pas*. A simple script can accomplish this. Then we run LinkSet on the modified training set and build a new French Lynx parsing model.

Voilà! The new monolingual French Lynx parser now correctly analyzes the *ne...pas* construction the majority of the time, with both words connected to the verb they modify with **N** links.

Figure 4.2 shows the effect of these changes on one fairly typical sentence, which in English means: "I am pointing out that we cannot continue to rip off the Canadian people." First it shows the analysis generated by the original French Lynx parser. Then, in (b), it shows the results of retraining the parser with its word

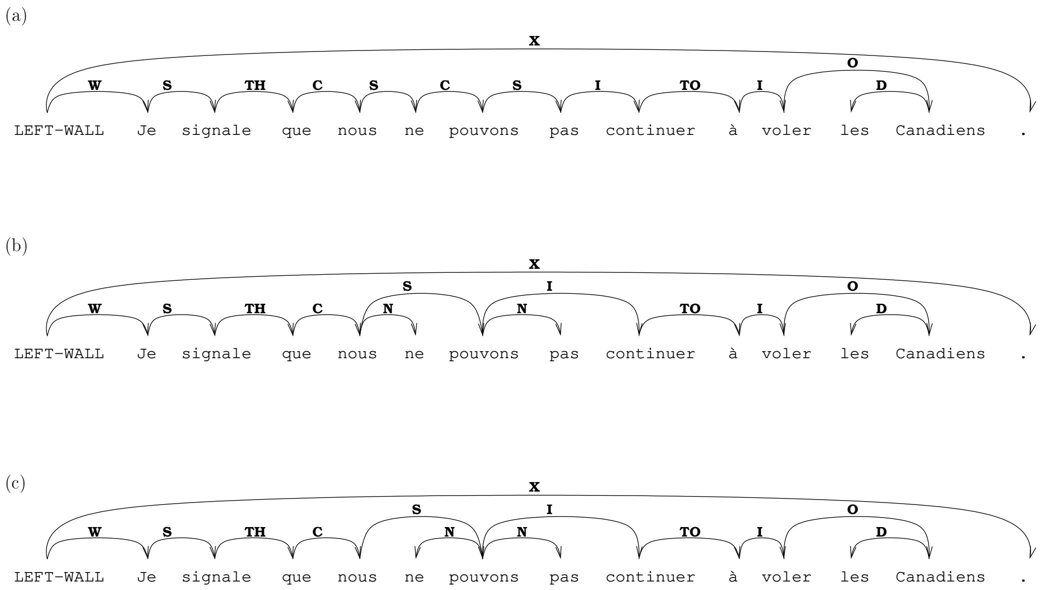

Figure 4.2: A single French sentence, parsed by the French Lynx parser at various stages of the improvements inspired by the phrase *ne...pas*: (a) the original French Lynx parser, (b) the parser retrained using a restricted word translation model, and (c) the parser retrained using duplication in addition to the restricted word translation model.

C	connects subordinating conjunctions and certain verbs and adjectives with subjects of clauses.
D	connects determiners to nouns.
I	connects certain verbs with infinitives.
N	connects the word “not” to preceding auxiliaries and modals.
O	connects transitive verbs to direct or indirect objects.
S	connects subject-nouns to finite verbs.
TH	connects words that take “that [clause]” complements with the word “that”.
TO	connects verbs and adjectives which take infinitival complements to the word “to”.
W	is used to attach main clauses to the *left wall*.
X	is used to connect punctuation symbols to words.

Table 4.5: The meanings of connector labels used in Figure 4.2.

	100 sentences		18 negative sentences	
	recall	precision	recall	precision
baseline	82.0%	82.0%	89.3%	89.3%
original	79.0%	82.9%	83.5%	85.1%
word translation fix	77.4%	84.1%	70.9%	88.0%
word fix + duplication	78.9%	84.6%	81.6%	96.6%

Table 4.6: A comparison of bracketing precision and recall for the French Lynx parser at various stages of the improvements inspired by the phrase *ne...pas*, evaluated both on a set of 100 sentences, and on the subset of those 100 that contain the phrase.

translation model restricted so that *ne* and *pas* cannot be aligned to anything but *not* or one of its synonyms. Third, in (c), it shows the same sentence analyzed using a parser trained using both the word translation restriction and the ability to duplicate leaf nodes in the English structures used for training. The first analysis is quite wrong, and does not recognize *ne* and *pas* as modifiers of the modal verb *pouvons*. The second analysis is much better, correctly pulling out *ne* and *pas* as negative words (indicated by the **N** link labels), but it incorrectly connects *ne* to *nous* as if that were the word being negated, rather than the verb. The third attempt, however, solves this problem as well, and achieves a fully correct analysis.

In addition to looking carefully at the output of our improved French Lynx parser, we evaluated it for bracketing precision and recall on a set of 100 hand-bracketed sentences at each step of the process, comparing the results to a baseline. We also performed the same tests on a selection of sentences for which our changes are particularly relevant; that is, we selected all 18 sentences, from among the 100, that contain the words *ne* and *pas*. The results are shown in Table 4.6. According to this metric, our changes produced a significant increase in precision, but a slight drop in recall.

This exercise shows some relatively simple modifications solving a significant problem in an induced parser's ability. This suggests that not only can a quick-and-sloppy parser be whipped together for a new language with little additional effort (once the LinkSet system is provided), but that such a parser can be incrementally improved with a little bit of analysis and simple intervention in the training process. It seems likely that many similar problems in a rough parser could be addressed in a similar way, adding up to significant improvement with a reasonable amount of ongoing effort.

4.5 Discussion

To our knowledge this is the first demonstration of a monolingual parser being induced across a bilingual corpus. (As mentioned above, others have induced more basic text analysis tools.) No knowledge was used except for the sentence-alignment of the corpus and the original English parser; given these resources, the building of a foreign-language parser was completely unsupervised.

In this unsupervised-learning setting, it came as somewhat of a surprise that EM training was unable to improve the quality of the induced parsers beyond their original levels. In some sense it could be considered a relief, considering the huge computational cost involved. On the down-side, this could indicate a problem with the model. However, it is likely that the problem lies instead with the way EM was applied. Our EM training was a fairly crude approximation of the ideal parameter estimation method, with sentence-level rather than feature-level granularity, so a better EM technique might work better. To address the glaring problem of sentence-level granularity, the EM training should retrain the model based on the scores assigned to each part of each sentence, rather than based on each entire sentence. A related improvement would be to use the entire lattice generated by the LinkSet bilingual parser for each sentence, rather than just the most probable sentence found within it. While this would probably increase computational and memory requirements, a cleverly integrated system could probably accomplish this kind of training without massively increasing the (already quite high) processing burden for training. Such a system would probably give better results; whether they would be significantly better is hard to say without having time to actually do the experiment.

The difficulty of comparing the quality of parsers for different languages should be obvious: since the language is different, it is impossible to use the same test set in each language for a direct comparison. Furthermore, limited resources may necessitate the use of test sets of differing domain, size, and quality. Even given a reasonably comparable set of test sentences, the canonical analyses available for them may differ in style across languages. Yet another difficulty is introduced by incompatibility between the form of available test data and the form used by the system under evaluation. We were able to partially overcome this last hurdle by designing a modified metric for comparison of a linkage and a constituent bracketing.

Even with all these caveats, the bracketing scores should tell us something useful, though perhaps not with great precision. Both the French and Chinese Lynx parsers, while unsurprisingly unable to reach the high quality of the English version unencumbered by a language barrier, achieve substantial success in surmising the syntactic structure of their respective languages. The quantitative comparison confirms what a less systematic look at the results seemed to say. In many cases, the induced parsers are able to analyze foreign-language sentences with complete or substantial correctness, but significant problems remain, both for unusual constructions not sufficiently represented in the training data, and for structures systematically misanalyzed due to mismatches in structure between the two languages that were too difficult for the LinkSet model to align properly.

Our case study on the French negation phrase (*ne...pas*) focused on one such mismatch. In the original formulation, the LinkSet model was unable to account for this phrase properly, because of its one-to-one alignment model. While the LinkSet model can, through the use of null links, deal reasonably well with contiguous multi-word phrases aligned to single words in the other language, a non-contiguous phrase presented a problem. Because the LinkSet parser was unable to correctly align this phrase, part of the phrase was incorrectly aligned each time it appeared in the training data, resulting in an incorrect model of the phrase in the French Lynx parser. However, we were able to address this problem fairly easily by narrowing the overly broad word-translation model for those specific words and by allowing some English words to effectively align to more than one foreign word.

The problem of discontiguous many-to-one alignment is quite general and may be at the root of many misaligned phrases. Therefore this case study is not just about one difficult phrase, but about a whole class of mappings between languages that many alignment techniques are unable to handle. While many translation systems use phrase-to-phrase alignment, these approaches typically deal only with contiguous phrases. As our experiment shows, a simple extension to the LinkSet model can enable it to handle discontiguous phrases. (While a relatively straightforward extension of the LinkSet model, it is still significant enough a change to be implemented with a quick fix to the existing code base; hence the narrowly focused experiment.) Making this modification to LinkSet in general would probably be a significant step forward, but undoubtedly some tricky mismatches would remain. However, our case study suggests that for any particular language pair, progress could quickly be made from a quick and dirty induced parser to a higher-quality parser through focused incremental improvements such as the one we have demonstrated.

Chapter 5

Reranking Translation Hypotheses with a Bilingual Parsing Model

5.1 Motivation

Automatic translation of foreign languages is a goal that hardly needs motivation. Not only would high-quality automatic translation open up to anyone a vast storehouse of worldwide information resources, it could enable collaboration and understanding between people who were previously unable to communicate, and make travel much easier. Although some might say such a technology could bring about a new era of global peace and prosperity, others realize that, while such a tool would indeed be useful, there is more to getting along than communication, and more to communication than information. Still others might complain about the inevitable misunderstandings that will be promoted by such a technology, which encourages people to assume they can easily communicate with those of a different culture without taking the effort to learn to understand where the others are coming from by learning their language and culture. Nevertheless, it is widely assumed that there is hope of making automatic translation work well enough to be very useful to many people.

There have been two main schools of thought in automatic translation work: theory-based approaches, and corpus-based approaches. The latter have been the more successful at producing practical solutions in the past two decades, even when using relatively simple statistical models that clearly ignore much of what we know about how language works. However, the simplicity of these models, which is important in making them trainable on a reasonable-size corpus and in a reasonable amount of time, also limits their ability to properly capture and transform the form and meaning of the languages they model. For example, N-gram language models are unable to ensure that their output is grammatical because they cannot model the relationships between words that are separated by other phrases, which can have arbitrary length. IBM-style statistical translation models are unable to transform the structure of a sentence properly when translating it, looking only at the words and their approximate positions rather than the structural relationships between them, and could easily produce a translation that means the opposite, or has no discernible meaning. Looking at the output of today's state-of-the-art statistical translation systems, the translations are obviously flawed; they often do not look like grammatical English sentences at all, even when the correct meaning can be inferred from the resulting jumble.

In order to address this situation, more complex models are needed, models that can capture the structure of a sentence in a way that properly transmits the meaning. These new models are a sort of hybrid, incorporating increasing amounts of theoretical linguistic knowledge while remaining corpus-driven statistical models. By using the Lynx model of English syntax and the LinkSet model of the structural relationship between English syntax and the syntax of a foreign language, we hope to equip automatic translation systems to generate grammatical English translations with the correct structural relationship to the original (and thus the correct meaning).

In order to economically test the efficacy of the syntax-based Lynx and LinkSet models in a translation system, we use the models to assign scores to the translation hypotheses generated by a current statistical translation system. Using these scores, we can rerank the hypotheses and select the best one according to the new models. This method allows various models to be compared on the translation task without requiring a whole new translation system to be built each time. However, it does have one major drawback: when the new models are used to rerank the hypotheses generated by another system, they are limited to selecting the best translation generated by that system, even if the new model would have assigned a higher score to a better translation never generated by the base system. Thus, a reranking approach will tend to underestimate the potential of a new model because its range of outputs is artificially constrained. On the other hand, a reranking experiment is a good first step in evaluating new models for translation, because of its relatively low cost. If a model will not work well, we would want to discover that fact before investing months of effort in building a new translation decoder. If a reranking experiment shows that a new model has good potential, the effort of building a decoder that can take advantage of its strengths may then be justified.

5.2 Related Work

5.2.1 Background: Statistical Translation

Early attempts at machine translation used linguistically-informed rule-based approaches, but in the 1980's statistical approaches emerged, enabling translation models to be learned automatically from parallel bilingual corpora. A group at IBM proposed a statistical approach to translating from French to English [4]. Based on concepts from information theory, the approach imagines that every French sentence is generated from an English sentence, so the task of translation from French to English is the task of recovering the original English sentence. According to Bayes' theorem, finding the most probable English sentence to have generated a French sentence is the same as maximizing the product of the probabilities of the English sentence alone and of the French sentence given the English sentence, which yields what they call the Fundamental Equation of Machine Translation:

$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e})$$

The English language model probability $\Pr(\mathbf{e})$ can exist independently of any translation work, and the n-gram language modeling technique is known well enough that no detail need be given here. To estimate the parameters of the translation model $\Pr(\mathbf{f}|\mathbf{e})$, Brown (et al.) use the bilingual corpus that comes from the records of the Canadian Parliament, which are kept in both French and English. They proposed five generative translation models, all of which involve word-level alignments between corresponding English and French sentences and are trained using an EM (expectation maximization) algorithm [5]. The training corpus is already aligned so that corresponding English and French sentences are matched together, but a major part of the parameter estimation is finding word-level alignments, which connect individual words in an English sentence to individual words in the corresponding French sentence. Each model generates a French string and a set of connections between English words and French words. The probability of a French string given an English string according to a model is then the sum over all alignments (all sets of connections) of the probability of the French sentence and the alignment given the English sentence.

$$\Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} \Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

The probability $\Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$ of a French string and alignment to a given English string is a product, over all the words in the French string, of the probability of its connection a_j given all previous connections a_1^{j-1}, all previous words f_1^{j-1}, the length m of the French string, and the English string $\mathbf{e}$, times the probability of the French word f_j given all connections up to the present a_1^j, all previous words f_1^{j-1}, the length m of the French string, and the English string $\mathbf{e}$.

$$\Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) \quad = \quad \Pr(m|\mathbf{e}) \prod_{j=1}^{m} \Pr(a_j|a_1^{j-1}, f_1^{j-1}, m, \mathbf{e}) \Pr(f_j|a_1^j, f_1^{j-1}, m, \mathbf{e})$$

In Models 1 and 2, a length m for the French string is generated first, and then for each position in that string a French word is generated along with how to connect it to the English string. Model 1 naïvely considers all connections equally likely, thus ignoring word order. Model 2 adds a dependency on string length and word position for the probability of a connection. These simple models do not give very accurate translation results, but they can be trained faster than more sophisticated models, and their output can be used as input to later models, thus improving both the speed and accuracy of parameter estimation overall.

In Models 3, 4, and 5, the French string is generated as follows: For each English word, choose how many French words to connect to it, then generate those words, and then choose positions for those words. In Model 3, the probability of a connection depends on string length and word position, as in Model 2. Model 4 adds a dependency on the actual words (French and English) being connected and on the positions of any other French words connected to the same English word. Model 5 is much like Model 4, except that unlike Models 3 and 4 it does not waste any of its probability on possibilities that are not valid French strings.

Candide, a system based on these models, demonstrated French-to-English translation with results comparable to those of Systran, a major rule-based translation system [2].

5.2.2 Parsing for Language Modeling

In our experiments on translation hypothesis reranking, the Lynx parser essentially acts as a syntax-savvy language model, assigning a fluency score to each hypothesis based on a lexicalized structural model of the English language. Several others have devised language models that take into account syntactic structure. Some of these, closely related to link grammar, are described in Section

2.2 above, such as the IBM long-range trigram model, and Chelba's dependency language model [23, 8]. Other language models that incorporate parsing include: Charniak's immediate-head parser [7], which demonstrates improvements over two strict left-to-right parsers without the need for any trilexical statistics; Roark's broad-coverage PCFG parser [24]; and the dependency language model of Jianfeng and Suzuki [15], which uses EM to learn a bilexical dependency model from unlabeled data, rather than requiring a preexisting grammar.

5.2.3 Reranking Hypothesis Lists

Similarly, Zechner's 1997 application of a chunk parser to spontaneous speech understanding made use of syntactic information in language modeling, using chunk scores to rerank hypotheses from the speech recognizer [31]. This N-best list reranking slightly decreased the word error rate of the speech recognizer, hopefully selecting more meaningful hypotheses for further parsing in order to improve the amount of useful information that could be gathered from the input.

Applying hypothesis-reranking to the parsing domain, Collins in 2000 developed a technique for efficiently using arbitrary features for high-quality parsing [10]. Using a simple parser to get a set of candidate parses, this approach then uses a second parsing model to rerank the hypotheses. Since reranking is done on complete parse structures, the second model can use arbitrary features which might be difficult to incorporate into an ordinary parser. Collins presents two discriminative reranking techniques, which could also be applied to domains such as translation.

Most recently, a summer 2003 workshop titled *Syntax for Statistical Machine Translation*, hosted by Johns Hopkins' Center for Language and Speech Processing [14] attacked the difficult problem of hypothesis reranking for translation using syntax, the same application we address in this chapter. They used the same evaluation set of 993 Chinese sentences with four reference translations. Some of their conclusions were fairly pessimistic. Oracle experiments showed that even a system that could select the best from 1000 hypotheses (quite unlikely!) would not have very good translation quality. Although a decent improvement was achieved, the majority of it could be accounted for by IBM Model 1, a very simple translation model with no syntactic knowledge. Several attempts to use grammar structure to score sentences gave results worse than the baseline. While progress was made, these results underscore the difficulty of this problem.

5.3 Model

Combining the model of monolingual sentence structure $\Pr(\mathcal{E})$ described in Chapter 2, and the model $\Pr(\mathcal{F}|\mathcal{E})$ of a structured foreign sentence given a structured English sentence described in Chapter 3, we can derive a model of a foreign sentence string given an English sentence string.

$$\begin{aligned}
\Pr(\mathbf{f}|\mathbf{e}) &= \sum_{\mathcal{E}} \Pr(\mathcal{E}|\mathbf{e}) \Pr(\mathbf{f}|\mathcal{E}) \\
&= \sum_{\mathcal{E}} \Pr(\mathcal{E}|\mathbf{e}) \sum_{\mathcal{F}} \Pr(\mathcal{F}|\mathcal{E}) \Pr(\mathbf{f}|\mathcal{F}) \\
&= \sum_{\mathcal{E}} \Pr(\mathcal{E}|\mathbf{e}) \sum_{\mathcal{F}:\textsc{words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E})
\end{aligned}$$

Here we have defined the probability of a foreign sentence given an English sentence to be the sum over all possible English structures of the probability of that structure given the English sentence times the sum over all foreign structures consistent with the given foreign sentence of the probability of the foreign structure given the English structure. Next we use Bayes' rule to break down the probability of an English structure given an English sentence string:

$$\begin{aligned}
\Pr(\mathcal{E}|\mathbf{e}) &= \frac{\Pr(\mathcal{E}) \Pr(\mathbf{e}|\mathcal{E})}{\sum_{\mathcal{E}'} \Pr(\mathcal{E}') \Pr(\mathbf{e}|\mathcal{E}')} \\
&= \frac{\Pr(\mathcal{E})\delta(\textsc{words}(\mathcal{E}) = \mathbf{e})}{\sum_{\mathcal{E}'} \Pr(\mathcal{E}')\delta(\textsc{words}(\mathcal{E}') = \mathbf{e})}
\end{aligned}$$

The delta function is 1 if it's condition is true, and 0 otherwise, so here it simply has the effect of requiring the structure to match the string. When we plug this expression back into the expression for $\Pr(\mathbf{f}|\mathbf{e})$, we get the following:

$$\Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathcal{E}:\textsc{words}(\mathcal{E})=\mathbf{e}} \frac{\Pr(\mathcal{E})}{\sum_{\mathcal{E}':\textsc{words}(\mathcal{E}')=\mathbf{e}} \Pr(\mathcal{E}')} \sum_{\mathcal{F}:\textsc{words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E})$$

This model can then (in theory) be used for translation, by finding the English sentence string most likely to have produced a given foreign sentence string.

$$\begin{aligned}
\hat{\mathbf{e}}_{\mathbf{f}} &= \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{f}|\mathbf{e}) \\
&= \operatorname*{argmax}_{\mathbf{e}} \sum_{\mathcal{E}:\textsc{Words}(\mathcal{E})=\mathbf{e}} \Pr(\mathcal{E}) \sum_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E})
\end{aligned}$$

As in the case of building a monolingual model of foreign sentence structure from a structure translation model and a model of English structure, it is impractical to build this model by iterating over all possible English and foreign structures.

The problem of reranking translation hypotheses is computationally easier than that of translation, because it is necessary only to assign a score to each translation, rather than searching over all possible translations to find the best. However, even when both the English and foreign sentence strings are given, the huge search space of all possible structures on both the English and foreign side can be prohibitive.

One way to approximate this is to replace a sum with a maximum. First search for the best English structure $\mathcal{E}$ consistent with the given English string $\mathbf{e}$, and then find either the probability $\Pr(\mathcal{F})$ of the best foreign structure or the sum of probabilities of foreign structures given the English structure $\mathcal{E}$.

$$\begin{aligned}
\hat{\mathbf{e}}_{\mathbf{f}} &= \operatorname*{argmax}_{\mathbf{e}} \sum_{\mathcal{E}:\textsc{Words}(\mathcal{E})=\mathbf{e}} \Pr(\mathcal{E}) \sum_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E}) \\
&\approx \operatorname*{argmax}_{\mathbf{e}} \max_{\mathcal{E}:\textsc{Words}(\mathcal{E})=\mathbf{e}} \Pr(\mathcal{E}) \sum_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E}) \\
&\approx \operatorname*{argmax}_{\mathbf{e}} \max_{\mathcal{E}:\textsc{Words}(\mathcal{E})=\mathbf{e}} \Pr(\mathcal{E}) \max_{\mathcal{F}:\textsc{Words}(\mathcal{F})=\mathbf{f}} \Pr(\mathcal{F}|\mathcal{E})
\end{aligned}$$

The output of the Lynx parser is the maximum $\Pr(\mathcal{E})$ (along with $\mathcal{E}$), while the output of the LinkSet bilingual parser is the maximum $\Pr(\mathcal{F}|\mathcal{E})$. Therefore, by composing them we can directly compute the expression to be maximized for each hypothesis $\mathbf{e}$, and find $\hat{\mathbf{e}}_{\mathbf{f}}$.

5.4 Search

Our translation system, which uses the Lynx and LinkSet models to rerank hypotheses, works as follows:

1. Begin with a foreign sentence needing translation into English.
2. Get an N-best list from some other translation system.
3. Rescore each translation hypothesis as follows:
   (a) Parse the candidate using the Lynx parser, obtaining the most probable structure and its probability.
   (b) Use LinkSet to find the probability of aligning the parsed candidate with the foreign sentence.
   (c) Combine the new scores with those from the translation system.
4. Rank the hypothesis list by the interpolated scores. The top-ranked candidate is now output as the English translation of the foreign sentence.

At this level of abstraction, this method is quite simple: use two existing systems to find scores for each hypothesis, combine these with the original decoder score, and select the top hypothesis. In practice, of course, it is more complex. First, an appropriate training corpus must be selected and used to train the Lynx parser, which involves parsing the English side of the training corpus with the Link parser. Then, the LinkSet model must be trained, which includes building a word-translation model from the sentence-aligned training corpus, and running LinkSet on the parsed training set. However, this last step can be omitted if the EM training of LinkSet's parameters is skipped. In the alignment experiments in the previous chapter, this training was not shown to improve results, so the building of a word-translation model may be sufficient to train LinkSet, at a lower computational cost. Once these models are trained, it is conceptually simple to run the parsers and score each sentence, although this may still incur a significant computational cost.

5.5 Results

To test this approach, we trained the Lynx and LinkSet parsers using a Chinese–English corpus, and used an existing statistical translation system to generate 100-best lists of English translations for another corpus of Chinese sentences, for each of which four reference translations were available. We reranked the translation hypotheses using scores from Lynx and LinkSet, and evaluated the improvement in translation quality using a variety of measures.

The test corpus, which was used to evaluate competing translation systems within the TIDES project, consists of 993 Chinese sentences from *Xinhua News*, each with 4 English references translated by different human experts.

Two different training corpora were used for different parts of the system. To train the Lynx parser, 343k English sentences from *The Wall Street Journal* and 3540 English sentences from *Xinhua News* were parsed using the Link parser. To train the word-translation model used by the LinkSet bilingual parser, we used a large bilingual corpus of 282k sentences from a variety of sources. We were careful to avoid using any sentences from the test set for training.

Parsing the 347k English training sentences with the Link parser and building the word-translation model based on 282k sentence pairs took about 2 days on a machine with eight 700MHz Intel processors.

For each of the 993 test sentences, a list of 100 English translation hypotheses was generated using a decoder built at Carnegie Mellon within the RADD-MT group. (Credit belongs to Stephan Vogel and Bing Zhao for generating these translation hypotheses.) Each hypothesis was accompanied by a score assigned by the decoder.

Using the Lynx parser, we parsed each hypothesis to find its most probable structure and the probability of that structure. This was the most computationally expensive step in the process. While the Lynx parser is reasonably fast on short sentences, its worst-case performance scales as the cube of the sentence length, with a fairly high constant factor. Compromises in search quality can improve the speed, but even so, extremely long sentences could not be parsed at all with reasonable time and space constraints, and long sentences that were parsed had results of declining quality. Parsing 99300 hypotheses took over a week on multiple machines, about 260 hours total processor time, which is an average of under 10 seconds per hypothesis. This speed was achieved only at the expense of a relatively lax beam ratio, combined with a time-out preventing any one hypothesis from taking more than ten minutes. This necessary compromise and constraint cause a decrease in search quality. Without the time-out, the longest sentences would probably have taken hours each to parse, and even in this case there was a very wide range of times, from a fraction of a second for short sentences, to the ten-minute maximum on long ones. Memory requirements for this parsing was also relatively high at several hundred megabytes per process, which meant that even on multi-processor machines, only one or two could run at a time. When this stage in the experiment was completed, we had a structure for each English translation hypothesis and a probability score for each structure.

As the final step, we used the LinkSet bilingual parser to align each structured

length	D	D+LS	D+L	D+L+LS	L	L+LS
0-9	**27.55**	29.12	29.55	29.05	29.48	29.02
10-19	36.52	29.64	28.89	**28.81**	28.89	**28.80**
20-29	38.67	31.94	28.87	**28.63**	28.87	**28.63**
30-39	40.43	38.78	**27.89**	27.94	**27.89**	27.94
40-49	40.63	39.80	**28.30**	28.33	**28.30**	28.33
50-59	41.23	41.23	**30.42**	**30.42**	**30.42**	**30.42**
60-69	42.35	42.35	**36.27**	**36.27**	**36.27**	**36.27**
70-79	42.05	42.05	42.05	42.05	42.05	42.05
80-89	36.64	36.64	36.64	36.64	36.64	36.64
90-99	46.24	46.24	46.24	46.24	46.24	46.24
all	37.93	33.83	28.92	**28.80**	28.91	**28.79**

Table 5.1: Average Kendall's-τ rank-distance between the NIST-score ranking and rankings generated by the Decoder, Decoder+LinkSet, Decoder+Lynx, Decoder+Lynx+LinkSet, Lynx, and Lynx+LinkSet, respectively, grouped by sentence length. The shortest (best) distances are highlighted for each length.

English hypothesis with its original Chinese version. To align 58200 hypotheses took 325 hours of processor time, an average of 20 seconds per hypothesis. The remaining 411 hypotheses could not be aligned for various reasons, the most common reason being that the current implementation of LinkSet can not handle sentences longer than 64 words. This problem could be remedied at an increased cost in space and processor usage, but again the longest sentences require a prohibitively long time to process. The sentences that could not be aligned were simply assigned a score of zero for alignment.

In order to evaluate the resulting hypothesis rankings, we scored each hypothesis using two standard measures of translation quality, the BLEU score and the NIST score. However, since standard BLEU scores include a geometric mean of various N-gram scores, they are unsuitable for evaluating individual sentences because they assign a score of zero to any sentence that does not contains at least one 4-gram in common with a reference. Therefore, we used a modified version of the BLEU score, which has an arithmetic mean rather than a geometric mean, and is much less prone to assign zero scores. To calculate the NIST and BLEU scores for each hypothesis, we used the four reference translations available for each sentence.

For each sentence, sorting the 100 translation hypotheses by score yields a

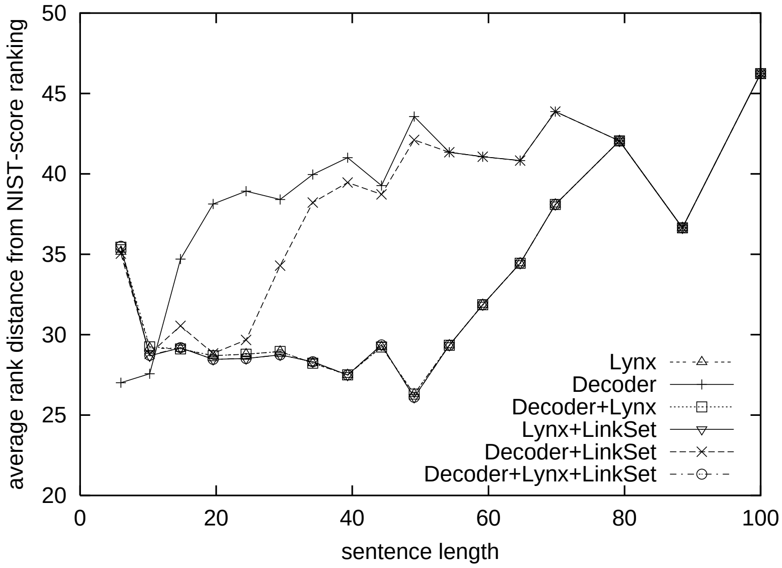

Figure 5.1: Average Kendall's-τ rank-distance between the NIST-score ranking and rankings generated by the various system combinations. Table 5.1 shows the data for this chart in detail.

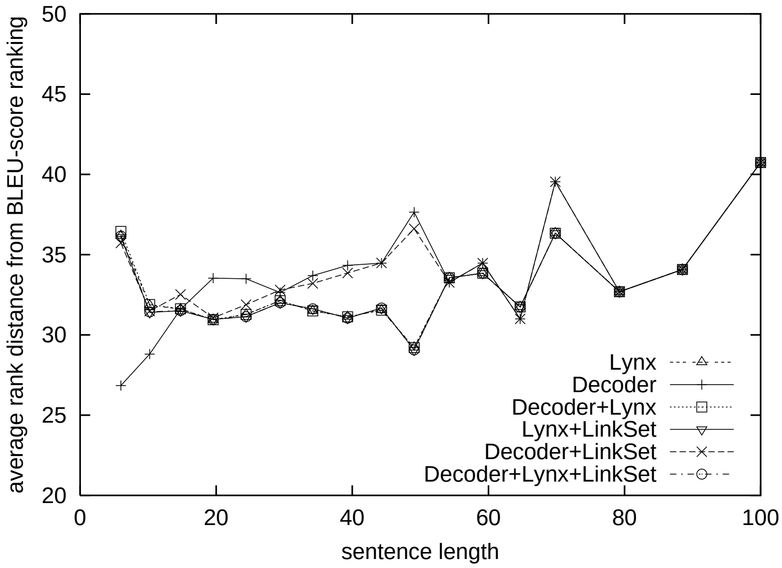

Figure 5.2: Average Kendall's-τ rank-distance between the BLEU-score ranking and rankings generated by the various system combinations.

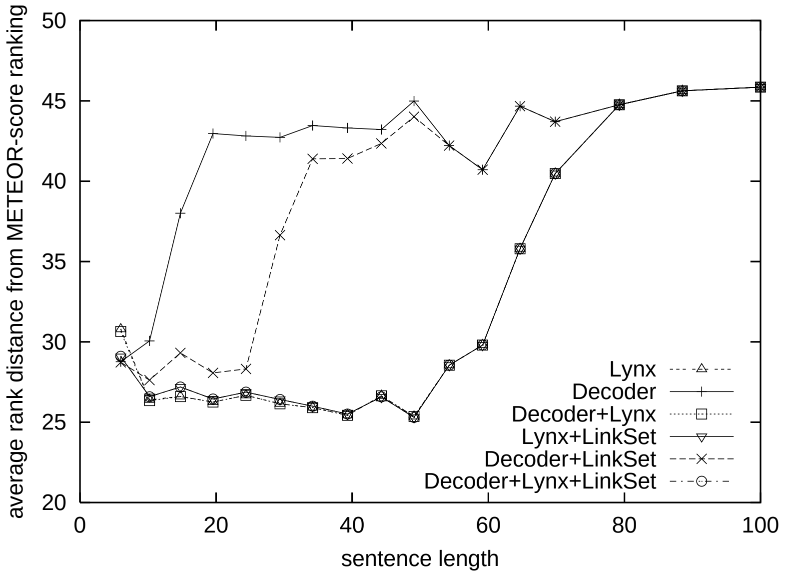

Figure 5.3: Average Kendall's-τ rank-distance between the METEOR-score ranking and rankings generated by the various system combinations.

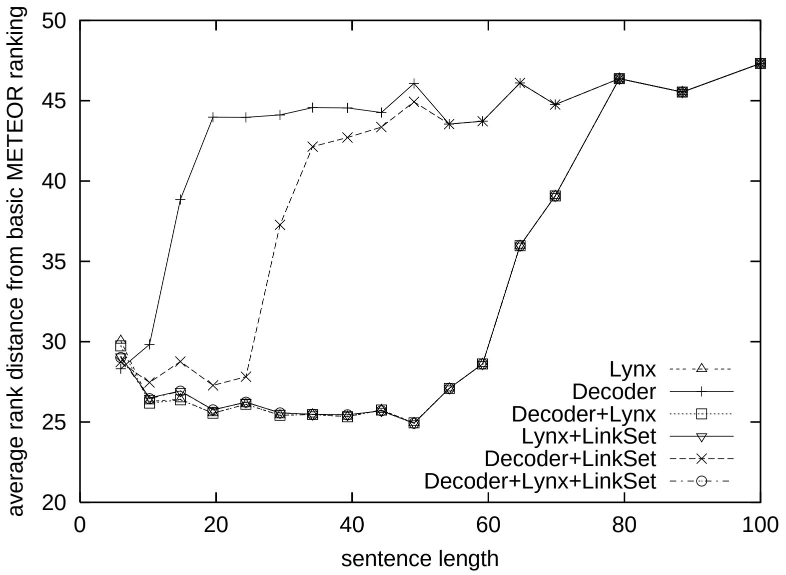

Figure 5.4: Average Kendall's-τ rank-distance between the basic METEOR-score ranking and rankings generated by the various system combinations.

hypothesis ranking. We ranked the hypotheses according to the original decoder score, the Lynx score, the LinkSet score, and by combinations of these scores. To evaluate these rankings, we also ranked the hypotheses according to their BLEU and NIST scores with respect to the reference translations.

To evaluate the system-generated rankings, we measured the rank-order distance between each system's ranking for each sentence and the corresponding rankings given by BLEU score and by NIST score. We used the Kendall rank correlation (τ) as a measure of agreement between rankings; this distance is the number of adjacent pairs that must be swapped in order to bring the rankings into agreement. Because the rankings are defined through sorting a list of scores, any sets of identical scores will yield partial orderings; this is not a problem for this distance metric. The longest possible distance for 100 hypotheses is $\sum_{i=1}^{99} i = 4950$, which would occur if the rankings being compared were in reverse order.

Figure 5.1 shows the average distance between the rankings of various systems and the NIST-score ranking, grouped by sentence length. The sentences were grouped by length into bins of width five. Six different system combinations were compared: the original decoder score, the Lynx score, both of these together, and each of these three with the LinkSet score. For these experiments, each system was weighted equally. Additional trials, in which varying weights were used for each system, gave almost identical results. In this graph, three distinct lines are visible, the top (worst) one representing the decoder alone, the next one representing the decoder plus LinkSet, and the bottom (best) line representing the rest of the system combinations, which all have very similar results. All of these best combinations include the Lynx score. This chart shows that among these systems, the Lynx score is the best approximation of the ranking given by the NIST scores, and that no combination of systems significantly outperforms the unadorned Lynx score. However, a look at the more detailed Table 5.1, which contains the same data, shows that, overall, the combination of all three systems does slightly better than the Lynx score alone. Meanwhile, the LinkSet score is able to bring a small improvement when added to the decoder score, but not as much as the Lynx score. For long sentences, especially those 80 words and longer, the lines converge. This happens because we were unable to get Lynx and LinkSet scores for the longest sentences due to the computational constraints mentioned above. The steady decline in quality as sentences get longer, clearly visible on the chart, is probably caused by a corresponding decline in Lynx parser accuracy caused by the combination of the increased complexity of parsing longer sentences and the effect of time-outs cutting short the search for the best parse according to the Lynx model.

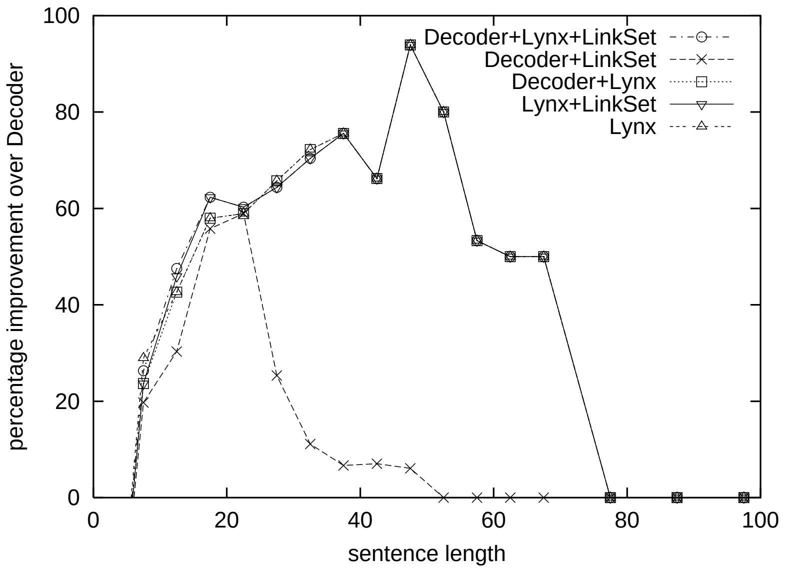

Figure 5.5: Percentage of sentences for which various systems achieve shorter (better) rank distances from the canonical ranking (according to NIST score) than the Decoder alone, grouped by sentence length.

The equivalent graph using BLEU scores is shown in Figure 5.2. While the difference in rank-distance between different systems is less pronounced with BLEU scores, it shares some important features with the NIST-score rank-distance comparison: the decoder alone is again worst, followed by the decoder plus LinkSet, and the combinations that include Lynx, being about equal to each other, have the shortest (best) distance from the optimal ranking according to BLEU scores.

A new translation evaluation metric under development at Carnegie Mellon, known as METEOR, explicitly deals with the relative positions of matching words and phrases in the translation hypothesis vs. the reference translations. The same rank-distance experiment was carried out using two variants of this metric, the full METEOR score and a more basic version, as shown in Figures 5.3 and 5.4, respectively. The results are similar to those of the NIST metric, but with an even more pronounced separation between the three major lines.

Figure 5.5 shows the proportion of sentences for which each system combination achieves a ranking closer to the NIST ranking than that of the original translation system. As in the previous chart, here every system incorporating Lynx

System	NIST	BLEU
Decoder	0.0%	0.0%
Decoder+LinkSet	28.0%	10.4%
Decoder+Lynx	59.1%	25.1%
Lynx	59.5%	24.9%
Lynx+LinkSet	59.9%	**26.3%**
Decoder+Lynx+LinkSet	**60.3%**	26.1%

Table 5.2: Percentage of sentences for which various systems achieve shorter (better) rank distances from the canonical ranking (according to NIST or BLEU scores, respectively) than the Decoder alone.

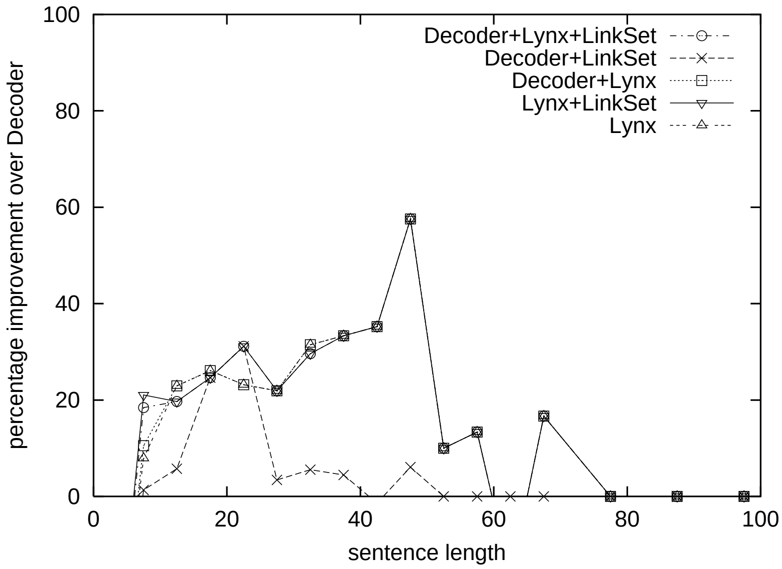

Figure 5.6: Percentage of sentences for which various systems achieve shorter (better) rank distances from the canonical ranking (according to BLEU score) than the Decoder alone, grouped by sentence length.

system	NIST	BLEU	sentence
ref			the construction of all these projects has been going on smoothly.
Lynx	3.1016	0.1488	conjectured average construction project proceeded smoothly .
Decoder	4.0897	0.1845	these engineering construction chong-ils proceeded smoothly .
ref			china to speed up construction of its talents market system
Lynx	10.2757	0.0714	china will speed talent market system construction
Decoder	6.0227	0.0714	china will accelerate talent market system building
ref			maclaren held talk with zou jiahua
Lynx	5.4181	0.1488	mccain crana jiahua mentenegro lengsavat held talks
Decoder	5.4181	0.1488	mccain crana year-on-year lynden zou held talks
ref			china and u.s. signed high-tech cooperation agreements in shanghai
Lynx	1.9568	0.0805	sino-us at shanghai-based high-tech japan-manchukuo cooperation agreement
Decoder	1.7388	0.0850	sino-us at shanghai-based japan-manchukuo hi-tech cooperation agreements
ref			tang jiaxuan replied as follows:
Lynx	3.9668	0.1071	tangxian tang is unscrupulously said the :
Decoder	5.5865	0.1429	jiaxuan is hears tang said the :
ref			now, let's first take a look at canada.
Lynx	5.5407	0.2394	sensing why we look canada first .
Decoder	5.5407	0.2394	eavesdrop ? we look canada first .
ref			the commercials will be televised early next year.
Lynx	2.6578	0.1387	126.6 will stay early next broadcast .
Decoder	2.6578	0.1387	advertisements will stay early next broadcast .
ref			reported by manila correspondent ferdinand
Lynx	3.6274	0.0714	manila defensives fee southern de newspaper reports
Decoder	3.6274	0.0714	manila fee-to-tax surgery yugoslav de newspaper reports
ref			correspondent liu zhenting reporting from bangkok
Lynx	5.2482	0.1071	bangkok baosheng liu zhuo tinghan newspaper reports
Decoder	5.2482	0.1071	bangkok surgery liu zhuo roman newspaper reports
ref			development of township enterprises in southeast fujian of china continues to take the lead
Lynx	1.3073	0.0975	china mawei-fuzhou southeastern township enterprises develop continued presentations
Decoder	1.6525	0.0948	china mawei-fuzhou southeastern township enterprise development presentations continue
ref			china's foreign capital utilization increased 27% as of november
Lynx	3.5693	0.1159	chinas 20-21 november using capital 15.639 rose 27%
Decoder	4.3576	0.1386	chinese 20-21 november using capital amounts 27% growth
ref			– good for the long-term prosperity and stability of hong kong's economy.
Lynx	4.2947	0.1994	hong shuo favorable economic prosperity long-term stability .
Decoder	2.9978	0.1289	hong pluralized favorable economic stability long-standing prosperity .
ref			china to continue the policy of opening up financial sector to the outside
Lynx	7.2309	0.3676	china will continue practiced outside financial opening policy
Decoder	4.6636	0.1920	china's will continue non-actionable outside financial opening-up policy
ref			the surplus from trade with japan was 1.7 billion us dollars.
Lynx	3.8873	0.1570	japan against trade surplus 17000 dollars dollars .
Decoder	2.6942	0.1191	china-japan imposed trade surplus 17000 dollars dollars .
ref			rapid development of foreign cooperation seen in china's construction material industry
Lynx	4.0760	0.0827	tung chinas ceramic industrial cooperation outside develops rapidly
Decoder	4.0760	0.0827	chinas nonferrous metallurgy industrial cooperation outside develops rapidly
ref			foreign-invested enterprises, a spotlight of growth in the national economy of shandong
Lynx	3.0519	0.1381	capital enterprises become shandong's national marked growth points
Decoder	2.5334	0.0975	foreign-funded enterprises become shandong's shanddong notable growth points
ref			china to use loans from world bank to build railroad communication network
Lynx	4.6301	0.1217	chinas use world 51.9 building railway network communications
Decoder	3.8767	0.1252	chinas using world discount building railway communication network

Table 5.3: The translation hypotheses ranked first by Lynx and by the decoder for sentences having 7-8 word translation hypotheses, with NIST and BLEU scores relative to four reference translations, one of which is shown.

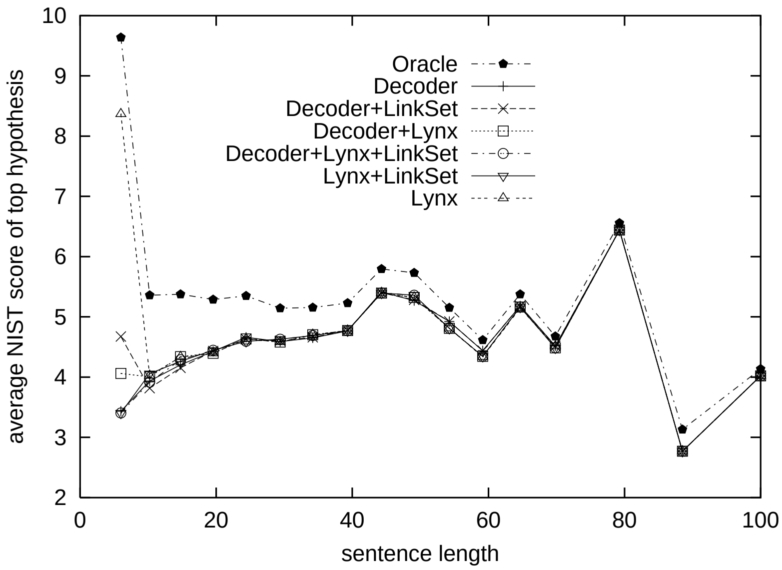

Figure 5.7: Average NIST score of top hypothesis for various systems, by sentence length. Higher scores are better.

scores has approximately the same curve, which is the highest (best) curve on the chart. An equivalent chart using BLEU scores is shown in Figure 5.6. While its percentages are lower, its basic shape is very similar to the NIST-score chart. In both charts, we see again that improvement due to Lynx drops off for very long sentences, which occurs because the results for long sentences were incomplete due to computational expense. It is interesting that moderately long sentences yield more improvement than shorter sentences; perhaps there is more room for improvement in the original decoder's output at these lengths. The same data are summarized on Table 5.2, where we can see clearly that all combinations that include Lynx yield better rankings on a majority of sentences according to NIST scores, and on over a quarter of sentences according to BLEU scores. For both metrics, the systems that include both Lynx and LinkSet do best.

While the figures above have shown a definite improvement over the original decoder in the ranking of translation hypotheses when the Lynx and LinkSet parsers are used for scoring, practical translation systems need to choose only one best translation for output. Improving the quality of the top-ranked hypothesis on a well-made translation system is quite difficult, and is the goal of many lines of research. When we limit ourselves to reranking hypotheses generated by a specific

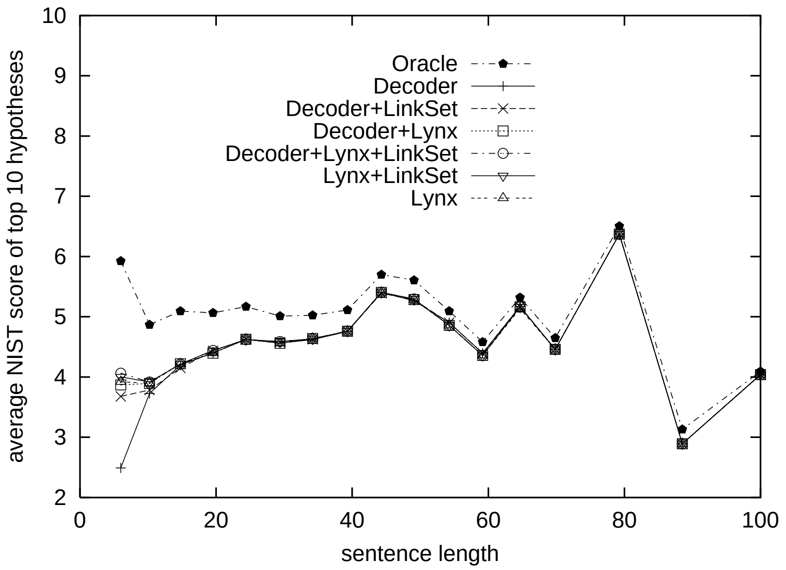

Figure 5.8: Average NIST score of top 10 hypotheses for various systems, by sentence length. Higher scores are better.

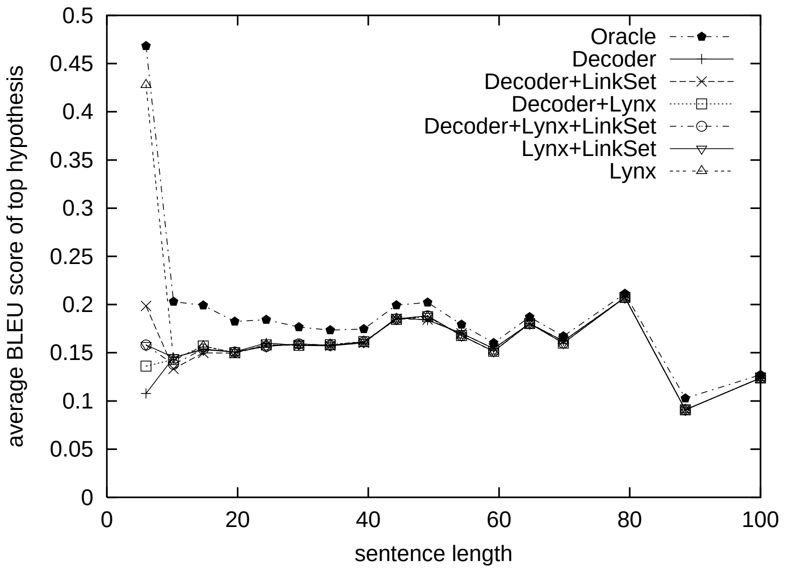

Figure 5.9: Average BLEU score of top hypothesis for various systems, by sentence length. Higher scores are better.

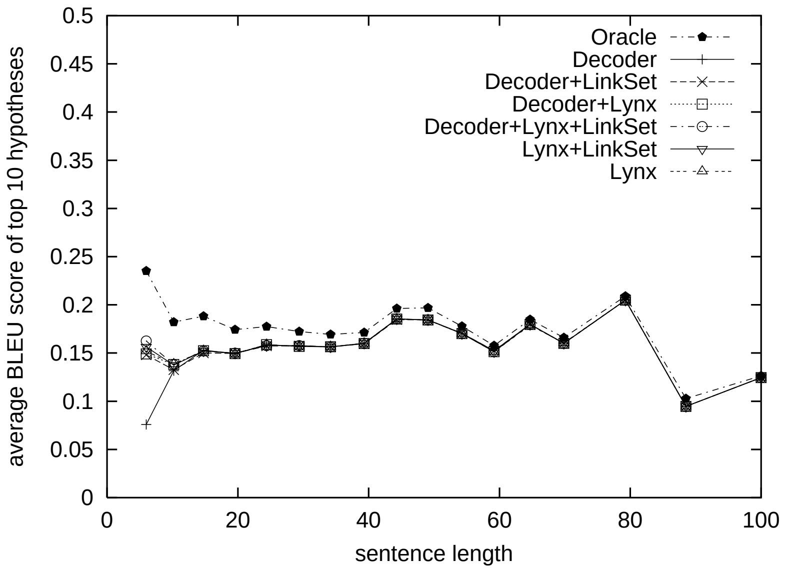

Figure 5.10: Average BLEU score of top 10 hypotheses for various systems, by sentence length. Higher scores are better.

decoder, we limit the possible improvement in translation quality by the quality of the best hypothesis generated. If all 100 hypotheses are equally bad, no amount of reranking will do any good.

The NIST score of the number-one-ranked hypothesis and the average NIST score of the top 10 hypotheses, as ranked by each system, are shown in Figures 5.7 and 5.8, respectively, averaged in groups of similar-length sentences. An Oracle score is also shown in each chart, indicating the best possible score, which would be received by a system that gave exactly the same ranking as the NIST scores do. In general, the six system combination score almost identically, with the exception of very short sentences. The closeness of the Oracle score to the other systems shows that there is not much room for improvement through reranking. Indeed, the addition of Lynx and LinkSet scores to the decoder scores does not give significant improvement in the top NIST score, nor the top ten NIST scores, except on very short sentences. Examination of the NIST scores for short sentences shows a much wider range of scores within each N-best list; this is probably due mostly to the length normalization in the NIST score. On these short sentences, where there is more room for improvement through reranking, significant improvement is shown when Lynx and/or LinkSet scores are used. Equivalent comparisons of

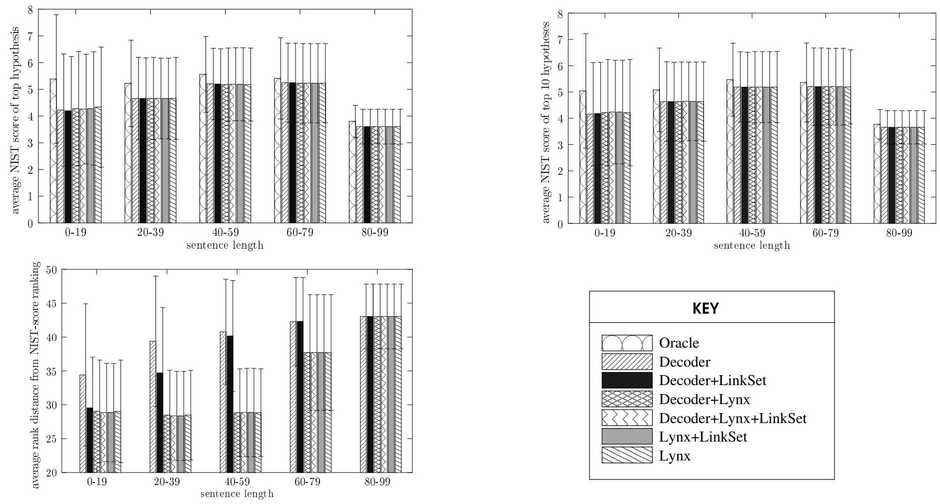

Figure 5.11: Charts showing mean scores with error bars at one standard deviation for the same data shown in Figures 5.7 (top left), 5.8 (top right), and 5.1 (bottom).

average BLEU scores for the number-one-ranked hypothesis the top 10 hypotheses, respectively, are shown in Figures 5.9 and 5.10. Their shape is essentially the same as for the NIST scores.

Because of the many lines shown on each of the above-mentioned figures, they are not shown with error bars, which would make the charts illegible. Instead, Figures 5.11 and 5.12 summarize the same data, this time using sentence-length buckets of width 20 (rather than 5) and including error bars. The variance in scores appears fairly high, but this is not surprising considering the wide range of sentences on which the translation system was evaluated. Unfortunately, the consequence is a result with very weak statistical significance.

5.6 Discussion

Our experiments have shown that scores assigned by the Lynx and LinkSet parsers to translation decoder output can produce a significant improvement in the ranking of the hypotheses over the decoder's original ranking. However, when only the number-one hypothesis is taken into account, no significant gains were made in the end-to-end performance of the translation system. Improving this bottom-line measure of real, useable translation quality is a very hard problem, especially when we confine ourselves to the space between current system performance and

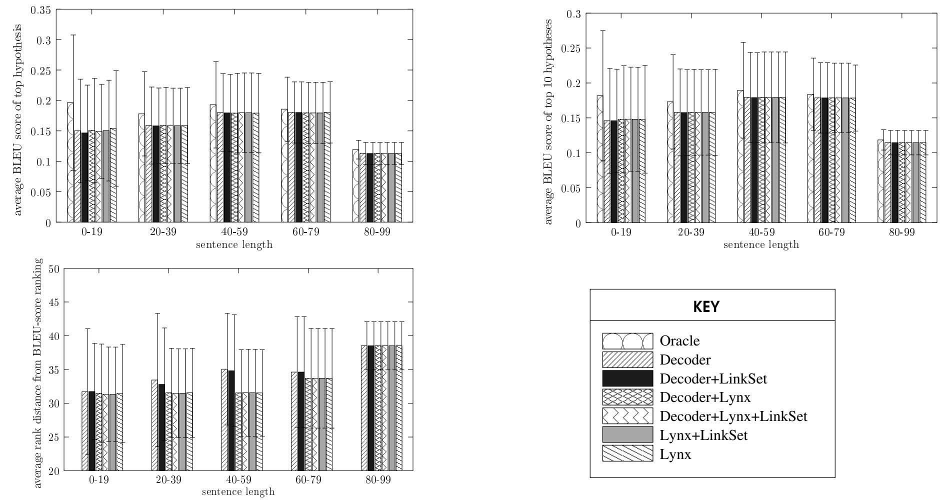

Figure 5.12: Charts showing mean scores with error bars at one standard deviation for the same data shown in Figures 5.9 (top left), 5.10 (top right), and 5.2 (bottom).

the best performance possible through reordering the hypotheses the decoder currently produces.

If a new translation system were designed to incorporate these models throughout, there would be more space for improvement. However, designing such a system would require some well-considered compromises between a complete search on the true model, which would be far too expensive for most applications, and a time-saving approximation such as the one we have demonstrated. A complete search would include not only all possible structures of the foreign sentence to be translated, but also all English structures of which it could be a transformation, and all English strings consistent with those structures. While this space may be too large for a practical system, it does suggest a way forward. The decoding problem could be treated as a parsing problem, as in the LinkSet bilingual parser. In this case, however, where no English string is given, the grammar with which the foreign sentence is parsed would be much less constrained. However, a dynamic beam search may be sufficiently efficient to make the problem tractable, though perhaps still impractical for real-time applications. The grammar with which a foreign sentence to be translated would be parsed would be an English grammar composed with a tree-translation grammar; that is, for example, the Lynx model combined with the LinkSet model. This would be a natural extension of the work presented in this chapter.

In order to enable a decoder built on the Lynx and LinkSet models to have reasonable speed, it is important to constrain the models to reduce the search space, without reducing it so far as to eliminate correct translations and structures. In practice, the IBM Model 2 word-translation models used to train and use the LinkSet structural model are extremely noisy. However, an overly clean model would fail to allow many sentence pairs to align at all, since many translations are not exact, and related words that are not proper translations of each other should still be able to align with each other. While EM can begin to address this problem, our experiments show that a much simpler dynamic-beam–style attenuation does just as well or better than a sentence-level approximation of EM (see Section 4.4.2). It is quite possible that a more exact EM formulation would not only deal better with this problem, but result in a better model all around (see Section 4.5).

In the interest of improved model quality, it would also be interesting to compare various parameterizations of the LinkSet model within the same framework and on the same experimental data. Perhaps changing the structure of the model in terms of its dependencies and independence assumptions would lead to a better model. Unfortunately, in the current framework and code base, varying the structure in this way is a nontrivial task.

An modification of the model that could potentially bring big rewards in translation accuracy, although it might bring corresponding costs in training and decoding time and model size, would be to condition word-translation probabilities on syntactic structures. For example, the probability of a foreign word could be conditioned not only on the corresponding English word, but also on the label of the link generating the word. One challenge in training such a model is that it might require a very large corpus of annotated sentences, although automatic annotation should be very useful here, as it was in the training of the Lynx parser.

The main point of this work has been the integration of syntactic structure into the statistical models used in translation — both monolingual language models and models of translation from one language to another. Through the development of the Lynx and LinkSet models and parsers, we have taken a big step in that direction, and shown that adding models of syntactic structure can improve the output of a state-of-the-art statistical translation system. In the future, we hope to see systems that continue in this direction to combine the best of both the knowledge-based and statistical worlds, using statistical models carefully crafted to capture all the features linguistic theory tells us are relevant, constrained enough to be trained on a reasonable amount of data, and general enough to apply to a broad range of languages and domains.

Chapter 6

Conclusions

6.1 Conclusions

Here is a concise summary of the conclusions detailed in the chapters above:

1. To train a new statistical parser, it is not necessary to hand-annotate a large corpus with structural information if a rule-based parser is available, because a parser trained on automatically-generated data can perform as well as or better than the original parser, taking advantage of the distributional information in the unannotated training corpus.

2. A statistical model of the relationship between the syntactic structures of two different languages can be effectively learned from a bilingual corpus by an unsupervised learning technique, even when syntactic annotations are available for only one of the languages.

   (a) The alignments found when training this model are significantly better than those given by a standard IBM-style alignment model.

   (b) Expectation maximization training improves the model.

   (c) Careful model design is important in order to avoid the complementary hazards of computational complexity and data sparsity.

3. Using a bilingual corpus and an existing parser in one language, a new parser can be automatically induced for the other language, without the aid of a language expert.

   (a) While the induced parser naturally does not work as well as the orignal English parser, it does better than baseline techniques at constituent bracketing, and in many cases yields syntactic analyses that are completely correct.
   (b) A rough-grain approximation of expectation maximization training does not improve the model. However, a more fine-grained EM approach might work much better.
   (c) The induced parser can be incrementally improved through cleaning of the translation lexicon and adjustment of the model to handle common structures missed during the original training.
   (d) The assumption that phrases for translation must be contiguous, a weakness of many translation systems, can be relaxed with a simple modification to the LinkSet model, resulting in improved alignments and parsing.

4. Integrating syntactic structure into the statistical models used for automatic translation can increase the quality of translated output.
   (a) The use of syntax-based models significantly improves the overall ranking of hypotheses according to the standard NIST evaluation of each hypothesis.
   (b) However, improvement of the top-ranked hypothesis, which has the most practical value, was small.
   (c) This improvement was limited by the small space available for improvement of a state-of-the-art statistical translation system through reranking of translation hypotheses.

6.2 Contributions

These are the main contributions of this thesis:

1. A new technique for training a statistical link grammar parser from examples, implemented as the Lynx parser.
2. A new syntax-based generative model of the relationship between two languages, with an efficient unsupervised training algorithm implemented as the LinkSet bilingual parser.

3. A new method of inducing a parser for a foreign language given a bilingual corpus and an English parser, implemented as a combination of the Lynx and LinkSet parsers.

4. A new method of incorporating syntactic structure into a statistical translation model, demonstrated by reranking decoder outputs using the Lynx and LinkSet parsers.

6.3 Future Research

The discussions throughout this thesis immediately suggest some low-hanging fruit for near-term future research:

1. Investigate how to improve the LinkSet model by altering its structure and parameters (see Section 3.7). Incorporate the leaf-node–duplication operation demonstrated above (in Section 4.4.4) into the model.

2. Implement fine-grained EM training for parser induction, and see how much it improves the resulting foreign parsers (see Section 4.5).

3. Try seeding an induced foreign-language parser with selected hand-annotated data, to investigate the work/payoff ratio for incremental improvement.

4. Develop better evaluation methods for statistical link grammar parsers (see Section 4.4.1).

5. Build a translation system that fully incorporates the structural model tested above (see Section 5.6).

The juiciest of these low-hanging fruits is probably the advanced EM training for foreign-parser induction. It seems quite possible that this will give a significant improvement in parser quality, while maintaining the fully-automatic nature of the induction algorithm.

Taking a step back from the details of this thesis, we can see a theme running throughout: **incorporating syntax into generative models of human language**, which is just a special case of a more general theme: **combining knowledge with statistical models**. In this thesis we incorporated linguistic knowledge into statistical models of language structure and of the relationship between different languages' structures. The linguistic knowledge appears in the guise of model structure and of the selection and dependencies of model parameters. That is, theory guides the construction of models so that they can capture the phenomena the theory tells us are relevant. An alternate way of combining knowledge with statistical modeling is to incorporate plenty of small statistical models into what is structurally a knowledge-based system, which could also be pictured as a rule-based system in which statistical models govern the application of each rule. Such approaches that give more explicit attention to rules may be more appropriate for certain applications, especially those in which the rules can be easily expressed. Our approach, which implicitly incorporates knowledge into the design of an over-arching statistical model, is in many ways more elegant because it has a simple probabilistic interpretation and is very flexible in its ability to be trained on new data without special programming. However, it can sometimes be difficult to modify such a model to capture phenomena previously missed, or to make sure it does not suffer too much from the complexity that requires too much time, space, and training data to be useful.

Therefore, it would be interesting and useful to investigate in a more general way how to incorporate theoretical knowledge into the design of generative models. How can such a model be brought quickly and flexibly from the drawing board to an implementation including EM training and search, allowing changes to the parameterization of the model without significant recoding? (Perhaps maximum entropy models would be a good starting point.) A toolkit for this purpose would be a very useful research tool.

A flexible platform for testing knowledge-motivated generative models would present an opportunity to explore a wide range of models for various language-processing tasks, including parsing and translation, not to mention the ubiquitous language modeling used in speech recognition, generation, translation, and many other tasks. It would be interesting to extend current models not only to include syntax, but also word-segmentation, morphology and phonology, or in the other direction, discourse structures larger than sentences.

In order to enable computers to handle the complexities of natural language, and of many other phenomena as well, we need the combined power of knowledge and statistics. Creating tools to bring these together should pay off not only in natural language processing, but in a variety of fields.

Bibliography

[1] Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1), March 2000.

[2] A. Berger, P. Brown, S. Della Pietra, V. Della Pietra, J. Lafferty, H. Printz, and L. Ures. The Candide system for machine translation. In *The Proceedings of the ARPA Conference on Human Language Technology*, pages 157–162, 1994.

[3] Eric Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, IT, 1992.

[4] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[5] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[6] E. Charniak, K. Knight, and K. Yamada. Syntax-based language models for machine translation. In *The Proceedings of MT Summit IX*, 2003.

[7] Eugene Charniak. Immediate-head parsing for language models. In *Meeting of the Association for Computational Linguistics*, pages 116–123, 2001.

[8] Ciprian Chelba, David Engle, Frederick Jelinek, Victor M. Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. Structure and performance of a dependency

language model. In *The Proceedings of Eurospeech '97*, pages 2775–2778, Rhodes, Greece, 1997.

[9] Michael Collins. Three generative, lexicalized models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 1997.

[10] Michael Collins. Discriminative reranking for natural language parsing. In *Proc. 17th International Conf. on Machine Learning*, pages 175–182. Morgan Kaufmann, San Francisco, CA, 2000.

[11] Jason Eisner. Three new probabilistic models for dependency parsing: An exploration. In *The Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345, 1996.

[12] Jason Eisner. An empirical comparison of probability models for dependency grammar. Technical report, University of Pennsylvania Technical Report, 1997.

[13] Jason Eisner. Bilexical grammars and their cubic-time parsing algorithms. In *Harry Bunt and Anton Nijholt, editors, Advances in Probabilistic and Other Parsing Technologies, chapter 1. Kluwer Academic Publishers*, 2000.

[14] Sanjeev Khudanpur Anoop Sarkar Kenji Yamada Alex Fraser Shankar Kumar Libin Shen David Smith Katherine Eng Viren Jain Zhen Jin Franz Josef Och, Daniel Gildea and Dragomir Radev. Syntax for statistical machine translation. Technical report, Johns Hopkins University, 2003.

[15] Jianfeng Gao and Hisami Suzuki. Unsupervised learning of dependency structure for language modeling. In *Association for Computational Linguistics*, 2003.

[16] Daniel Gildea. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*, 2003.

[17] John Lafferty, Daniel Sleator, and Davy Temperley. Grammatical trigrams: A probabilistic model of link grammar. In *The Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 89–97, 1992.

[18] Bruce T. Lowerre. *The HARPY Speech Recognition System*. PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, 1976.

[19] Eva Wai man Fong and Dekai Wu. Learning restricted probabilistic link grammars. Technical report, University of Science and Technology, Hong Kong, 1995.

[20] Arul Menezes and Stephen D. Richardson. A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics – DDMT Workshop*, Toulouse, France, 2001.

[21] H. Ney, D. Mergel, A. Noll, and A. Paeseler. A data-driven organization of the dynamic programming beam search for continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 833–836, 1987.

[22] F. J. Och and H. Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hongkong, China, October 2000.

[23] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ures. Inference and estimation of a long-range trigram model. In *The Proceedings of the Second International Colloquium on Grammatical Inference and Applications*, 1994.

[24] Brian Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276, 2001.

[25] Daniel Sleator and Davy Temperley. Parsing English with a link grammar. Technical report, Carnegie Mellon, 1991.

[26] Ye-Yi Wang. *Grammar Inference and Statistical Machine Translation*. PhD thesis, Language Technologies Institute, Carnegie Mellon, 1998.

[27] Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, 1997.

[28] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

[29] David Yarowsky and Grace Ngai. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *The Proceedings of the Annual Meeting of the North American Association for Computational Linguistics*, pages 200–207, 2001.

[30] David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *The Proceedings of NLT-2001*, pages 109–116, 2001.

[31] Klaus Zechner. Building chunk level representations for spontaneous speech in unrestricted domains: The CHUNKY system and its application to reranking nbest lists of a speech recognizer. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.