# A Framework for Incorporating Alignment Information in Parsing

**Mark Hopkins**
Dept. of Computational Linguistics
Saarland University
Saarbrücken, Germany
mhopkins@coli.uni-sb.de

**Jonas Kuhn**
Dept. of Computational Linguistics
Saarland University
Saarbrücken, Germany
jonask@coli.uni-sb.de

## Abstract

The standard PCFG approach to parsing is quite successful on certain domains, but is relatively inflexible in the type of feature information we can include in its probabilistic model. In this work, we discuss preliminary work in developing a new probabilistic parsing model that allows us to easily incorporate many different types of features, including crosslingual information. We show how this model can be used to build a successful parser for a small handmade gold-standard corpus of 188 sentences (in 3 languages) from the Europarl corpus.

## 1 Introduction

Much of the current research into probabilistic parsing is founded on probabilistic context-free grammars (PCFGs) (Collins, 1999; Charniak, 2000; Charniak, 2001). For instance, consider the parse tree in Figure 1. One way to decompose this parse tree is to view it as a sequence of applications of CFG rules. For this particular tree, we could view it as the application of rule "NP → NP PP," followed by rule "NP → DT NN," followed by rule "DT → that," and so forth. Hence instead of analyzing $P(tree)$, we deal with the more modular:

$$P(NP \rightarrow NP\ PP,\ NP \rightarrow DT\ NN,\ DT \rightarrow that,\ NN \rightarrow money,\ PP \rightarrow IN\ NP,\ IN \rightarrow in,\ NP \rightarrow DT\ NN,\ DT \rightarrow the,\ NN \rightarrow market)$$

Obviously this joint distribution is just as difficult to assess and compute with as $P(tree)$. However there exist cubic time algorithms to find the most likely parse if we assume that all CFG rule applications are marginally independent of one another. In other words, we need to assume that the above expression is equivalent to the following:

$$P(NP \rightarrow NP\ PP) \cdot P(NP \rightarrow DT\ NN) \cdot P(DT \rightarrow that) \cdot P(NN \rightarrow money) \cdot P(PP \rightarrow IN\ NP) \cdot P(IN \rightarrow in) \cdot P(NP \rightarrow DT\ NN) \cdot P(DT \rightarrow the) \cdot P(NN \rightarrow market)$$

It is straightforward to assess the probability of the factors of this expression from a corpus using relative frequency. Then using these learned probabilities, we can find the most likely parse of a given sentence using the aforementioned cubic algorithms.

The problem, of course, with this simplification is that although it is computationally attractive, it is usually too strong of an independence assumption. To mitigate this loss of context, without sacrificing algorithmic tractability, typically researchers annotate the nodes of the parse tree with contextual information. For instance, it has been found to be useful to annotate nodes with their parent labels (Johnson, 1998), as shown in Figure 2. In this case, we would be learning probabilities like: P(PP-NP → IN-PP NP-PP).

The choice of which annotations to use is one of the main features that distinguish parsers based on this approach. Generally, this approach has proven quite effective in producing English phrase-structure grammar parsers that perform well on the Penn Treebank.

One drawback of this approach is that it is somewhat inflexible. Because we are adding probabilistic context by changing the data itself, we make our data increasingly sparse as we add features. Thus we are constrained from adding too

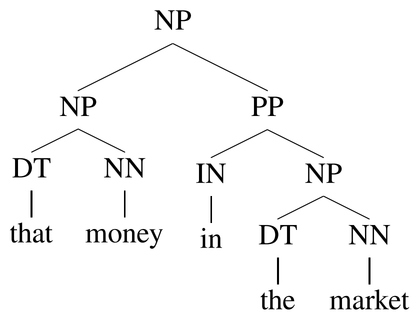

Figure 1: Example parse tree.

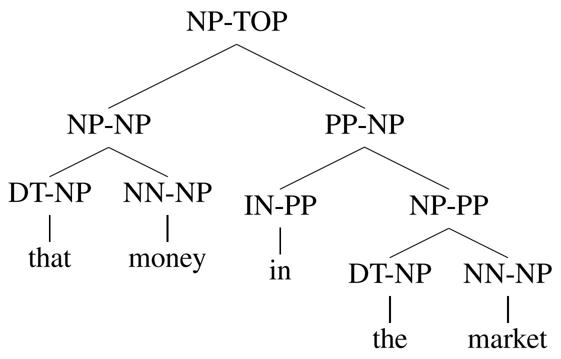

Figure 2: Example parse tree with parent annotations.

many features, because at some point we will not have enough data to sustain them. Hence in this approach, feature selection is not merely a matter of including good features. Rather, we must strike a delicate balance between how much context we want to include versus how much we dare to partition our data set.

This poses a problem when we have spent time and energy to find a good set of features that work well for a given parsing task on a given domain. For a different parsing task or domain, our parser may work poorly out-of-the-box, and it is no trivial matter to evaluate how we might adapt our feature set for this new task. Furthermore, if we gain access to a new source of feature information, then it is unclear how to incorporate such information into such a parser.

Namely, in this paper, we are interested in seeing how the cross-lingual information contained by sentence alignments can help the performance of a parser. We have a small gold-standard corpus of shallow-parsed parallel sentences (in English, French, and German) from the Europarl corpus. Because of the difficulty of testing new features using PCFG-based parsers, we propose a new probabilistic parsing framework that allows us to flexibly add features. The closest relative

|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | true | true | false | false | true |
| 2 | - | true | false | false | false |
| 3 | - | - | true | false | true |
| 4 | - | - | - | true | true |
| 5 | - | - | - | - | true |

Figure 3: Span chart for example parse tree. Chart entry $(i, j) = true$ iff span $(i, j)$ is a constituent in the tree.

of our framework is the maximum-entropy parser of Ratnaparkhi(Ratnaparkhi, 1997). Both frameworks are bottom-up, but while Ratnaparkhi's views parse trees as the sequence of applications of four different types of tree construction rules, our framework strives to be somewhat simpler and more general.

## 2 The Probability Model

The example parse tree in Figure 1 can also be decomposed in the following manner. First, we can represent the unlabeled tree with a boolean-valued chart (which we will call the *span chart*) that assigns the value of $true$ to a span if it is a constituent in the tree, and $false$ otherwise. The *span chart* for Figure 1 is shown in Figure 3.

To represent the labels, we simply add similar charts for each labeling scheme present in the tree. For a parse tree, there are typically three types of labels: words, preterminal tags, and nonterminals. Thus we need three labeling charts. Labeling charts for our example parse tree are depicted in Figure 4. Note that for words and preterminals, it is not really necessary to have a two-dimensional chart, but we do so here to motivate the general model.

The general model is as follows. Define a *labeling scheme* as a set of symbols including a special symbol $null$ (this will designate that a given span is unlabeled). For instance, we might define $L^{NT} = \{null, NP, PP, IN, DT\}$ to be a labeling scheme for non-terminals. Let $\mathcal{L} = \{L^1, L^2, ...L^m\}$ be a set of labeling schemes. Define a *model variable* of $\mathcal{L}$ as a symbol of the form $S_{ij}$ or $L^k_{ij}$, for positive integers $i$, $j$, $k$, such that $j \geq i$ and $k \leq m$. The *domain* of model variable $S_{ij}$ is $\{true, false\}$ (these variables indicate whether a given span is a tree constituent). The *domain* of model variable $L^k_{ij}$ is $L^k$ (these variables indicate which label from $L^k$ is assigned to span

|  | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | that | null | null | null | null |
| 2 | - | money | null | null | null |
| 3 | - | - | in | null | null |
| 4 | - | - | - | the | null |
| 5 | - | - | - | - | market |

|  | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | DT | null | null | null | null |
| 2 | - | NN | null | null | null |
| 3 | - | - | IN | null | null |
| 4 | - | - | - | DT | null |
| 5 | - | - | - | - | NN |

|  | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | null | NP | null | null | NP |
| 2 | - | null | null | null | null |
| 3 | - | - | null | null | PP |
| 4 | - | - | - | null | NP |
| 5 | - | - | - | - | null |

Figure 4: Labeling charts for example parse tree: the top chart is for word labels, the middle chart is for preterminal tag labels, and the bottom chart is for nonterminal labels. $null$ denotes an unlabeled span.

$i, j$). Define a *model order* of $\mathcal{L}$ as a total ordering $\Omega$ of the model variables of $\mathcal{L}$ such that for all $i, j, k$: $\Omega(S_{ij}) < \Omega(L^k_{ij})$ (i.e. we decide whether a span is a constituent before attempting to label it). Let $\Omega_n$ denote the finite subset of $\Omega$ that includes precisely the model variables of the form $S_{ij}$ or $L^k_{ij}$, where $j \le n$.

Given a set $\mathcal{L}$ of labeling schemes and a model order $\Omega$ of $\mathcal{L}$, a preliminary generative story might look like the following:

1. Choose a positive integer $n$.
2. In the order defined by $\Omega_n$, assign a value to every model variable of $\Omega_n$ from its domain, conditioned on any previous assignments made.

Thus some model order $\Omega$ for our example might instruct us to first choose whether span (4, 5) is a constituent, for which we might say "true," then instruct us to choose a label for that constituent, for which we might say "NP," and so forth.

There are a couple of problems with this generative story. One problem is that it allows us to make structural decisions that do not result in a well-formed tree. For instance, we should not be permitted to assign the value $true$ to both variable $S_{13}$ and $S_{24}$. Generally, we cannot allow two model variables $S_{ij}$ and $S_{kl}$ to both be assigned $true$ if they *properly overlap*, i.e. their spans overlap and one is not a subspan of the other. We should also ensure that the leaves and the root are considered constituents. Another problem is that it allows us to make labeling decisions that do not correspond with our chosen structure. It should not be possible to label a span which is not a constituent.

With this in mind, we revise our generative story.

1. Choose a positive integer $n$ from distribution $P_0$.
2. In the order defined by $\Omega_n$, process model variable $x$ of $\Omega_n$:
   (a) If $x = S_{ij}$, then:
      i. Automatically assign the value $false$ if there exists a properly overlapping model variable $S_{kl}$ such that $S_{kl}$ has already been assigned the value $true$.
      ii. Automatically assign the value $true$ if $i = j$ or if $i = 1$ and $j = n$.
      iii. Otherwise assign a value $s_{ij}$ to $S_{ij}$ from its domain, drawn from some probability distribution $P_S$ conditioned on all previous variable assignments.
   (b) If $x = L^k_{ij}$, then:
      i. Automatically assign the value $null$ to $L^k_{ij}$ if $S_{ij}$ was assigned the value $false$ (note that this is well-defined because of way we defined model order).
      ii. Otherwise assign a value $l^k_{ij}$ to $L^k_{ij}$ from its domain, drawn from some probability distribution $P_k$ conditioned on all previous variable assignments.

Defining $\Omega^<_n(x) = \{y \in \Omega_n | \Omega(y) < \Omega(x)\}$ for $x \in \Omega_n$, we can decompose $P(tree)$ into the following expression:

$$P_0(n) \cdot \prod_{S_{ij} \in \Omega_n} P_S(s_{ij} | n, \Omega_n^<(S_{ij})) \cdot \prod_{L_{ij}^k \in \Omega_n} P_k(l_{ij}^k | n, \Omega_n^<(L_{ij}^k))$$

where $P_S$ and $P_k$ obey the constraints given in the generative story above (e.g. $P_S(S_{ii} = true) = 1$, etc.)

Obviously it is impractical to learn conditional distributions over every conceivable history, so instead we choose a small set $\mathbf{F}$ of feature variables, and provide a set of functions $\mathcal{F}_n$ that map every partial history of $\Omega_n$ to some feature vector $\mathbf{f} \in \mathbf{F}$ (later we will see examples of such feature functions). Then we make the assumption that:

$$P_S(s_{ij} | n, \Omega_n^<(S_{ij}) = P_S(s_{ij} | \mathbf{f})$$

where $\mathbf{f} = \mathcal{F}_n(\Omega_n^<(S_{ij}))$ and that

$$P_k(l_{ij}^k | n, \Omega_n^<(S_{ij}) = P_k(l_{ij}^k | \mathbf{f})$$

where $\mathbf{f} = \mathcal{F}_n(\Omega_n^<(L_{ij}^k))$.

In this way, our learning task is simplified to learn functions $P_0(n)$, $P_S(s_{ij}|\mathbf{f})$, and $P_k(l_{ij}^k|\mathbf{f})$. Given a corpus of labeled trees, it is straightforward to extract the training instances for these distributions and then use these instances to learn distributions using one's preferred learning method (e.g., maximum entropy models or decision trees).

For this paper, we are interested in parse trees which have three labeling schemes. Let $\mathcal{L} = \{L^{word}, L^{PT}, L^{NT}\}$, where $L^{word}$ is a labeling scheme for words, $L^{PT}$ is a labeling scheme for preterminals, and $L^{NT}$ is a labeling scheme for nonterminals. We will define model order $\Omega$ such that:

1. $\Omega(S_{ij}) < \Omega(L_{ij}^{word}) < \Omega(L_{ij}^{PT}) < \Omega(L_{ij}^{NT})$.

2. $\Omega(L_{ij}^{NT}) < \Omega(S_{kl})$ iff $j - i < l - k$ or $(j - i = l - k$ and $i < k)$.

In this work, we are not as much interested in learning a marginal distribution over parse trees, but rather a conditional distribution for parse trees, given a tagged sentence (from which $n$ is also known). We will assume that $P_{word}$ is conditionally independent of all the other model variables, given $n$ and the $L_{ij}^{word}$ variables. We will also assume that $P_{pt}$ is conditionally independent of the other model variables, given $n$, the $L_{ij}^{word}$ variables, and the $L_{ij}^{pt}$ variables. These assumptions allow us to express $P(tree | n, L_{ij}^{word}, L_{ij}^{pt})$ as the following:

$$\prod_{S_{ij} \in \Omega_n} P_S(s_{ij} | \mathbf{f}_S) \cdot \prod_{L_{ij}^{nt} \in \Omega_n} P_{nt}(l_{ij}^{nt} | \mathbf{f}_{nt})$$

where $\mathbf{f}_S = \mathcal{F}_n(\Omega_n^<(S_{ij}))$ and $\mathbf{f}_{nt} = \mathcal{F}_n(\Omega_n^<(L_{ij}^{nt}))$. Hence our learning task in this paper will be to learn the probability distributions $P_S(s_{ij}|\mathbf{f}_S)$ and $P_{nt}(l_{ij}^{nt}|\mathbf{f}_{nt})$, for some choice of feature functions $\mathcal{F}_n$.

## 3 Decoding

For the PCFG parsing model, we can find $argmax_{tree} P(tree|sentence)$ using a cubic-time dynamic programming-based algorithm. By adopting a more flexible probabilistic model, we sacrifice polynomial-time guarantees. Nevertheless, we can still devise search algorithms that work efficiently in practice. For the decoding of the probabilistic model of the previous section, we choose a depth-first branch-and-bound approach, specifically because of two advantages. First, this approach is linear space. Second, it is anytime, i.e. it finds a (typically good) solution early and improves this solution as the search progresses. Thus if one does not wish the spend the time to run the search to completion (and ensure optimality), one can use this algorithm easily as a heuristic.

The search space is simple to define. Given a set $\mathcal{L}$ of labeling schemes and a model order $\Omega$ of $\mathcal{L}$, the search algorithm simply makes assignments to the model variables (depth-first) in the order defined by $\Omega$.

This search space can clearly grow to be quite large, however in practice the search speed is improved drastically by using branch-and-bound backtracking. Namely, at any choice point in the search space, we first choose the least cost child to expand. In this way, we quickly obtain a greedy solution. After that point, we can continue to keep track of the best solution we have found so far, and if at any point we reach an internal node of our search tree with partial cost greater than the total cost of our best solution, we can discard this node and discontinue exploration of that subtree. This technique can result in a significant aggregrate savings of computation time, depending on

**EN:** [$_1$ [$_2$ On behalf of the European People 's Party , ] [$_3$ I] call [$_5$ for a vote [$_6$ in favour of that motion ] ] ]

**FR:** [$_1$ [$_2$ Au nom du Parti populaire européen ,] [$_3$ je] demande [$_5$ l' adoption [$_6$ de cette résolution] ] ]

**DE:** [$_1$ [$_2$ Im Namen der Europäischen Volkspartei ] rufe [$_3$ ich] [$_4$ Sie] auf , [$_5$ [$_6$ diesem Entschließungsantrag] zuzustimmen ] ]

**ES:** [$_1$ [$_2$ En nombre del Grupo del Partido Popular Europeo ,] solicito [$_5$ la aprobación [$_6$ de la resolución] ] ]

Figure 5: Annotated sentence tuple

the nature of the cost function. For our limited parsing domain, it appears to perform quite well, taking fractions of a second to parse each sentence (which are short, with a maximum of 20 words per sentence).

## 4 Experiments

Our parsing domain is based on a "lean" phrase correspondence representation for multitexts from parallel corpora (i.e., tuples of sentences that are translations of each other). We defined an annotation scheme that focuses on translational correspondence of *phrasal units* that have a distinct, language-independent *semantic status*. It is a hypothesis of our longer-term project that such a semantically motivated, relatively coarse phrase correspondence relation is most suitable for weakly supervised approaches to parsing of large amounts of parallel corpus data. Based on this lean phrase structure format, we intend to explore an alternative to the annotation projection approach to cross-linguistic bootstrapping of parsers by (Hwa et al., 2005). They depart from a standard treebank parser for English, "projecting" its analyses to another language using word alignments over a parallel corpus. Our planned bootstrapping approach will not start out with a given parser for English (or any other language), but use a small set of manually annotated seed data following the lean phrase correspondence scheme, and then bootstrap consensus representations on large amounts of unannotated multitext data. At the present stage, we only present experiments for training an initial system on a set of seed data.

The annotation scheme underlying in the gold standard annotation consists of (A) a bracketing for each language and (B) a correspondence relation of the constituents across languages. Neither the constituents nor the embedding or correspondent relations were labelled.

The guiding principle for bracketing (A) is very simple: all and only the units that clearly play the role of a semantic argument or modifier in a larger unit are bracketed. This means that function words, light verbs, "bleeched" PPs like *in spite of* etc. are included with the content-bearing elements. This leads to a relatively flat bracketing structure. Referring or quantified expressions that may include adjectives and possessive NPs or PPs are also bracketed as single constituents (e.g., [ *the president of France* ]), unless the semantic relations reflected by the internal embedding are part of the predication of the sentence. A few more specific annotation rules were specified for cases like coordination and discontinuous constituents.

The correspondence relation (B) is guided by semantic correspondence of the bracketed units; the mapping need not preserve the tree structure. Neither does a constituent need to have a correspondent in all (or any) of the other languages (since the content of this constituent may be implicit in other languages, or subsumed by the content of another constituent). "Semantic correspondence" is not restricted to truth-conditional equivalence, but is generalized to situations where two units just serve the same rhetorical function in the original text and the translation.

Figure 5 is an annotation example. Note that index 4 (the audience addressed by the speaker) is realized overtly only in German (*Sie* 'you'); in Spanish, index 3 is realized only in the verbal inflection (which is not annotated). A more detailed discussion of the annotation scheme is presented in (Kuhn and Jellinghaus, to appear).

For the current parsing experiments, only the bracketing within each of three languages (English, French, German) is used; the cross-linguistic phrase correspondences are ignored (although we intend to include them in future experiments). We automatically tagged the training and test data in English, French, and German with Schmid's decision-tree part-of-speech tagger (Schmid, 1994).

The training data were taken from the sentence-aligned Europarl corpus and consisted of 188 sentences for each of the three languages, with max-

| Feature Notation | Description |
|---|---|
| p(*language*) | the preterminal tag of word $x-1$ (*null* if does not exist) |
| f(*language*) | the preterminal tag of word $x$ |
| l(*language*) | the preterminal tag of word $y$ |
| n(*language*) | the preterminal tag of word $y-1$ (*null* if does not exist) |
| lng | the length of the span (i.e. $y-x+1$) |

Figure 6: Features for span $(x, y)$. E = English, F = French, G = German

| English features | Crosslingual features | Rec. | Prec. | F-score | No cross |
|---|---|---|---|---|---|
| p(E), f(E), l(E) | none | 40.3 | 63.6 | 49.4 ($\pm$3.9%) | 57.1 |
| | p(F), f(F), l(F) | 43.1 | 67.6 | 52.6 ($\pm$4.0%) | 61.2 |
| | p(G), f(G), l(G) | 45.9 | 66.8 | 54.4 ($\pm$4.0%) | 69.4 |
| | p(F), f(F), l(F), p(G), f(G), l(G) | 44.5 | 65.5 | 53.0 ($\pm$3.9%) | 65.3 |
| p(E), f(E), l(E), n(E) | none | 57.2 | 68.6 | 62.4 ($\pm$4.0%) | 65.3 |
| | p(F), f(F), l(F), n(F) | 56.6 | 71.9 | 63.3 ($\pm$4.0%) | 75.5 |
| | p(G), f(G), l(G), n(G) | 57.9 | 67.7 | 62.5 ($\pm$3.9%) | 67.3 |
| | p(F), f(F), l(F), n(F), p(G), f(G), l(G), n(G) | 57.9 | 72.1 | 64.2 ($\pm$4.0%) | 77.6 |
| p(E), f(E), l(E), n(E), lng | none | 64.8 | 71.2 | 67.9 ($\pm$4.0%) | 79.6 |
| | p(F), f(F), l(F), n(F), lng | 62.1 | 74.4 | 67.7 ($\pm$4.0%) | 83.7 |
| | p(G), f(G), l(G), n(G), lng | 61.4 | 78.8 | 69.0 ($\pm$4.1%) | 83.7 |
| | p(F), f(F), l(F), n(F), p(G), f(G), l(G), n(G), lng | 63.1 | 76.9 | 69.3 ($\pm$4.1%) | 81.6 |
| | BIKEL | 57.9 | 60.2 | 59.1 ($\pm$3.8%) | 57.1 |

Figure 7: Parsing results for various feature sets, and the Bikel baseline. The F-scores are annotated with 95% confidence intervals.

imal length of 21 words in English (French: 38; German: 24) and an average length of 14.0 words in English (French 16.8; German 13.6). The test data were 50 sentences for each language, picked arbitrarily with the same length restrictions. The training and test data were manually aligned following the guidelines.[1]

For the word alignments used as learning features, we used GIZA++, relying on the default parameters. We trained the alignments on the full Europarl corpus for both directions of each language pair.

As a baseline system we trained Bikel's reimplementation (Bikel, 2004) of Collins' parser (Collins, 1999) on the gold standard (English) training data, applying a simple additional smoothing procedure for the modifier events in order to counteract some obvious data sparseness issues.[2]

Since we were attempting to learn unlabeled trees, in this experiment we only needed to learn the probabilistic model of Section 3 with no labeling schemes. Hence we need only to learn the probability distribution:

$$P_S(s_{ij}|\mathbf{f}_S)$$

In other words, we need to learn the probability that a given span is a tree constituent, given some set of features of the words and preterminal tags of the sentences, as well as the previous span decisions we have made. The main decision that

[1]A subset of 39 sentences was annotated by two people independently, leading to an F-Score in bracketing agreement between 84 and 90 for the three languages. Since finding an annotation scheme that works well in the bootstrapping set-up is an issue on our research agenda, we postpone a more detailed analysis of the annotation process until it becomes clear that a particular scheme is indeed useful.

[2]For the nonterminal labels, we defined the left-most lexical daughter in each local subtree of depth 1 to project its part-of-speech category to the phrase level and introduced a special nonterminal label for the rare case of nonterminal nodes dominating no preterminal node.

remains, then, is which feature set to use. The features we employ are very simple. Namely, for span $(i, j)$ we consider the preterminal tags of words $i-1$, $i$, $j$, and $j+1$, as well as the French and German preterminal tags of the words to which these English words align. Finally, we also use the length of the span as a feature. The features considered are summarized in Figure 6.

To learn the conditional probability distribututions, we choose to use maximum entropy models because of their popularity and the availability of software packages. Specifically, we use the MEGAM package (Daumé III, 2004) from USC/ISI.

We did experiments for a number of different feature sets, with and without alignment features. The results (precision, recall, F-score, and the percentage of sentences with no cross-bracketing) are summarized in Figure 7. Note that with a very simple set of features (the previous, first, last, and next preterminal tags of the sequence), our parser performs on par with the Bikel baseline. Adding the length of the sequence as a feature increases the quality of the parser to a statistically significant difference over the baseline. The crosslingual information provided (which is admittedly naive) does not provide a statistically significant improvement over the vanilla set of features. The conclusion to be drawn is not that crosslingual information does not help (such a conclusion should not be drawn from the meager set of crosslingual features we have used here for demonstration purposes). Rather, the take-away point is that such information can be easily incorporated using this framework.

## 5 Discussion

One of the primary concerns about this framework is speed, since the decoding algorithm for our probabilistic model is not polynomial-time like the decoding algorithms for PCFG parsing. Nevertheless, in our experiments with shallow parsed 20-word sentences, time was not a factor. Furthermore, in our ongoing research applying this probabilistic framework to the task of Penn Treebank-style parsing, this approach appears to also be viable for the 40-word sentences of Sections 22 and 23 of the WSJ treebank. A strong mitigating factor of the theoretical intractibility is the fact that we have an *anytime* decoding algorithm, hence even in cases when we cannot run the algorithm to completion (for a guaranteed optimal solution), the algorithm always returns some solution, the quality of which increases over time. Hence we can tell the algorithm how much time it has to compute, and it will return the best solution it can compute in that time frame.

This work suggests that one can get a good quality parser for a new parsing domain with relatively little effort (the features we chose are extremely simple and certainly could be improved on). The cross-lingual information that we used (namely, the foreign preterminal tags of the words to which our span was aligned by GIZA) did not give a significant improvement to our parser. However the goal of this work was not to make definitive statements about the value of crosslingual features in parsing, but rather to show a framework in which such crosslingual information could be easily incorporated and exploited. We believe we have provided the beginnings of one in this work, and work continues on finding more complex features that will improve performance well beyond the baseline.

## Acknowledgement

The work reported in this paper was supported by the *Deutsche Forschungsgemeinschaft* (DFG; German Research Foundation) in the Emmy Noether project PTOLEMAIOS on grammar learning from parallel corpora.

## References

Daniel M. Bikel. 2004. Intricacies of collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Eugene Charniak. 2000. A maximum entropy-inspired parser. In *NAACL*.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *ACL*.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at http://www.isi.edu/ hdaume/docs/daume04cg-bfgs.ps, implementation available at http://www.isi.edu/ hdaume/megam/, August.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Mark Johnson. 1998. PCFG models of linguistic tree representation. *Computational Linguistics*, 24:613–632.

Jonas Kuhn and Michael Jellinghaus. to appear. Multilingual parallel treebanking: a lean and flexible approach. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, Genoa, Italy.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *EMNLP*.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*.