A NOVEL PROBABILISTIC MODEL FOR LINK UNIFICATION GRAMMAR[1]

Fuliang Weng[+], Naiyong Jin[▽], Jie Meng*, and Yujia Zhu[□]

[+]Research and Technology Center
Robert Bosch Corporation
Palo Alto, CA 94304
fuliang.weng@rtc.bosch.com

[▽]Department of Computer Science
East China Normal University
Shanghai, China

[*]Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
mengj@cs.pitt.edu

[□]Computer Science Department
University of California at Irvine
Irvine, CA 92697

Abstract

Since the early 90s, there has been a rising interest in applying probabilistic models in syntactic parsing, and significant progress has been achieved. The most influential work include, among others, research using probabilistic lexicalized context free grammars in the form of production rules (or similarly, tree-adjoining grammars and its derivatives), probabilistic dependency grammars; and probabilistic constraints-based grammars [Lafferty, J., Sleator, D., and Temperley, D., 1992; Magerman, D., 1995; Collins, M, 1997; Eisner, J., 1997; Chelba and Jelinek, 1997; Ratnaparkhi, A., 1999; Charniak, E., 2000]. Based on the discussion of different methods, we propose a new bottom-up link unification parsing algorithm in the link grammar framework together with a novel probabilistic model. In addition, we also try to provide some intuitive justification for the assumptions made in the probabilistic model. Initial results in terms of bracket recall and precision on the Penn TreeBank close-test set have reached beyond 91%, which shows that our proposed probabilistic model is leading to an encouraging direction.

1. Introduction

Since the early 90s, there has been a rising interest in apply probabilistic models in natural language processing, particularly in syntactic parsing, and significant progress has been achieved. The most influential

[1] This work is mostly done while the authors were with Intel China Research Center, Intel Corp, Beijing.

work includes, among others, research using probabilistic lexicalized context free grammars in the form of production rules (or similarly, tree-adjoining grammars and its derivatives), probabilistic dependency grammars; and probabilistic constraints-based grammars.

In [Briscoe, E. and J. Carroll 1993; Magerman, D., 1995; Hermjakob and Mooney 1997; Manning and Carpenter 1997; Chelba and Jelinek 1998; Ratnaparkhi, A., 1999], English grammars are represented by context-free production rules or compiled into parsing tables. A parsing decision is made based on the parsing history, usually, the left context in the form of constructed partial and full phrases, stack, or LR states[2]. The probabilistic decision is computed through various methods, such as decision tree approach and maximum entropy model. To make the approaches work well, systematic and automatic selection of the features to be included in history becomes very important, since, on one hand, one needs to account for various linguistic phenomena, and on the other hand, one needs to avoid data-sparseness.

In [Collins, M, 1997; Eisner, J., 1997; Charniak, E., 2000], a derivation of a sentence is modeled through the successful application of context-free production rules, and its probability is computed accordingly by applying the chain rule to the joint probability of all the rules applied (generative models). The head word of each production rule is kept for accurately characterizing its behavior. To properly handle the data sparseness problem and provide a feasible training, the probability of a production rule is computed by making explicit Markovian assumption so that only limited context is included in conditional part when chain rule is applied. To refine the general approach, [Charniak, E., 2000] integrates features in a ME-like fashion.

[Lafferty, J., Sleator, D. and Temperley, D., 1992; Collins, M., 1996; Eisner, J., 1996; Chelba, C., et al, 1997] directly models dependency relations in sentences. While [Lafferty, J., Sleator, D. and Temperley, D., 1992] computes the probability of a sentence strictly following the steps in the top-down link parsing algorithm, and making. Another approach [Chelba, C., et al, 1997] tries to compute the probability of the dependency strictly from left to right in a way similar to history-based approach.

This paper describes our parsing work that directly models the dependency relations through link grammar formalism. Instead of decomposing the models based on the parsing algorithms, as described before, it focuses on the linguistic sound independence assumptions for model decomposition. The key idea is to grow regions bottom-up, similar to unification process with certain features encoded in the link names, and its probabilistic model for a unified region is computed through its two sub-regions with dependence ratios attached, which has a good intuitive explanation. Initial results on Penn Treebank close-test data have also shown encouraging sign in terms of parsing accuracy.

The paper is organized as follows. Section 2 describes a novel link unification algorithm. Section 3 gives our probabilistic model, its derivation, and its rational, as well as an automatic conversion from the tree representation into the link grammar representation for training. Then, in Section 4, we present encouraging initial parsing results on Penn Treebank close-test data set, as well as some error analysis. Section 5 concludes with comparison with other work and future directions.

2. The Link Unification Algorithm

In this paper, we introduce a new bottom-up link unification algorithm that simplifies the procedure of link analysis, nicely deals with partial parsing, and can smoothly integrate with a novel probabilistic model to be introduced in the next section.

---

[2] [Ratnaparkhi, A. 1999] uses both left and right contexts.

The original link parser works in a top-down fashion. It enumerates each disjunct of a word and tries to find a match with a disjunct from another word in consideration. Since a disjunct of a word represents a usage of that word (or a collocation), it is often that there are many different usages for one word. While the CMU link parser uses look-up in multiple pass pruning to reduce parsing complexity, the newly proposed algorithm takes advantages of the common prefix and suffix of the different usages of a word during parsing in a bottom-up fashion.

The new parsing algorithm takes the lexicon as the linguistic knowledge source and a sentence as its input. It starts by considering every neighboring word pairs, unifies them if possible, and builds up the phrase chunks until the structure for the whole sentence is complete. It returns a set of possible word chunks with the largest combined probability when the input sentence is not covered by the word usage in the specified lexicon.

Similar to CMU's link parser, for each word in lexicon, its neighborhood information (disjuncts) is represented in the form of an and-or-tree. The alternative and/or nodes in the tree allow the sharing of the same prefix and suffix in different usages of a word/phrase, where and-node indicates the ordered sequence of disjuncts/disjunct sets, and or-node reflects the alternative disjuncts for this position. We may either deeply compress and-or-tree or take a relatively flat and-or-tree, which only permits the sharing of the leftmost and rightmost remaining connectors in different disjuncts of the same word/phrase.

The word/phrase pair unification procedure in the algorithm works as follows. For any pair of neighboring words/phrases, the link unification algorithm tries to see whether any leftmost disjunct node of the right word/phrase can match any rightmost disjunct node of the left word/phrase to form a partially matched disjunct set for the connected word region, and this set of partially matched disjuncts is again represented in the form of an and-or-tree. As before, the resulting and-or-tree can be compressed for removing the redundancy in the prefix and suffix, and verified for legitimacy of all its disjuncts. This operation is called link unification (or ***lunification***), since it is similar to the unification process.

Because of the compression in the unification procedure, the number of link unification operations to be performed is reduced, hence the speed of parsing can be increased proportionally. The pseudo-code of the link unification parsing algorithm works as follows:

1. Get input word string W
2. Look up lexicon for the disjuncts of every word w_i in the form of and-or-tree, place it in $unified[i][i+1]$
3. for i = 2, …, n do {
4. for j = 0, …, $n - i$ do {
5. for k = j +1, …, $j + i - 1$ do {
6. if (unify($unified[j][k]$, $unified[k][j+i]$) is not NULL) then
7. add the unified result to the top or-node of $unified[j][j+i]$
8. }
9. }
10. }

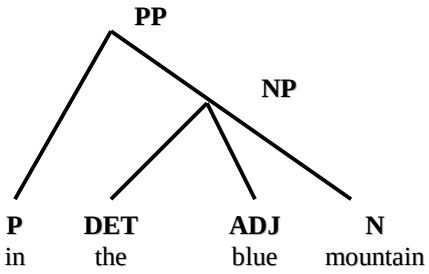
In the above algorithm, $unified[i][j]$ stores unified word strings from position i to position j, and its remaining disjuncts towards outside of this region are represented in and-or-tree form. Function unify(tree1,

tree2) tries to unify two consecutive phrasal chunks represented by tree1 and tree2, and returns the remaining disjuncts towards outside of this region in an and-or-tree.

3. The Probabilistic Model for the Lunification Algorithm

To improve the performance of the link parser, we made a few important modifications on the probabilistic aspects. First, we derive the disjuncts of a word directly from the Penn Treebank and collect statistics accordingly. Second, we use a novel probabilistic model during the lunification process.

The derivation process of the disjuncts for each word is as follows. Let's explain this through an example in Fig. 1. Assuming PP(in)→P(in) NP(mountain) is a single layer in a phrase structure tree, where P and NP are non-terminals/pre-terminals, "in" and "mountain" are the corresponding heads of the non-terminals/pre-terminals, and "in" is the head word for PP. We derive one disjunct for word "in" from this level in the tree: +PP#P#NP#1 connecting NP from right. We also derive one disjunct for word "mountain" from this level in the tree: -PP#P#NP#1 connecting P from left. In this notation, the sign symbol – and + indicate the leftward or rightward disjuncts, and # is the delimiter. In +PP#P#NP#1, PP is the parent node in the sub-tree, and NP is to the right of P. Similarly, in -PP#P#NP#1, symbol PP is the parent node in that sub-tree, and P is a constituent to the left of constituent NP. The last symbol 2 (1) in the strings means that the head is on the right (left) side of the link. For the sub-tree with NP as its root in Fig. 1, we can derive additional word disjuncts. For probabilistic models, their statistics can be collected in the Treebank by traversing the trees. This notation and conversion are similar in spirit to [Collins, 1996], where, however, only base NPs were processed.



Rules:
PP(in) → P(in) NP(mountain)
NP(mountain) → DET(the) ADJ(blue) N(mountain)
Word disjuncts:
In: (+PP#P#NP#1)
The: (+NP#DET#N#2)
Blue: (+NP#ADJ#N#2)
Mountain: (-NP#ADJ#N#2 -NP#DET#N#2 -PP#P#NP#1)

1a) The original parse tree. 1b) The converted word disjuncts.

Fig. 1. Converting parse trees to word disjuncts of Link Grammar.

The probabilistic model for the Link Grammar is defined as follows. Let's denote $\overline{L}_{i,j}$ as the partial linkage covering the region between word i and word j, and $L_{i,j}$ is a triple of ($l_{i,j}, w^{i,k}, w^{k+1,j}$) where $l_{i,j}$ is the last link that connects its two head words $w^{i,k}$ and $w^{k+1,j}$, representing the two sub-regions (i,k) and $(k+1, j)$ to be unified.

$$P(\overline{L}_{i,j}) = P(\overline{L}_{i,k}, \overline{L}_{k+1,j}, L_{i,j})$$

$$= P(\overline{L}_{i,k}, \overline{L}_{k+1,j} \mid L_{i,j})\ P(L_{i,j})$$

(Conditional independence assumption, given $L_{i,j}$)

$$= P(\overline{L}_{i,k} \mid L_{i,j})\ P(\overline{L}_{k+1,j} \mid L_{i,j})\ P(L_{i,j})$$

$$= \frac{P(\overline{L}_{i,k})P(L_{i,j} \mid \overline{L}_{i,k})}{P(L_{i,j})}\ \frac{P(\overline{L}_{k+1,j})P(L_{i,j} \mid \overline{L}_{k+1,j})}{P(L_{i,j})}\ P(L_{i,j})$$

$$= P(\overline{L}_{i,k})\ P(\overline{L}_{k+1,j})\ \frac{P(L_{i,j} \mid \overline{L}_{i,k})P(L_{i,j} \mid \overline{L}_{k+1,j})}{P(L_{i,j})}$$

(1st order Markov assumption)

$$= P(\overline{L}_{i,k})\ P(\overline{L}_{k+1,j})\ \frac{P(L_{i,j} \mid L_{i,k})P(L_{i,j} \mid L_{k+1,j})}{P(L_{i,j})} \qquad (1)$$

$$= P(\overline{L}_{i,k})\ P(\overline{L}_{k+1,j})\ P(L_{i,j})\ \frac{P(L_{i,k}, L_{i,j})}{P(L_{i,k})P(L_{i,j})}\ \frac{P(L_{k+1,j}, L_{i,j})}{P(L_{k+1,j})P(L_{i,j})}$$

That is:

$$P(\overline{L}_{i,j}) = P(\overline{L}_{i,k})P(\overline{L}_{k+1,j})P(L_{i,j})\frac{P(L_{i,k}, L_{i,j})}{P(L_{i,k})P(L_{i,j})}\frac{P(L_{k+1,j}, L_{i,j})}{P(L_{k+1,j})P(L_{i,j})} \qquad (2)$$

Therefore, the probability for large region (i, j) is computed through its two sub-regions, i.e. (i,k) and $(k+1, j)$, and the last link, with an adjustment of two symmetrical ratios. Each ratio specifies the degree of the dependency of two consecutively built links. To look at the equation from an information-theoretic viewpoint, let's take a log to both sides of equation (2). Then, we get the following equation.

$$[-\log P(\overline{L}_{i,j})] = [-\log P(\overline{L}_{i,k})] + [-\log P(\overline{L}_{k+1,j})] + [-\log P(L_{i,j})]$$
$$-\log\frac{P(L_{i,k}, L_{i,j})}{P(L_{i,k})P(L_{i,j})} - \log\frac{P(L_{k+1,j}, L_{i,j})}{P(L_{k+1,j})P(L_{i,j})}$$

So, the left side is the description length for the whole region (i, j), the items on the right side of the first line is the sum of the description lengths of the two sub-regions plus the length for the current link, and the second line is the mutual information of the *representatives* of the smaller regions and the unified region. So, an intuitive explanation for this equation is as follows.

The description length for the unified region is the sum of the description length for its sub-regions and the final link, removing the redundancy of the mutual information between the representatives of the sub-regions and the representative of the unified region.

This interpretation sheds some light on possible improvements to the model in the future, i.e., to make this model more accurate, we should consider other representative features for the model, possibly a wider context.

We now list the boundary cases for equation (2), i.e., when one or both sub-regions are actually single words, here is the derivation.

Case 1: One side of the link is a word, while this other side is a linked chunk. Assume the right side is a word, we have:

$$P(\overline{L}_{i,j}) = P(\overline{L}_{i,j-1})\ P(w^{k+1,j})\ \frac{P(L_{i,j} \mid L_{i,j-1})P(L_{i,j} \mid w^{k+1,j})}{P(L_{i,j})}$$

$$= P(\overline{L}_{i,j-1})\ P(L_{i,j} \mid L_{i,j-1})\ \frac{P(L_{i,j} \mid w_j)P(w^{k+1,j})}{P(L_{i,j})}$$

$$= P(\overline{L}_{i,j-1})\ P(L_{i,j} \mid L_{i,j-1})$$

The last equation is because $P(L_{i,j} \mid w^{k+1,j})P(w^{k+1,j}) = P(L_{i,j}, w^{k+1,j}) = P(L_{i,j})$. So,

$$P(\overline{L}_{i,j}) = P(\overline{L}_{i,j-1})P(L_{i,j} \mid L_{i,j-1}) \quad (3)$$

When we consider a degenerated case, i.e., for a right-branching tree with uniform links, the probability for the whole sentence has the following form:

$$P(W_{1,n}) = P(w_1)P(w_x \mid w_1)P(w_3 \mid w_1, w_2)....P(w_n \mid w_1, w_{n-1})$$

where all the links are removed because of the redundancy.

Case 2: Both sides of the link are words. This implies that these two words must be adjacent. The following equation can be easily derived from (3).

$$P(\overline{L}_{i,i+1}) = P(L_{i,i+1}) = P(l_{i,i+1}, w_i, w_{i+1}) \quad (4)$$

For practical reasons such as robust estimation of the parameters, we need to make further approximation to equation (1). For the sake of simplicity, assume that the head word for sub-region (i,k) is $w^{i,k}$ (i.e., $w^{i,l}$) and it is also the left end of link $l_{i,k}$, its right end is $w^{l+1,k}$ (we can do the same if the right end is the head word). For sub-region $(k+1, j)$, similarly, we assume that $w^{k+1,r}$ is on the left end of link $l_{k+1,j}$ while $w^{k+1,j}$ (i.e., $w^{r+1,j}$) is on its right end and is the head word for the sub-region. Therefore, for sub-region (i, j), the two ends of link $l_{i,j}$ are $w^{i,k}$ and $w^{k+1,j}$.

$$P(L_{i,j} \mid L_{i,k}) = P(l_{i,j}, w^{i,k}, w^{k+1,j} \mid l_{i,k}, w^{i,k}, w^{l+1,k}) = P(l_{i,j}, w^{k+1,j} \mid l_{i,k}, w^{i,k}, w^{l+1,k})$$

$$= P(l_{i,j} \mid l_{i,k}, w^{i,k}, w^{l+1,k})\ P(w^{k+1,j} \mid l_{i,k}, l_{i,j}, w^{i,k}, w^{l+1,k})$$

$$= P(l_{i,j} \mid l_{i,k}, w^{i,k})\ P(w^{k+1,j} \mid l_{i,k}, l_{i,j}, w^{i,k}) \quad \text{(Markov assumption)} \quad (5)$$

Similarly, we can approximate the other half.

$$P(L_{i,j} \mid L_{k+1,j}) = P(l_{i,j} \mid l_{k+1,j}, w^{k+1,j})\ P(w^{i,k} \mid l_{i,j}, w^{k+1,j})$$

As in other probabilistic models proposed for lexicalized CFGs, due to the data sparseness, we need to estimate the cases that do not occur in the training data. One commonly used technique is the interpolation. Take (5) as an example, we can get the following equations, where *T(w)* is the tag for word *w*.

$$P^*(l_{i,j} \mid l_{i,k}, w^{i,k}) = \lambda_1^1 * P(l_{i,j} \mid l_{i,k}, w^{i,k}) + \lambda_2^1 * P(l_{i,j} \mid l_{i,k}) + \lambda_3^1 * P(l_{i,j}) \quad (6)$$

$$P^*(w^2 \mid l_{i,j}, w^1) =$$

$$\lambda_1^2 * P(w^2 \mid l_{i,j}, w^1) + \lambda_2^2 * P(T(w^2) \mid l_{i,j}, T(w^1)) + \lambda_3^2 * P(w^2 \mid l_{i,j}) + \lambda_4^2 * P(T(w^2) \mid l_{i,j}) \quad (7)$$

To see how this model differs from other models, such as [Collins, M., 1997; Charniak, E., 2000], let's examine the first assumption made in the model, the conditional independence assumption. That is,

$$P(\overline{L}_{i,k}, \overline{L}_{k+1,j} \mid L_{i,j}) = P(\overline{L}_{i,k} \mid L_{i,j})\ P(\overline{L}_{k+1,j} \mid L_{i,j})$$

It is equivalent to the following equation:

$$P(\overline{L}_{k+1,j} \mid L_{i,j}, \overline{L}_{i,k}) = P(\overline{L}_{k+1,j} \mid L_{i,j}) = P(\overline{L}_{k+1,j} \mid l_{i,j}, w^{i,k}, w^{k+1,j})$$

This is essentially to say that the model is to predict the right child based on the left/right head, the current head, and the relation between the left/right head and the current head. Because $l_{i,j}$ encodes the information of non-terminals for both heads, which is somewhat close to Charniak's 1st order Markov model. Notice that each disjunct contains information beyond single level context free rules. This, on one hand, makes the parser more powerful, while on the other hand the current probabilistic model doesn't fully utilize the wider contextual features encoded in disjuncts, which can be further refined in the future, as pointed out as well from the above discussion on information-theoretic interpretation of the model.

The computation for the probabilistic link unification algorithm using this model is through synchronized steps in the link unification algorithm. The details are given bellow, where n is the number of words in input word string W :

1. Get input word string W
2. Look up lexicon for the disjuncts of every word w_i in the form of and-or-tree, place it in $unified[i][i+1]$
3. for i = 2, …, n do {
4. for j = 0, …, $n - i$ do {
5. for k = $j+1$, …, $j+i-1$ do {
6. if (unify($unified[j][k]$, $unified[k][j+i]$) is not NULL) then
7. $$p(j, j+i) = p(j,k)\,p(k, j+i)\,p_l(L_{j,j+i}) \frac{p_{l,l}(L_{j,k}, L_{j,j+i})}{p_l(L_{j,k})\,p_l(L_{j,j+i})} \frac{p_{l,l}(L_{k,j+i}, L_{j,j+i})}{p_l(L_{k,j+i})\,p_l(L_{j,j+i})}$$
8. add the unified result to the top or-node of $unified[j][j+i]$
9. }
10. }
11. }
12. Viterbi search the best path in score chart $P(0,n)$.

In the above algorithm, $unified[i][j]$ stores unified word strings from position i to position j , and its remaining disjuncts towards outside of this region are represented in and-or-tree form. $p(i, j)$ is the array for probability $P(i, j)$, $p_l(L_{i,j})$ for link probability $P(L_{i,j})$, and $p_{l,l}(L_1, L_2)$ for $P(L_1, L_2)$. Function unify(tree1, tree2) tries to unify two consecutive phrasal chunks represented by tree1 and tree2, and returns the remaining disjuncts towards outside of this region in an and-or-tree.

4. Experiments

To evaluate the lunification algorithm and the new probabilistic model, we conducted a few experiments on the WSJ part of the English Penn Treebank. In order to compare with other results, sections 2-21 are used for training. We first convert the Treebank to the Linkbank so to remove gaps (such as *NONE*) for easier processing. While currently, the Linkbank is still in a parsing tree format, we plan to make it a representation that directly encodes link dependency information of each word in all the Treebank sentences. The link & word statistics are computed on the Linkbank. Because the output of the lunification algorithm is the linkage(s), similar to the Link Parser, a mapping procedure is constructed to map the linkage to the phrase structure representation for bracket evaluation.

The interpolation weights for smoothing the models in equation (6,7) are selected by using Section 24 of Penn Treebank WSJ data.

To examine the effect of our model, we use Section 18 as our test data, and further assume that its sentences have correct tags labeled. That is, we perform a close-test on Section 18. Section 18 has 1806 sentences of about 40,000 words. Using the un-smoothed model, we obtained the link recall and precision of 93.12% and 93.11%. On the bracketing test, we obtained 91.42% and 91.21% for sentences, and 92.18% and 91.75% for sentences of shorter than 40 words. Using the smoothed model, we obtained the link recall and precision of 90.17% and 90.16%. On the bracketing test, we obtained 87.86% and 85.93% for all the sentences, and 88.58% and 86.43% for sentences of shorter than 40 words. These numbers, though not fully comparable, are approaching to the systems reported in [Collins, 1997; Ratnaparkhi, 1999; Charniak, 2000], which are quite encouraging. We are going to refine our model to conduct experiments on Penn Treebank WSJ open test sentences.

We also performed an error analysis on the close-test data set. While some bracketing errors are caused by the conversion from linkages to brackets, the major sources of the errors can be classified into three categories: 1) the flat structure problem; 2) the reverse structure problem; and 3) the long distance attachment confusion problem. The typical example of category 1) is the case where noun-noun compounds have only one level structure in Penn Treebank and no internal structure within the compounds is given there. For the second category, there are significant amount of OVS or OSV in WSJ that the current system does not handle properly. These sentences usually have the patterns “XXX, said Mr. ZZZ”. For the third category, it is usually the case that the sentences are long with various conjunctions. We will fix these problems in near future.

5. Conclusion

In this paper, we present a new bottom-up link unification parsing algorithm for link/dependency grammar, the lunification algorithm, and a novel probabilistic model that tries to directly capture the collocation of head words and their links. The bottom-up approach picks up the best path (the one with the highest probability) from found phrases/phrasal chunks through Viterbi search. The proposed probabilistic model decomposes a linkage model covering a large region into two smaller ones. It has an interpretation that matches well with both linguistic intuition and information theory. This decomposition also allows a smooth integration with lower level processing, and it can be well suitable to the integration with word segmentation algorithm for Chinese and other oriental languages. In addition, the proposed training procedure and the parsing approach reflect the connection among the production rule based model, the dependency grammar

based model, and the link grammar based model. Although still at its early stage, the initial results have shown a quite encouraging sign in terms of parsing accuracy.

[Eisner, J., 1996]'s dependency models only capture the dependency between words but ignore the types of the dependency (bare-bone dependency structures). In [Eisner, J., 1996]'s model C, the closest one to ours among his three models, the score (or probability) for a combined region is computed through its two smaller regions multiplied with the probability of the covering link, i.e., score(C)=score(A)score(B)Pr(covering link). [Collins, M., 1996] gives a bi-gram lexical dependency model for baseNPs, which is similar with the current approach in terms of converting the parse Treebank to the dependency links. However, in [Collins, M., 1996], the model assumes that the dependency links are independent in the process of decomposing the probabilistic model for baseNPs. This newly proposed approach differentiates itself from the above two with a model that directly models the interdependency among consecutive dependency links. The interdependency is nicely captured by the two dependency ratios in equation (2).

[Lafferty, J., Sleator, D. and Temperley, D., 1992; Chelba C., et al, 1997] directly model dependency relations in sentences. While [Lafferty, J., Sleator, D. and Temperley, D., 1992] computes the probability of a sentence following the steps in the top-down link parsing algorithm, and making corresponding Markovian assumptions for practical purposes. Another approach [Chelba, C. et al, 1997] tries to compute the probability of the dependency strictly from left to right in a way similar to history-based approach, but using Maximum Entropy approach to combine different features.

In a different framework from above, English grammars are represented by context-free production rules [Briscoe, E. and J. Carroll 1993; Magerman, D., 1995; Hermjakob and Mooney 1997; Manning and Carpenter 1997; Chelba and Jelinek 1998; Ratnaparkhi, A., 1999]. A parsing decision is made based on the parsing history, usually, the left context in the form of constructed partial and full phrases, stack, or LR states. The probabilistic decision is computed through various methods, including decision tree approach and maximum entropy model. To make the approaches feasible, systematic and automatic selection of the features to be included in history becomes very important, since, on one hand, one needs to account for various linguistic phenomena, and on the other hand, one needs to avoid data-sparseness. Furthermore, as pointed out by [Eisner, J., 1996], probability models derived from parsers sometimes focus on incidental properties, which might not be desirable. Our approach, on the contrast, is more independent of a particular parsing algorithm, i.e., the model can be either taken as a generative model, or as a comprehensive model.

[Collins, M., 1997]'s probabilistic models start from generating the head in a production rule, then given the head, the approach generates its right modifiers and left modifiers in the rule consecutively. The distance between the head and its modifier is also included in the conditional part of the probability. [Charniak, E. 2000] proposed a generative model that also starts from lexicalized CFG. His approach leans towards more on the flexibility of incorporating new features and tries to ease the difficulty in isolating the conditional events as what has been done by Maximum-Entropy approaches. However, as he pointed out, the integration of the new features in the model is not strictly probabilistic. In contrast, our approach tries to directly model the dependency relation and tries to capture not only the words but also the dependency relations among them with a solid probabilistic formation.

Our future work will include improving and finishing our probabilistic lunifier for an open-test set, integrating wider contextual features in the models naturally, and establishing connections with lower level language processing. Additional tunings and improvements on the models, such as the ones used in [Collins, M., 2000; Johnson, M. et al 1999], will also be considered.

Acknowledgement: The authors would like to thank the anonymous reviewers for useful comments in improving this paper.

Reference

1. Briscoe, E. and J. Carroll (1993) "Generalised Probabilistic LR Parsing of Natural Language (Corpora) with Unification-based Grammars", Computational Linguistics, 19(1). 25-59.

2. Charniak, E. (2000). "A Maximum-Entropy-Inspired Parser", NAACL'2000, Seattle, Washington.

3. Chelba, C., et al. (1997). "Structure and Performance of a Dependency Language Model", Proceedings of Eurospeech'1997, Greece.

4. Chelba, C. and Jelinek, F. (1998). "Exploiting Syntactic Structure for Language Modeling", Proceedings of the 35-th Annual Meeting of the Association for Computational Linguistics, COLING-ACL'1998.

5. Collins, M (1996). "A New Statistical Parser Based on Bigram Lexical Dependencies", Proceedings of the 34-th Annual Meeting of the Association for Computational Linguistics, ACL'1996.

6. Collins, M. (1997). "Three Generative, Lexicalized Models for Statistical Parsing", Proceedings of the 35-th Annual Meeting of the Association for Computational Linguistics, ACL, Madrid, Spain, 1997.

7. Collins, M. (2000). "Discriminative Reranking for Natural Language Parsing", Proceedings of International Conference on Machine Learning, Stanford University, CA, 2000.

8. Eisner, J. (1996). "Three New Probabilistic Models for Dependency Parsing: An Exploration", COLING-96, 1996.

9. Eisner, J. (1997). "Bilexical Grammars and a Cubic-Time Probabilistic Parser", Proceedings of 5-th IWPT by ACL/SIGPARSE, Cambridge, MA, 1997.

10. Hermjakob, U. and Mooney, R.J. (1997). Learning parse and translation decisions from examples with rich context", Proceedings of the 35-th Annual Meeting of the Association for Computational Linguistics, ACL, Madrid, Spain, 1997.

11. Johnson, M., Geman, S., Canon, S., Chi, Z., and Riezler, S. "Estimators for Stochastic 'Unification-Based' Grammars", ACL'1999.

12. Lafferty, J., Sleator, D., and Temperley, D. (1992) "Grammatical Trigrams: a Probabilistic Model of Link Grammar", in AAAI Conf. on Probabilistic Approaches to Natural Language, 1992.

13. Magerman, D. (1995) "Statistical Decision-Tree Models for Parsing", Proceedings of the 33-rd Annual Meeting of the Association for Computational Linguistics, ACL'1995, 276-283.

14. Manning, C. and Carpenter, B. (1997) "Probabilistic Parsing Using Left Corner Language Models", Proceedings of 5-th IWPT by ACL/SIGPARSE, Cambridge, MA, 1997.

15. Ratnaparkhi, A. (1999) "Learning to Parse Natural Language with Maximum Entropy Models", Machine Learning 34, pp151-175, Kluwer Academic Publishers, 1999.

16. Sleator, D. and Temperley, D (1993) "Parsing English with a Link Grammar", Proceedings of 3rd IWPT by ACL/SIGPARSE, Tilburg, The Netherlands, 1993.