Reestimation and Best-First Parsing Algorithm for Probabilistic Dependency Grammars

Seungmi Lee and Key-Sun Choi
Dept. of Computer Science
Korea Advanced Institute of Science and Technology
373-1, Kusung-Dong, YuSung-Gu, Taejon, 305-701, KOREA
e-mail: {leesm,kschoi}@world.kaist.ac.kr

Abstract

This paper presents a reestimation algorithm and a best-first parsing (BFP) algorithm for probabilistic dependency grammars (PDG). The proposed reestimation algorithm is a variation of the inside-outside algorithm adapted to probabilistic dependency grammars. The inside-outside algorithm is a probabilistic parameter reestimation algorithm for phrase structure grammars in Chomsky Normal Form (CNF). Dependency grammar represents a sentence structure as a set of dependency links between arbitrary two words in the sentence, and can not be reestimated by the inside-outside algorithm directly. In this paper, non-constituent objects, **complete-link** and **complete-sequence** are defined as basic units of dependency structure, and the probabilities of them are reestimated. The reestimation and BFP algorithms utilize CYK-style chart and the non-constituent objects as chart entries. Both algorithms have $O(n^3)$ time complexities.

1 Introduction

There have been many efforts to induce grammars automatically from corpus by utilizing the vast amount of corpora with various degrees of annotations. Corpus-based, stochastic grammar induction has many profitable advantages such as simple acquisition and extension of linguistic knowledges, easy treatment of ambiguities by virtue of its innate scoring mechanism, and fail-soft reaction to ill-formed or extra-grammatical sentences.

Most of corpus-based grammar inductions have concentrated on phrase structure grammars (Black, Lafferty, and Roukos, 1992, Lari and Young, 1990, Magerman, 1994). The typical works on phrase structure grammar induction are as follows(Lari and Young, 1990, Carroll, 1992b): (1) generating all the possible rules, (2) reestimating the probabilities of rules using the inside-outside algorithm, and (3) finally finding a stable grammar by eliminating the rules which have probability values close to 0. Generating all the rules is done by restricting the number of nonterminals and/or the number of the right hand side symbols in the rules and enumerating all the possible combinations. Chen extracts rules by some heuristics and reestimates the probabilities of rules using the inside-outside algorithm (Chen, 1995). The inside-outside algorithm learns a grammar by iteratively adjusting the rule probabilities to minimize the training corpus entropy. It is extensively used as reestimation algorithm for phrase structure grammars.

Most of the works on phrase structure grammar induction, however, have partially succeeded. Estimating phrase structure grammars by minimizing the training corpus entropy does not lead to the desired grammars which is consistent with human intuitions (de Marcken, 1995). To increase the correctness of the learned grammar, Marcken proposed to include lexical information to the phrase structure grammar. A recent trend of parsing is also to include lexical information to increase the correctness (Magerman, 1994, Collins, 1996). This means that the lack of lexical information in phrase structure grammar is a major weak point for syntactic disambiguation. Besides the lack of lexical information, the induction of phrase structure grammar may suffer from structural data sparseness with medium sized training corpus. The structural data sparseness means the lack of information on the grammar rules. An approach to increase the correctness of grammar induction is to learn a grammar from a tree-tagged corpus or bracketed corpus (Pereira and Schabes, 1992, Black, Lafferty, and Roukos, 1992). But the construction of vast sized tree-corpus or bracketed corpus is very labour-intensive and manual construction of such corpus may produce serious inconsistencies. And the structural-data sparseness problem still remains.

The problems of structural-data sparseness and lack of lexical information can be lessened with PDG. Dependency grammar defines a language as a set of dependency relations between any two words. The basic units of sentence structure in DG, the dependency relations are much simpler than the rules in phrase structure grammar. So, the search space of dependency grammar may be smaller and the grammar induction may be less affected by the structural-data sparseness. Dependency grammar induction has been studied by Carroll (Carroll, 1992b, Carroll, 1992a). In the works, however, the dependency grammar was rather a restricted form of phrase structure grammars. Accordingly, they extensively used the inside-outside algorithm to reestimate the grammar and have the same problem of structural-data sparseness.

In this paper, we propose a reestimation algorithm and a best-first parsing algorithm for PDG. The reestimation algorithm is a variation of the inside-outside algorithm adapted to PDG. The inside-outside algorithm is a probabilistic parameter reestimation algorithm for phrase structure grammars in CNF and thus can not be directly used for reestimation of probabilistic dependency grammars. We define non-constituent objects, **complete-link** and **complete-sequence** as basic units of dependency structure. Both of reestimation algorithm and best-first parsing algorithm utilize a CYK-style chart and the non-constituent objects as chart entries. Both algorithms have $O(n^3)$ time complexities.

The rest of the paper is organized as follows. Section 2 defines the basic units and describes best-first parsing algorithm. Section 3 describes the reestimation algorithm. Section 4 shows the experimental results of reestimation algorithm on Korean and finally section 5 concludes this paper.

2 PDG Best First Parsing Algorithm

Dependency grammar describes a language with a set of head-dependent relations between any two words in the language. Head-dependent relations represent specific relations such as modifiee-modifier, predicate-argument, etc. In general, a functional role is assigned to a dependency link and specifies the syntactic/semantic relation between the head and the dependent. However, in this paper, we use the minimal definition of dependency grammar with head-dependent relations only. In the future we will extend our dependency grammar into one with functions of dependency links.

A dependency tree of a n-word sentence is always composed of n-1 dependency links. Every word in the sentence must have its head, except the word which is the head of the sentence. In a dependency tree, crossing links are not allowed.

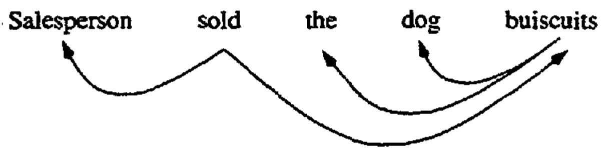


Figure 1: Dependency tree: link representations

Figure 1 shows a dependency tree as a hierarchical representation and a link representation respectively. In both, the word "sold" is the head of the sentence.

Here, we define the non-constituent objects, *complete-link* and *complete-sequence* which are used in PDG reestimation and BFP algorithms. A set of dependency links constructed for word sequence $w_{i,j}$ is defined as complete-link if the set satisfies following conditions:

- The set has exclusively $(w_i \rightarrow w_j)$ or $(w_i \leftarrow w_j)$.
- There is neither link-crossing nor link-cycle.
- The set is composed of $j - i$ dependency links.
- Every inner word of $w_{i,j}$ must have its head and thus a link from the head.

Complete-link has directionality. It is determined by the direction of the outermost dependency relation. If the complete-link has $(w_i \rightarrow w_j)$, it is rightward, and if the complete-link has $(w_i \leftarrow w_j)$, then it is leftward. Basic complete-link is a dependency link between adjacent two words.

Complete-sequence is defined as a sequence of null or more adjacent complete-links of same direction. Basic complete-sequence is null sequence of complete-links which is defined on one word, the smallest word sequence. The direction of complete-sequence is determined by the direction of component complete-links. If the complete-sequence is composed of leftward complete-links, the complete-sequence is leftward, and vice versa.

Figure 2 shows abstract rightward complete-link for $w_{i,j}$, rightward complete-sequence for $w_{i,m}$, and leftward complete-sequence for $w_{m+1,j}$. Double-slashed line means a complete-sequence. Whatever the direction is, a complete-link for $w_{i,j}$ is always constructed with a dependency link between w_i and w_j, a rightward complete-sequence from i to m, and a leftward complete-sequence from j to $m + 1$, for an m between i and $j - 1$. Rightward complete-sequence is always composed with a combination of a rightward complete-sequence and a rightward complete-link. On the contrary, leftward complete-sequence is always composed with a combination of a leftward complete-link and a leftward complete-sequence. These restrictions on composition of complete-sequence is for the ease of description of algorithm. The basic complete-link and complete-sequence are also shown in the Figure 2. Following notations are used to represent the four kinds of objects for a word sequence $w_{i,j}$ and for an m from i to j-1.

- $L_r(i,j)$: rightward complete-link for $w_{i,j}$, i.e. $\{(w_i \rightarrow w_j),\ S_r(i,m),\ S_l(m+1,n)\}$
- $L_l(i,j)$: leftward complete-link for $w_{i,j}$, i.e. $\{(w_i \leftarrow w_j),\ S_r(i,m),\ S_l(m+1,n)\}$

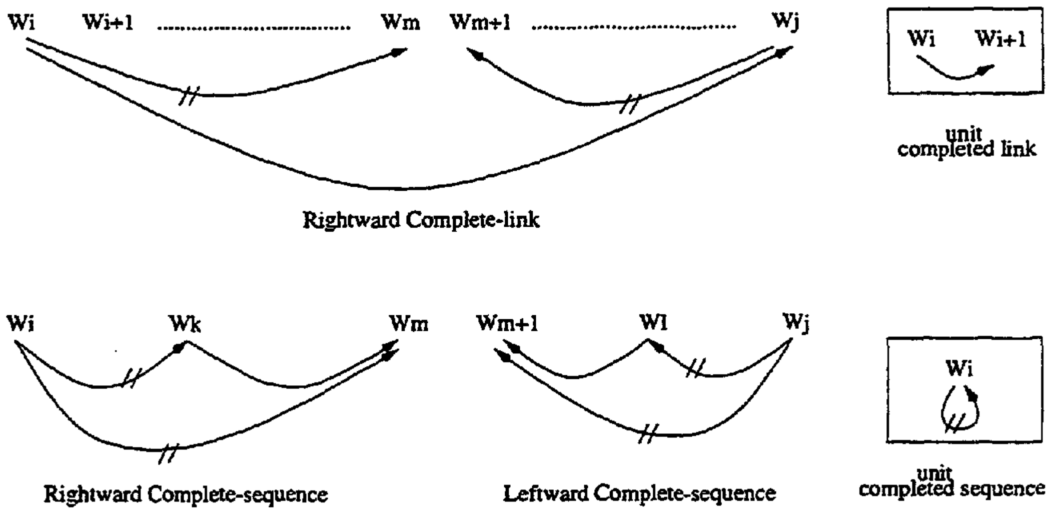


Figure 2: Abstract complete-link and complete-sequence

- $S_r(i,j)$: rightward complete-sequence for $w_{i,j}$, i.e. $\{S_r(i,m),\ L_r(m,j)\}$
- $S_l(i,j)$: leftward complete-sequence for $w_{i,j}$, i.e. $\{L_l(i,m),\ S_l(m,j)\}$

To generalize the structure of a dependency tree, we assume that there are marking tags, BOS (Begin Of Sentence) before w_1 and EOS (End Of Sentence) after w_n and that there are always the dependency links, $(w_{BOS} \rightarrow w_{EOS})$ and $(w_k \leftarrow w_{EOS})$ when w_k is the head word of the sentence. Then, by definition, any dependency tree of a sentence, $w_{i.n}$ can be uniquely represented with either a $L_r(BOS, EOS)$ or a $S_l(1, EOS)$ as depicted in Figure 3. This is because $L_r(BOS, EOS)$ for any sentence is always composed of null $S_r(BOS, BOS)$ and $S_l(1, EOS)$. The head of a dependency tree w_k can be found in the rightmost $L_l(k, EOS)$ of $S_l(1, EOS)$.

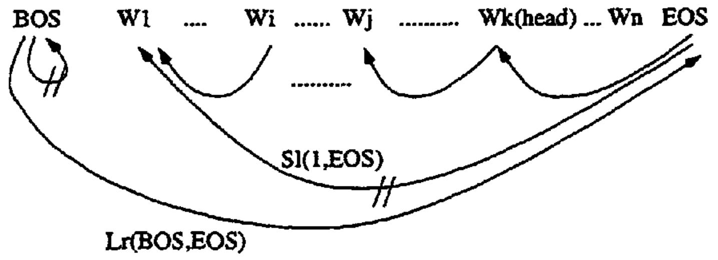


Figure 3: Abstract dependency tree of a sentence

The probability of each object is defined as follows.

$$p(L_r(i,j)) = p(w_i \rightarrow w_j)p(S_r(i,m))p(S_l(m+1,j))$$

$$p(L_l(i,j)) = p(w_i \leftarrow w_j)p(S_r(i,m))p(S_l(m+1,j))$$

$$p(S_r(i,j)) = p(S_r(i,m))p(L_r(m,j))$$

$$p(S_l(i,j)) = p(L_l(i,m))p(S_l(m,j))$$

The m varies from i to $j-1$ for L_l, L_r and S_r, and from $i+1$ to j for S_l. The best L_l and the best L_r always share the same m. This is because both are composed of the same sub-S_r and sub-S_l with maximum probabilities. Basis probabilities are as follows:

$$p(L_r(i,i+1)) = p(w_i \rightarrow w_{i+1})$$
$$p(L_l(i,i+1)) = p(w_i \leftarrow w_{i+1})$$
$$p(S_r(i,i+1)) = p(L_r(i,i+1))$$
$$p(S_l(i,i+1)) = p(L_l(i,i+1))$$
$$p(S_l(i,i)) = p(S_r(i,i)) = 1$$

Thus, the probability of a dependency tree is defined either by $p(L_r(BOS, EOS))$ or by $p(S_l(1, EOS)$.

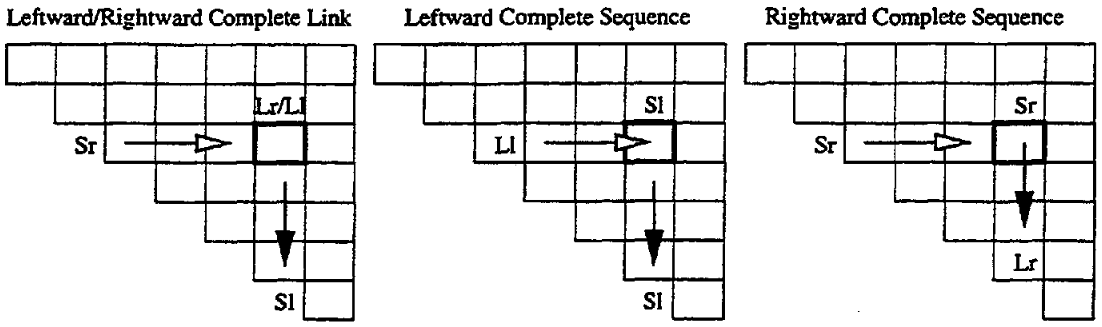

Leftward/Rightward Complete Link
Lr/Ll
Sr
Sl
Leftward Complete Sequence
Sl
Ll
Sl
Rightward Complete Sequence
Sr
Sr
Lr

Figure 4: Best first parsing entries

The PDG best-first parsing algorithm constructs the best dependency tree in bottom-up manner, with dynamic programming method using CYK style chart. It is based on complete-link and complete-sequence of non-constituent concept. The parsing algorithm constructs the complete-links and complete-sequences for substring, and merges incrementally the complete-links into larger complete-sequence and complete-sequences into larger complete-link until the $L_r(BOS, EOS)$ with maximum probability is constructed.

Eisner (Eisner, 1996) proposed an $O(n^3)$ parsing algorithm for PDG. In their work, basic unit of chart entry is *span* which is also of non-constituent concept. But, the span slightly differs from our complete-sequence and complete-link. When two adjacent spans are merged into a larger span, some conditional tests must be satisfied. In our work, best-first parsing is done by inscribing the four entries with maximum probabilities, $L_l(i,j)$, $L_r(i,j)$, $S_l(i,j)$, and $S_r(i,j)$ to each chart positions in bottom-up/left-to-right manner without any extra condition checking.

Figure 4 depicts the possible combinations of chart entries into a larger L_r, L_l, S_l, and S_r each. The sub-entries under the white-headed arrow and the sub-entries under the black-headed arrow are merged into a larger entries. The larger entries are inscribed into the bold box.

There is an exception for chart entries of *n+1*th column. In the *n+1*th column, only the $L_l(k, EOS)$ whose sub S_l is null can be inscribed. This is because there can be only one head word for a tree structure. If $L_l(k, EOS)$ whose sub S_l is not null is inscribed into the chart, the overall tree structure will have two or more heads.

The best parse is maximum $L_r(BOS, EOS)$ in the chart position $(0, n+1)$. The best parse can also be found by the maximum $S_l(1, EOS)$ because the $L_r(BOS, EOS)$ is always composed of $S_r(BOS, BOS)$ and $S_l(1, EOS)$.

The chart size is $\frac{n^2+4n+3}{2}$ for n word sentence. For four items (L_r, L_l, S_r, and S_l) of each chart position, there can be maximally n searches. Thus, the time complexity of the best-first parsing algorithm is $O(n^3)$.

3 PDG Reestimation Algorithm

For reestimation of dependency probabilities of PDG, eight kinds of chart entries are defined based on three factors: inside/outside, complete-link/complete-sequence, and leftward/rightward. In following definitions, β is for inside probability and α is for outside probability. Superscripts represent whether the object is complete-link or complete-sequence, l for complete-link and s for complete-sequence. Subscripts of β and α are for the directionality, r for rightward and l for leftward.

Complete-link Inside Probabilities: β_r^l, β_l^l Inside probability of a complete-link is the probability that word sequence $w_{i,j}$ will be generated when there is a dependency relation between w_i and w_j.

$$\begin{aligned}
\beta_r^l(i,j) &= p(w_{i,j}|L_r(i,j)) \\
&= \sum_{m=i}^{j-1} p(w_i \rightarrow w_j)\beta_r^s(i,m)\beta_l^s(m+1,j) \\
\beta_l^l(i,j) &= p(w_{i,j}|L_l(i,j)) \\
&= \sum_{m=i}^{j-1} p(w_i \leftarrow w_j)\beta_r^s(i,m)\beta_l^s(m+1,j)
\end{aligned}$$

In Figure 5, $\beta_r^l(i,j)$, the inside probability of $L_r(i,j)$ is depicted. In the left part of the

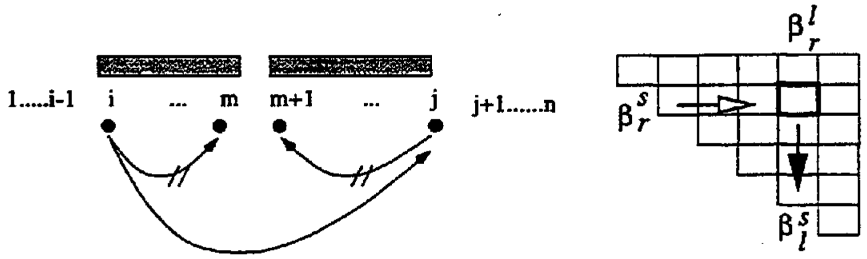


Figure 5: Rightward complete-link Inside probability

figure, the gray partitions indicate all the possible constructions of $S_r(i,m)$ and all the possible constructions of $S_l(m+1,j)$ respectively. Double-slashed links depict complete-sequences which compose the L_r together with the outermost dependency $(w_i \rightarrow w_j)$. The right part of the figure represents the chart. The bold box is the position where the β_r^l is to be inscribed. Inside probability of a complete-link is the sum of the probabilities of all the possible constructions of the complete-link. As explained in the previous section, a

$L_r(i,j)$ is composed of the dependency link between word i and word j (either $(w_i \rightarrow w_j)$ or $(w_i \leftarrow w_j)$), $S_r(i,m)$ and $S_l(m+1,j)$ for an m from i to j-1. Inside probability of $L_l(i,j)$ can be computed the same as that of $L_r(i,j)$ except for the direction of dependency link between w_i and w_j. The outermost dependency $(w_i \rightarrow w_j)$ must be replaced with $(w_i \leftarrow w_j)$. L_r and L_l are not defined on word sequence of 1 length, w_i. The unit probabilities for β_r^l and β_l^l are as follows:

$$\beta_r^l(i,i+1) = p(w_i \rightarrow w_{i+1})$$

$$\beta_l^l(i,i+1) = p(w_i \leftarrow w_{i+1})$$

$$\beta_r^l(BOS,EOS) = p(w_{1,n})$$

Any dependency tree of a sentence always has the dependency $(w_{BOS} \rightarrow w_{EOS})$ as the outermost dependency. So the $\beta_r^l(BOS,EOS)$ is the same as the sentence probability which is the sum of probabilities of all the possible parses.

Complete-sequence Inside Probabilities: β_r^s, β_l^s Inside probability of complete-sequence is the probability that word sequence $w_{i,j}$ is generated when there is $S_l(i,j)$ or $S_r(i,j)$.

$$\begin{aligned} \beta_r^s(i,j) &= p(w_{i,j}|S_r(i,j)) \\ &= \sum_{m=i}^{j-1} \beta_r^s(i,m)\beta_r^l(m,j) \\ \beta_l^s(i,j) &= p(w_{i,j}|S_l(i,j)) \\ &= \sum_{m=i+1}^{j} \beta_l^l(i,m)\beta_l^s(m,j) \end{aligned}$$

In Figure 6 and 7, the double-slashed link means complete-sequence, a sequence of null or more adjacent complete-links of same direction. A complete-sequence is composed of sub-

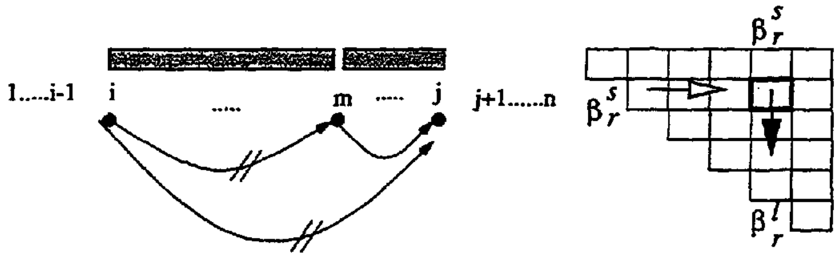


Figure 6: Rightward complete-sequence Inside probability

complete-sequence and sub-complete-link. Figure 6 depicts rightward complete-sequence for an m. The value of m varies from i to j-1. In Figure 7, S_l is composed of sub-L_l and sub-S_l. The inside probability of complete-sequence is the sum of the probabilities that the complete-sequences are constructed with. The basis for inside probabilities of complete-sequence are as follows.

$$\beta_r^s(i,i) = \beta_l^s(i,i) = 1$$

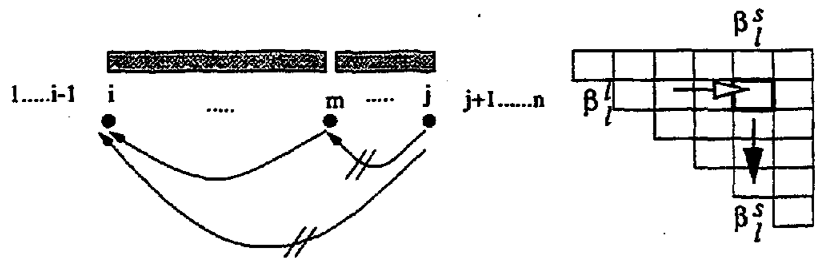


Figure 7: Leftward complete-sequence Inside probability

$$\beta^s_l(1, EOS) = \beta^s_r(BOS, EOS) = p(w_{1,n})$$

$$\beta^s_r(i, i+1) = p(L_r(i, i+1)) = p(w_i \rightarrow w_{i+1})$$

$$\beta^s_l(i, i+1) = p(L_l(i, i+1)) = p(w_i \leftarrow w_{i+1})$$

Because *n+1*th word, w_{EOS} can not be a dependent of any other word, $\beta^l_r(k, EOS)$ or $\beta^s_r(k, EOS)$ for k from 1 to n is not computed. And because there can be only one head of a tree, w_{EOS} must be head of only one word. Thus, in computation of $\beta^s_l(x, EOS)$ and $\beta^l_l(x, EOS)$, only the L_ls whose sub S_l is null are considered.

Complete-link Outside Probability: α^l_r, α^l_l This is the probability of producing the words before i and after j of a sentence while complete-link(i,j) generates $w_{i,j}$.

$$\begin{aligned}
\alpha^l_r(i,j) &= p(w_{1,i-1}, L_r(i,j), w_{j+1,n}) \\
&= \sum_{v=1}^{i} \alpha^s_r(v,j)\beta^s_r(v,i) \\
\alpha^l_l(i,j) &= p(w_{1,i-1}, L_l(i,j), w_{j+1,n}) \\
&= \sum_{h=j}^{n} \alpha^s_l(i,h)\beta^s_l(j,h)
\end{aligned}$$

Figure 8 and 9 depict the cases of L_r and L_l to become a substructure of larger S_r and

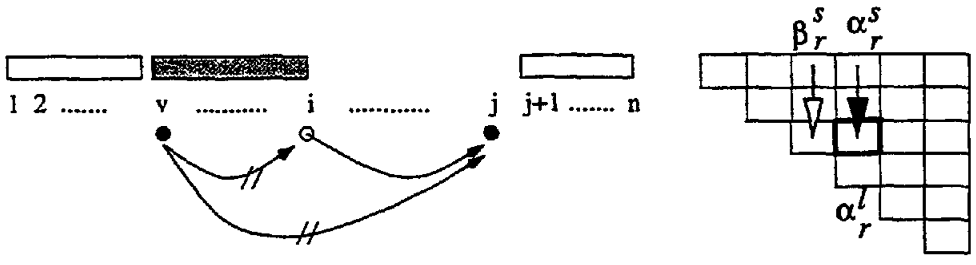


Figure 8: Rightward complete-link Outside probability

S_l respectively. In Figure 8, the outside probability of L_r which is inscribed in the bold box is computed by summing the products of the inside probabilities in the boxes under the white-headed arrow and the outside probabilities in the boxes under the black-headed

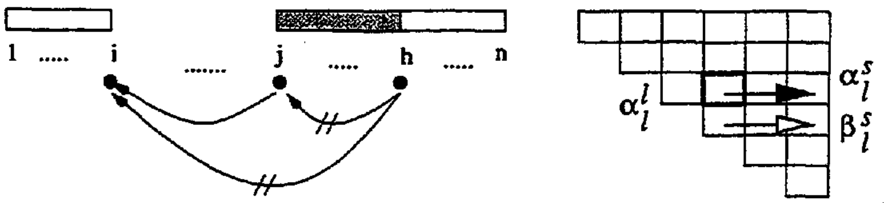


Figure 9: Leftward complete-link Outside probability

arrow. Likewise, in Figure 9, the outside probability of L_l in the bold box is computed by summing all the products of the inside probabilities under the white-headed arrow and the outside probabilities under the black-headed arrow each. This is because, in parsing, the subentries under the white-headed arrows and L_r/L_l in the bold boxes are merged into larger entries which are to be inscribed in the boxes under the black-headed arrows. Basis probability for complete-link outside probability is as follow.

$$\alpha_r^l(BOS, EOS) = 1$$

$\alpha_r^l(k, EOS)$ is always 0, for $k = 1, n+1(EOS)$ because w_{EOS} can not be a dependent of any other word.

Complete-sequence Outside Probabilities: α_r^s, α_l^s This is the probability of producing word sequence $w_{1,i-1}$ and $w_{j+1,n}$ while words i through j construct a complete-sequence.

$$\begin{aligned}
\alpha_r^s(i,j) &= p(w_{1,i-1}, S_r(i,j), w_{j+1,n}) \\
&= \sum_{h=j+1}^{n} \alpha_r^s(i,h)\beta_r^l(j,h) \\
&\qquad +\alpha_r^l(i,h)\beta_l^s(j+1,h)p(w_i \rightarrow w_h) \\
&\qquad +\alpha_l^l(i,h)\beta_l^s(j+1,h)p(w_i \leftarrow w_h)
\end{aligned}$$

In the above expression, The first term is for the construction of larger $S_r(i,h)$ from the combination of $S_r(i,j)$ and its adjacent $L_r(j,h)$. The second term means the construction of larger $L_r(i,h)$ from the combination of $S_r(i,j)$, $S_l(j+1,h)$, and the dependency link from w_i to w_h. The third term is for the larger $L_l(i,h)$ from the combination of $S_r(i,j)$, $S_l(j+1,h)$ and the dependency link from w_h to w_i. The three terms in the expression are depicted in Figure 10.

$$\begin{aligned}
\alpha_l^s(i,j) &= p(w_{1,i-1}, S_l(i,j), w_{j+1,n}) \\
&= \sum_{v=1}^{i-1} \alpha_l^s(v,j)\beta_l^l(v,i) \\
&\qquad +\alpha_r^l(v,j)\beta_r^s(v,i-1)p(w_v \rightarrow w_j) \\
&\qquad +\alpha_l^l(v,j)\beta_r^s(v,i-1)p(w_v \leftarrow w_j)
\end{aligned}$$

α_l^s is the sum of all the probabilities that S_l is to become a subentry of larger entries: S_l, L_r, and L_l. The first term in the above expression is for the combination of $S_l(v,j)$ from $L_l(v,i)$ and $S_l(i,j)$. The second is for the construction of $L_r(v,j)$ from $S_r(v,i-1)$, $S_l(i,j)$,

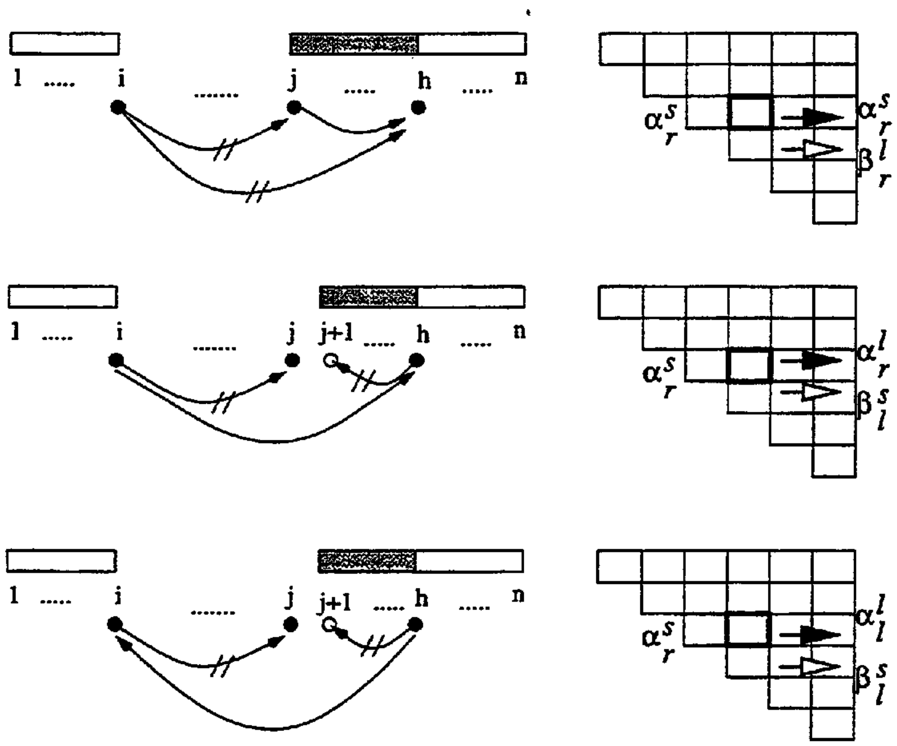


Figure 10: Rightward complete-sequence Outside probability

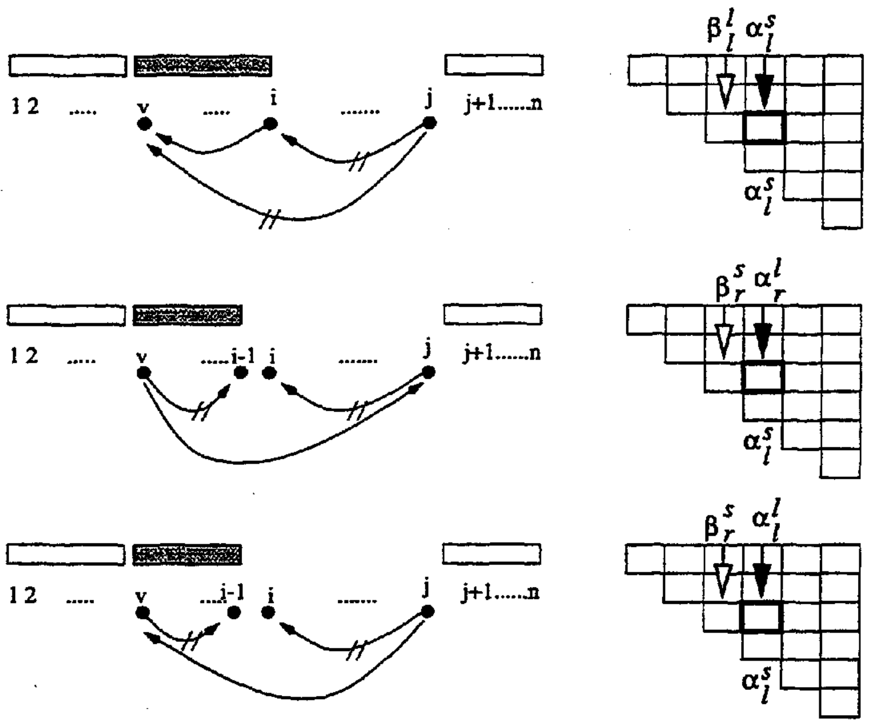
and the dependency link from w_v to w_j. The third term is for the construction of $L_l(v,j)$ from $S_r(v,i-1)$, $S_l(i,j)$, and dependency relation from w_j to w_v. The three cases are depicted in Figure 11. The basis probabilities for complete-sequence outside probabilities are as follows.

$$\alpha_r^s(BOS,EOS) = \alpha_r^l(BOS,EOS) = \alpha_l^s(1,EOS) = 1$$

The reestimation algorithm computes the inside probabilities(β_r^l, β_l^l, β_r^s, and β_l^s) inscribing them into the chart in bottom-up and left-to-right. The outside probabilities(α_r^l, α_l^l, α_r^s, and α_l^s) are computed and inscribed into the chart in top-down and right-to-left.

Training The training process is as follow;

1. Initialize the probabilities of dependency relations between all the possible word pairs.
2. Compute initial entropy.
3. Analyze the training corpus using the known probabilities,
   and recalculate the frequency of each dependency relation based on the analysis result.
4. Compute the new probabilities based on the newly counted frequencies.
5. Compute the new entropy.
6. Continue 3 through 5 until the new entropy $\approx$ previous entropy.



β_l^l α_l^s
1 2 ..... v ..... i .... j j+1......n
α_l^s
β_r^s α_r^l
1 2 ..... v .....i-1 i .... j j+1......n
α_l^s
β_r^s α_l^l
1 2 ..... v ....i-1 i .... j j+1......n
α_l^s

Figure 11: Leftward complete-sequence Outside probability

The above iteration is continued until all the probabilities are settled down or the training corpus entropy converges to the minimum. The new usage count of a dependency relation is calculated as follows. In the following expression, the O_{cc} is 1 if the dependency relation is used in the given tree and 0 otherwise.

$$\begin{aligned}
c(w_i \to w_j) &= \sum_{tree} p(tree|w_{1,n}) O_{cc}(w_i \to w_j, tree, w_{1,n}) \\
&= \frac{1}{p(w_{1,n})} \sum_{tree} p(tree, w_{1,n}) O_{cc}(w_i \to w_j, tree, w_{1,n}) \\
&= \frac{1}{p(w_{1,n})} \sum_{tree} p(tree, w_{1,n}, w_i \to w_j) \\
&= \frac{1}{p(w_{1,n})} (w_{1,n}, w_i \to w_j) \\
&= \frac{1}{p(w_{1,n})} \sum_{m=i}^{j-1} p(w_{1,n}, L_r(i,j), S_r(i,m), S_l(m+1,j)) \\
&= \frac{1}{p(w_{1,n})} \sum_{m=i}^{j-1} p(w_{1,i-1}, w_{i,m}, w_{m+1,j}, w_{j+1,n}, L_r(i,j), S_r(i,m), S_l(m+1,j)) \\
&= \frac{1}{p(w_{1,n})} \sum_{m=i}^{j-1} p(w_{1,i-1}, L_r(i,j), w_{j+1,n}) \\
&\qquad p(S_r(i,m), S_l(m+1,j)|L_r(i,j))
\end{aligned}$$

$$
\begin{aligned}
& p(w_{i,m}|S_r(i,m)) \\
& p(w_{m+1,j}|S_l(m+1,j)) \\
= \; & \frac{1}{p(w_{1,n})} \sum_{m=i}^{j-1} \alpha_r^l(i,j) p(w_i \rightarrow w_j) \beta_r^s(i,m) \beta_l^s(m+1,j) \\
= \; & \frac{1}{p(w_{1,n})} \alpha_r^l(i,j) \beta_r^l(i,j)
\end{aligned}
$$

Similarly, the usage count of $(w_i \leftarrow w_j)$, $c(w_i \leftarrow w_j)$ is $\frac{1}{p(w_{1,n})} \alpha_l^l(i,j) \beta_l^l(i,j)$.

Chart has $\frac{n^2+4n+3}{2}$ number of boxes. The reestimation algorithm computes eight items for each chart box and the computation of each item needs maximally n number of productions and summations respectively. So the time complexity of the algorithm is $O(n^3)$.

The algorithm can be used for class-based (or tag-based) dependency grammar. With the concept of word class/tag, the complexity is affected by the class/tag size due to the class/tag ambiguities of each word. In the worst case, the time needed is $8 \times t^2 \times \frac{n^3+4n^2+3n}{2}$, so the complexity will be $O(t^2 n^3)$ with respect to t, the number of classes and n, the length of input string.

4 Experimental results on Korean

We performed experiment of the reestimation algorithm on Korean language. Korean is a partially ordered language in which the head words are always placed to the right of their dependent words. Under such restriction on word order, an abstract dependency structure for Korean is depicted in Figure 12. Thus, in Korean, both of $S_r(i,j)$ and $L_r(i,j)$ are meaningless and not constructed. Only S_l, L_l and null $S_r(i,i)$ are considered. $L_l(i,j)$ is always composed with the combination of null $S_r(i,i)$ and $S_l(i+1,j)$.

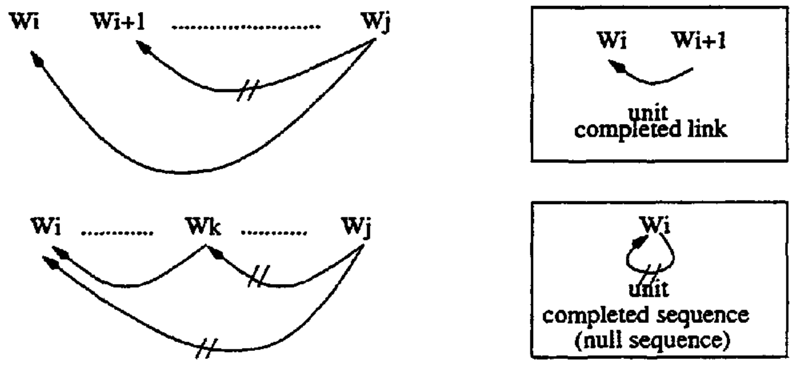


Figure 12: Abstract complete-link and complete-sequence for Korean

Korean sentence is spaced by "word-phrases" which is a sequence of content word and its functional words. In this experiment, the final part of speech(POS) of a "word-phrase" is selected as the representative POS and the inter-word-phrase dependencies are reestimated. We used 54 POS set. The initial probabilities of all the possible dependencies were set equal.

The experiment was performed on three kinds of training and test sets extracted from

Table 1: Train and test entropies

iteration	Set-1 (14,427 words)	Set-2 (40,818 words)	Set-3 (336,824 words)
1	4.439422	4.486444	4.486717
2	2.273562	2.356933	2.357249
3	2.194952	2.283601	2.289591
4	2.167079	2.258157	2.265482
...	....	....	....
13	2.139625	2.235572	2.243396
14	2.139398	2.235414	2.243232
15	2.139225	2.235297	2.243109
16	2.139093	2.235207	2.243015
17	2.138989		
18	2.138906		
	Set-1 (2,170 words)	Set-2 (4,662 words)	Set-3 (15,903 words)
test entropy	2.476484	2.150553	2.251653

KAIST corpora[1]. The convergence of entropies(bits/word) through training iterations and the test corpus entropy are shown in the table 1. Set-1 is extracted from information and communication related texts; the train corpus of set-1 is 1,124 sentences (14,427 words) and the test corpus is 162 sentences (2,170 words). Set-2 is extracted from economy related texts; the train corpus is 3,499 sentences (40,818 words) and the test corpus is 409 sentences (4,662 words). Set-3 is not restricted to a domain; train corpus is 29,169 sentences (336,824 words) and the test corpus is 1917 sentences (15,903 words).

The experiment result shows that the proposed reestimation algorithm converges to a (local) minimum entropy. It shows also that the train and test entropies are not affected much by the domain nor by the size of training corpus. It may be because the reestimation was done on inter-POS dependencies, which is relatively simple. If the reestimation would be done on the dependencies between POS sequences for "word-phrase" or on the dependencies between lexical entities, the entropies may be affected much by the domain and the size of corpus. We plan such experiments.

Below we show the parsing results of two example Korean sentences. We used the proposed best-first parsing algorithm to find the most probable parse of each sentence. The inter-word-phrase probabilities used for parsing are the reestimated ones for the training set-3. To the right of each Korean word-phrase, the meaning of it in English is given in the square brackets. In the parse representations, each individual inter-word-phrase probability is given to the right of the dependent word-phrase. The probability of each parse is the product of all the inter-word-phrase probabilities in the parse and is given on the top of each parse.

Input Sentence:

[1]*KAIST*(Korean Advanced Institute of Science and Technology) *corpora* have been under construction since 1994 under *Korean Information Base System construction (KIBS)* project. It is supported by the Ministry of Culture and Sports, Korea. The KAIST corpora consist of raw text corpus(45,000,000 word-phrases), POS tagged corpus(6,750,000 word-phrases), and tree tagged corpus(30,000 sentences) at present. For our experiments, we extracted each train/test set from the POS tagged corpus.

1. 실험실-에-는 [the laboratory]
2. 발전기-가 [an electric generator]
3. 설치-되-어 [equipped with]
4. 있-다 [is]

Parse: 7.893022e-04

```
(EOS
    (4. 있-다 [is]    9.996915e-01
        (1. 실험실-에-는 [the laboratory]    7.129873e-02)
        (3. 설치-되-어 [equipped with]    1.150859e-01
            (2. 발전기-가 [an electric generator]    9.622178e-02)))
```

Input Sentence:

1. 터널 [tunnel]
2. 앞-을 [the front of]
3. 지나-ㄹ [passing]
4. 무렵 [when]
5. 갑자기 [suddenly]
6. 차-가 [the car]
7. 멈추-었-다 [stopped]

Parse: 3.293545e-06

```
(EOS
    (7. 멈추-었-다 [stopped]    9.996915e-01
        (4. 무렵 [when]    3.828747e-02
            (3. 지나-ㄹ [passing]    9.044279e-01
                (2. 앞-을 [the front of]    1.785020e-01
                    (1. 터널 [tunnel]    9.537951e-02))))
        (5. 갑자기 [suddenly]    5.879496e-02)
        (6. 차-가 [the car]    9.504489e-02)))
```

5 Conclusion and Further Works

In this paper we have proposed a reestimation algorithm and a best-first parsing algorithm for probabilistic dependency grammars(PDG). The reestimation algorithm is a variation of the inside-outside algorithm adapted to PDG. In both of the reestimation algorithm and the parsing algorithm, the non-constituent objects, complete-link and complete-sequence, are defined as basic units for dependency structures and a CYK-style chart is utilized. Both algorithms have $O(n^3)$ time complexities and can be used for corpus-based, stochastic

PDG induction. By experiment on Korean, we have shown that the reestimation algorithm converges to a local minimum and constitute a stable grammar.

Compared to phrase structure grammars, PDG can be a useful and practical scheme for parsing model and language model. It is because dependency tree is much simpler and easily understood than the structure constructed by the phrase structure grammars. Besides the search space of the grammar may be smaller and the effect of structural data sparseness may be less. This also means that the reestimation algorithm for PDG can converge with smaller training corpus. We are planning to evaluate the parsing model based on the reestimated PDG and the PDG-based language model.

References

Black, E., J. Lafferty, and S. Roukos. 1992. "Development and evaluation of a broad-coverage probabilistic grammar of English-language computer manuals". In *30th Annual Meeting of the Association for Computatinal Linguistics*, pages 185–192.

Carroll, G. 1992a. "Learning probabilistic dependency grammars from labeled text". In *Working Notes, Fall Symposium Series, AAAI*, pages 25–31.

Carroll, G. 1992b. "Two Experiments on Learning Probabilistic Dependency Grammars from Corpora". Technical Report CS-92-16, Brown University.

Chen, S.F. 1995. "Bayesian grammar induction for language modeling". In *33rd Annual Meeting of the Association for Computatinal Linguistics*, pages 228–235.

Collins, Michael John. 1996. "A New Statistical Parser Based on Bigram Lexical Dependencies". In *COLING-96*.

de Marcken, C. 1995. "Lexical heads, phrase structure and the induction of grammar". In *Third Workshop on Very Large Corpora*.

Eisner, Jason M. 1996. "Three New Probabilistic Models for Dependency Parsing: An Exploration". In *COLING-96*, pages 340–345.

Lari, K. and S.J. Young. 1990. "The estimation of stochastic context-free grammars using the inside-outside algorithm". *Computer Speech and Language*, 4:35–56.

Magerman, David M. 1994. *"Natural Language Parsing as Statistical Pattern Recognition"*. Ph.D. thesis, Stanford University.

Pereira, F. and Y. Schabes. 1992. "Inside-outside reestimation from partially bracketed corpora". In *30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135.