Computational Linguistics and Chinese Language Processing
Vol. 4, No. 1, February 1999, pp. 25-58
©Computational Linguistics Society of R.O.C.

A Level-Synchronous Approach to Ill-formed Sentence Parsing and Error Recovery

Yi-Chung Lin* and Keh-Yih Su⁺

Abstract

In this paper, a level-synchronous parsing mechanism, named *Phrase-Level Building* (PLB), is proposed to incorporate wide-scope contextual information for parsing ill-formed sentences. This mechanism regards the task of parsing a sentence as the task of building the phrase-levels for the sentence. Therefore, the wide-scope contextual information in the phrase-levels can be used to help narrow down the search space and to select the most likely partial parses. Compared with the system which uses both the stochastic context-free grammar and the heuristics of preferring the longest phrase, the proposed PLB approach improves the precision rate for brackets in the partial parse forests from 69.37% to 79.49%. The recall rate for brackets is also improved from 78.73% to 81.39%.

The proposed PLB parsing method can also be used to recover errors in ill-formed sentences, so that more complete syntactic information can be provided in later stages. Experimental results show that 35% of the ill-formed sentences can be recovered to well-formed parses. The recall rate for brackets is also significantly improved from 68.49% to 76.60% while the precision rate for brackets is improved slightly from 79.49% to 80.69%.

Keywords: natural language parsing, ill-formed sentence parsing, partial parsing, robust parsing, error recovery.

1. Introduction

Parsing techniques play important roles in various applications of natural language processing, such as machine translation [Hutchins, 1986; Su and Chang, 1990], speech recognition [Su, Chiang, and Lin 1990; Seneff, 1992; Meteer and Gish, 1994], and information extraction [Hobbs et al., 1992; McDonald, 1992]. It constructs the syntactic

* Advanced Technology Center, Computer & Communication Research Lab., Industrial Technology Research Institute, Chutung, Taiwan, R.O.C. Email: lyc@atc.ccl.itri.org.tw

⁺ Behavior Design Corporation, 2F, No. 5, Industrial East Road Ⅳ, Science-Based Industrial Park, Hsinchu, Taiwan, R.O.C. Email: kysu@bdc.com.tw

relationship of the words in an input sentence according to a given grammar, which formally specifies the allowable syntactic structures in the language. However, in real applications, a parser often encounters the problems resulting from ambiguous syntactic structures and ill-formed input sentences.

Ambiguous syntactic structures are generated due to the implied ambiguity in language usage or due to the over-generation of the given grammar. The number of syntactic ambiguities in a sentence depends on the given grammar. In practical applications, a sentence of average length can have up to hundreds of ambiguities, and the number of ambiguities can even reach millions in case of parsing a long sentence. To correctly interpret an input sentence, a natural language parser, therefore, must be able to choose the correct syntactic structure from a large number of ambiguities. In the past, many algorithms have been proposed solving this problem, and significant improvements have been made [Briscoe and Carroll, 1993; Chiang, Lin and Su, 1995].

As for ill-formedness, its major sources are: (1) incorrect sentences resulting from typographical errors, OCR scanning, etc., (2) unknown words which are not contained in the system dictionary, and (3) insufficient coverage of the grammar. An ill-formed sentence cannot be fitted into any well-formed syntactic structures generated by the given grammar. In other words, it is beyond the scope allowed by the system grammar. Compared with the topic of syntactic disambiguation, the issue of recovering ill-formed input has been less investigated in the literature and has often been ignored in experimental works. However, ill-formed sentences are usually inevitable in real applications because incorrect sentences always exist in the real world, and it is impossible to force users to only use the pre-specified artificial grammar and the built-in vocabulary. Therefore, the parser in a real natural language processing system must cope with the problem of ill-formed input.

In dealing with ambiguity problems in natural language processing, past researches have shown that contextual information is helpful. (One example is the well-known trigram part-of-speech tagger [Church, 1989].) Therefore, a level-synchronous parsing mechanism, called *Phrase-Level-Building* (PLB) parsing, which incorporates contextual information while parsing ill-formed sentences is proposed in this paper. In this framework, the syntactic structure of a sentence is represented by a set of *phrase-levels*, and sentence parsing is achieved by building a phrase-level set. Here, a phrase-level refers to a sequence of terminals and nonterminals captured by a particular snapshot of the parsing process. For example, in the sentence "*Printer buffers are made by DRAM*", suppose that a parser has just constructed a noun phrase for "*Printer buffers*" and a verb phrase for "*are made by DRAM*". Then, the phrase-level at this particular time would consist of "[N3[1] *Printer buffers*]" and "[V2 *are made by DRAM*]". To accelerate the parsing process, a fast level-synchronous searching mechanism is also proposed in this paper to remove less likely phrase-levels. As a result, only the partial parse forests [Tomita, 1987] with large likelihood values are generated by the PLB parser. Whenever all the different active phrase-levels in the search beam cannot be further reduced by means of any grammar rules, the process of building phrase-levels stops, and the forests retained in the search beam are scored by consulting the contextual information in the phrase-levels. Compared with the baseline system which uses both the stochastic context-free grammar and the "longest leftmost phrase first" heuristics, the proposed PLB approach improves the precision rate for brackets in the forest from 69.37% to 79.49%. The recall rate for brackets is also improved from 78.73% to 81.39%.

However, parsing ill-formed input sentences without recovering their errors only provides limited syntactic information. It would be more helpful in later stages if more complete syntactic structures can be provided. Therefore, errors in ill-formed input should be fixed. In 1981, Kwasny and Sondheimer [1981] proposed parsing and recovering errors of ill-formed sentences with an Augmented Transition Network (ATN) parser. In their approach, different types of errors are first carefully identified, and then a set of corresponding transition arcs, called relaxed arcs, is manually built into the network to recover the errors. Those relaxed arcs are blocked in normal cases. They are attempted only after all the grammatical paths have failed. Weischedel and Sondheimer [1983] used a similar approach. They used meta-rules to associate certain ill-formed sentence patterns with particular well-formed structures, which were obtained by modifying the violated grammar rules.

In 1989, Mellish [1989] proposed finding the full parse by running a modified top-down parser over the partial parses, which were generated by a bottom-up chart parser. This modified top-down parser attempts to find a full parse tree by considering only one word error. On the other hand, in 1990, Abney [1990; 1991] proposed parsing natural language by segmenting a part-of-speech sequence into chunks and then assembling the chunks into a complete parse tree. In his work, the chunks were repaired and assembled based on predefined heuristic rules. Recently, Lee *et al.* [1995] generalized the least-error recognition algorithm [Lyon, 1974] to find the full parses of minimum errors with a small grammar of only 192 grammar rules. Since exhaustively finding the full parses with minimum errors is very time-consuming, they used heuristic rules and scores to cut down the search space.

[1] See Appendix A for the definitions of the grammar symbols used in this paper.

All the approaches mentioned above employ ad hoc heuristic rules to fix errors or to restrict the search space. These heuristic rules are usually system-specific and are not easily re-used by other systems. Furthermore, although these approaches may work well for some small tasks in a given specific domain, they do not possess the characteristic of extensibility and are hard to scale-up. Therefore, a generalized approach, independent of any particular system and domain, is highly demanded.

In this paper, we propose an error recovery mechanism which uses a generalized probabilistic score function to identify and recover errors. Since errors can occur at any place in an input sentence, exhaustively searching all the possibilities is infeasible. Thus, a two-stage strategy for limiting the search space is proposed in this paper. In the first stage, the most likely forest of partial parses is checked to see if it will fit into the S-productions (i.e., the production rules whose left-hand side symbols are the start symbol "S"). If the partial parse forest cannot be well fitted by applying one or two modification actions, fixing part-of-speech errors is attempted in the second stage. Experimental results show that 35% of ill-formed sentences can be recovered to their correct well-formed full parses using this proposed approach. The recall rate for brackets is increased from 68.49% to 76.60%, while the precision rate for brackets is improved slightly from 79.49% to 80.69%.

2. Baseline System

In many frameworks [Jensen, Miller and Ravin, 1983; Mellish, 1989; Seneff, 1992; Hobbs *et al.*, 1992], the heuristics of preferring the longest phrase is used either alone or with some other system-dependent heuristics to constrain the number of possible partial parses when ill-formed sentences are parsed. We will call this type of approaches the *longest phrase first* (LF) approach. On the other hand, some parsers (such as the LR parser) parse natural language sentences from left to right and use the left context to limit the search space. If the input sentence is ungrammatical, only the partial parses beginning at the left are available in these systems. Therefore, these parsers usually select the longest leftmost phrase, which we will call *longest leftmost phrase first* (LLF) approach, as the candidate and then start to parse again from the subsequent word. To make a comparative study, both the LLF approach and the LF approach are implemented in our baseline system for handling ill-formed sentences.

The baseline system consists of two components: a *Cocke-Younger-Kasami* (CYK) parser [Aho and Ullman, 1972] and a partial parse assembler. As the CYK parser can efficiently parse an ill-formed sentence into all of its possible partial parses if the grammar is written in the Chomsky normal form [Chomsky, 1959], we first convert our normal context-free grammar into its corresponding Chomsky Normal Form (CNF)[2]. Then, the CYK parser is used to obtain all the possible partial parses. Finally, according to the adopted heuristic rule (LF or LLF), the partial parses are assembled by the partial parse assembler, which ranks these partial parses based on the probabilities of their stochastic context-free rules [Fujisaki *et al.*, 1989; Ng and Tomita, 1991]. The probability parameters adopted in this system are smoothed using the Good-Turing method [Good, 1953].

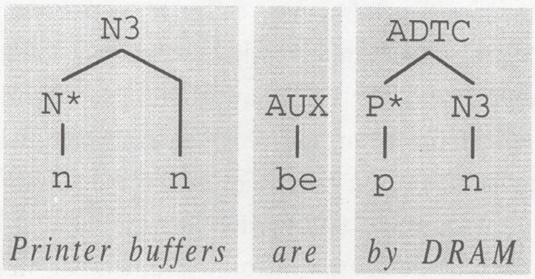

***Figure 1** A forest of partial parses for the ill-formed sentence "Printer buffers are by DRAM".*

2.1 Evaluation Method

The performance of different approaches is measured based on three factors: the bracket precision rate, the bracket recall rate and the forest accuracy rate. Take the forest[3] of partial parses in Figure 1 as an example. There are six brackets in the forest as shown below:

[N*	*Printer*],
[N3	*Printer buffers*],
[AUX	*are*],
[P*	*by*],
[N3	*DRAM*],
[ADTC	*by DRAM*].

Each of the brackets corresponds to the application of a production rule. The precision rate and the recall rate are then computed as follows:

[2] It is not necessary for the proposed approach to adhere to a CNF grammar. The CYK parsing algorithm can also work on a context-free grammar [Hopcroft and Ullman, 1976]. The only reason for converting a CFG into its CNF is to make the parsing mechanism as simple as possible.

[3] It should be noticed that the forest in Figure 1 is only one of the possible forests.

$$\text{bracket precision rate} = \frac{\text{number of exactly matched brackets}}{\text{number of brackets generated by the parser}},$$

$$\text{bracket recall rate} = \frac{\text{number of exactly matched brackets}}{\text{number of brackets in correct parses}}.$$

In the literature [Black *et al*., 1991; Doran *et al*., 1994], the evaluation of precision and recall rates, in general, does not take the associated bracket labels into account because the grammar used by the parser may differ from the one that is used in the treebank for benchmarking. However, in this task, both the parser and the treebank adopt the same grammar. Therefore, the labels of brackets are also taken into account in this paper during computation of precision and recall rates [Magerman, 1995]. In other words, we regard two brackets as being "matched" only if they also have the same label. Last, the forest accuracy rate is calculated by dividing the number of correctly parsed ill-formed sentences by the total number of ill-formed sentences. An ill-formed sentence is said to be correctly parsed if its parsed brackets are the same as those labeled manually.

Since the performance of a robust parser strongly depends on whether the parser can accurately partition the input sentence into fragments, the "*fragment precision*" and "*fragment recall*" are also measured to determine the performance of a robust parser in partitioning ill-formed sentences. Here, fragments are the brackets which are not enclosed by any other brackets, i.e., the outermost brackets. For example, there are three fragments in the forest shown in Figure 1: "[N3 *Printer buffers*]", "[AUX *are*]" and "[ADTC by *DRAM*]".

2.2 Simulation Results and Discussion

In the baseline system, 8,727 well-formed sentences, collected from computer manuals, and their correct parse trees were used as the training data. The average length of these sentences was about 13 words. All the training sentences were parsed by a context-free grammar[4] provided by the Behavior Design Corporation. This grammar consists of 29 terminals, 140 nonterminals and 1,013 production rules. To test the performance of the baseline system, 200 ill-formed sentences and their manually corrected forests were used as the testing data and the benchmark, respectively. The average length of the test sentences was also about 13 words.

[4] This grammar is a wide-coverage English grammar and has been used in commercial English-Chinese machine translation applications for several years.

Table 1 lists the simulation results obtained using different heuristic rules. The precision and recall rates for the labeled brackets are given in the first and second columns. The third column shows the accuracy rate for forests, and the last column gives the average processing time for parsing a sentence using a "SUN SPARC station ELC". The experimental results show that the LLF heuristics slightly outperforms the LF heuristics. This is because the LF heuristics, compared with the LLF heuristics, is more likely to grab the words belonging to neighboring constituents. In fact, due to the preference for a longer partial parse rather than a shorter one, the LF heuristics always partitions a sentence into as few fragments as possible. As shown in Table 2, the number of fragments generated using the LLF heuristics is only 355, which is significantly smaller than the number of correct fragments (which is 605, as shown in Table 2). However, the number of fragments generated using the LF heuristics is even smaller. In other words, the LF heuristic rule partitions ill-formed sentences more inaccurately than the LLF heuristic rule does.

Table 1. *The performance of the baseline system in the testing set using "longest phrase first" (LF) and "longest leftmost phrase first" (LLF) heuristics.*

	Bracket and its label		Forest accuracy	Parsing time (sec./sent.)
	Precision	Recall		
LF	67.98%	77.54%	16.5%	2.16
LLF	69.37%	78.73%	16.5%	2.16

The performance of this baseline system is similar to that of other systems. For example, Doran *et al.* [1994] reported that their XTAG system, based on Lexicalized Tree Adjoining Grammar formalism, achieved an 84.32% recall rate and a 59.28% precision rate in parsing part of the IBM-manual treebank. Skut and Brants [1998] reported a 72.6% precision rate in parsing the Penn Treebank using a chunk tagger. However, the values reported in those studies should not be used to infer which approach is superior because the performance of different approaches is usually measured based on different testing data.

Although the heuristics of preferring the longest phrase is adopted in many systems, it uses rather coarse knowledge to assemble partial parses. These systems always append the largest partial parse, either the leftmost one or a global one, to the forest, regardless of the context of the given partial parse. Therefore, poor performance can be expected. To remedy this drawback, a level-synchronous parsing mechanism is, thus, proposed in the next section to parse ill-formed sentences by consulting more contextual information.

Table 2. *The performance in partitioning ill-formed sentences in the testing set using LF and LLF heuristics.*

	Number of fragments		Precision	Recall
	Total	Matched		
Treebank	605	—	—	—
LF	331	160	48.3%	26.5%
LLF	355	179	50.4%	29.6%

3. PLB Parsing

To incorporate wide-scope contextual information in parsing ill-formed sentences, a *Phrase-Level Building* (PLB) parsing algorithm is proposed in this section. This algorithm treats the parsing process as a procedure for building a set of *phrase-levels*. In building the phrase-levels, a fast level-synchronous searching mechanism is used to cut down the search space. Unlike other approaches, instead of using heuristics, the final parse trees are ranked by a probabilistic score function which makes use of the contextual information in the phrase-levels. The details of this algorithm are presented in the following sections.

3.1 Phrase-Levels of Partial Parses

The basic idea in PLB parsing is to model the parsing process as a series of transformations between adjacent phrase-levels. Figure 2 is an example showing the relationship between a forest and its corresponding phrase-levels under a particular sequence of parsing actions. In this example, the forest is decomposed into four phrase-levels. The lowest one, $\mathbf{L}_1$, corresponds to the input words. The second phrase-level consists of the parts of speech of the input words. The other phrase-levels are sequences of grammar symbols (i.e., terminals and nonterminals) that are obtained by applying some production rules to the grammar symbols of their preceding adjacent phrase-level. As a result, the parsing process can be considered as the procedure of building the phrase-levels from $\mathbf{L}_1$ to $\mathbf{L}_4$ in a bottom-up manner.

To make the PLB technique, mentioned above, more clear, we further detail the operational process as follows. In Figure 2, $\mathbf{L}_3$ has five grammar symbols and is denoted as $\mathbf{L}_3$={`N* n AUX P* N3`}. By applying the productions "`N3` → `N* n`" and "

ADTC → P* N3" to the leftmost two and the rightmost two grammar symbols respectively, $\mathbf{L}_3$ can be built into $\mathbf{L}_4$, which has three grammar symbols and is denoted as $\mathbf{L}_4$={N3 AUX ADTC}. As a result, the forest in Figure 2 can be represented as **T**={$\mathbf{L}_1$, $\mathbf{R}_1$, $\mathbf{L}_2$, $\mathbf{R}_2$, $\mathbf{L}_3$, $\mathbf{R}_3$, $\mathbf{L}_4$}, where $\mathbf{R}_i$ denotes a sequence of actions which are applied to build $\mathbf{L}_i$ into $\mathbf{L}_{i+1}$. For example, to build $\mathbf{L}_4$ from $\mathbf{L}_3$ in Figure 2, $\mathbf{R}_3$ contains two actions, which correspond to applying the productions "N3 → N* n" and "ADTC → P* N3" to the leftmost two and the rightmost two grammar symbols in $\mathbf{L}_3$, respectively.

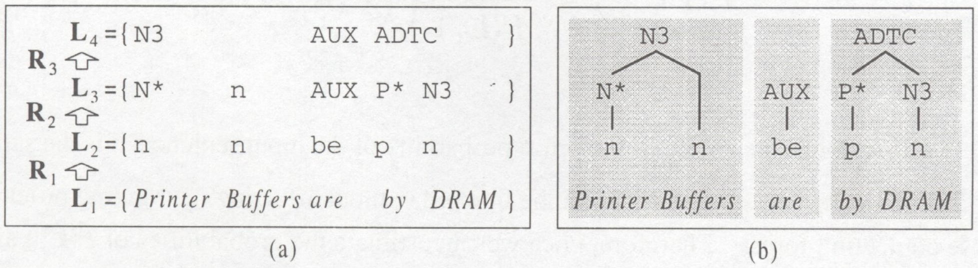

Figure 2 *(a) Parsing by building phrase-levels.*
(b) The corresponding partial parses.

Making the parsing process a procedure for building phrase-levels, the rich contextual information in a phrase-level, thus, can be used to decide which production rules are more likely to be applied (which helps us to stop building improbable structures earlier). In the following section, two scoring functions are introduced to make use of contextual information.

3.2 Scoring a Forest of Partial Parses

The likelihood value of a forest with N phrase-levels is computed as follows:

$$\begin{aligned}
&P\left(\mathbf{T}=\mathbf{L}_1,\mathbf{R}_1,\cdots,\mathbf{L}_{N-1},\mathbf{R}_{N-1},\mathbf{L}_N \mid w_1^n\right)=\prod_{i=2}^{N} P(\mathbf{L}_i,\mathbf{R}_{i-1}\mid \mathbf{L}_1,\mathbf{R}_1,\cdots,\mathbf{L}_{i-2},\mathbf{R}_{i-2},\mathbf{L}_{i-1})\\
&\approx \prod_{i=2}^{N} P(\mathbf{L}_i,\mathbf{R}_{i-1}\mid \mathbf{L}_{i-1}) = \prod_{i=2}^{N}\frac{P(\mathbf{L}_{i-1}\mid \mathbf{L}_i,\mathbf{R}_{i-1})\times P(\mathbf{L}_i,\mathbf{R}_{i-1})}{P(\mathbf{L}_{i-1})}\\
&= \prod_{i=2}^{N}\frac{P(\mathbf{L}_{i-1}\mid \mathbf{L}_i,\mathbf{R}_{i-1})\times P(\mathbf{R}_{i-1}\mid \mathbf{L}_i)\times P(\mathbf{L}_i)}{P(\mathbf{L}_{i-1})} \qquad (1)\\
&= \frac{P(\mathbf{L}_N)}{P(\mathbf{L}_1)}\prod_{i=2}^{N}\{P(\mathbf{L}_{i-1}\mid \mathbf{L}_i,\mathbf{R}_{i-1})\times P(\mathbf{R}_{i-1}\mid \mathbf{L}_i)\}\\
&= \frac{P(\mathbf{L}_N)}{P(\mathbf{L}_1)}\prod_{i=2}^{N} P(\mathbf{R}_{i-1},\mathbf{L}_{i-1}\mid \mathbf{L}_i).
\end{aligned}$$

In the above equation, the priori probability $P(\mathbf{L}_N)$ is intentionally introduced by applying the Bayesian formula, as it is useful information for assembling partial parses into a forest. Taking the case of Figure 2 as an example, the preference of the forest, according to Equation (1), is considered to be closely related to the likelihood value of the nonterminal sequence "N3 AUX ADTC" (i.e., $\mathbf{L}_N$, where N=4).

Becuase $\mathbf{L}_{i-1}$ is uniquely determined by $\mathbf{R}_{i-1}$ and $\mathbf{L}_i$, i.e., $P(\mathbf{R}_{i-1}, \mathbf{L}_{i-1} | \mathbf{L}_i) = P(\mathbf{R}_{i-1} | \mathbf{L}_i)$, Equation (1) can be rewritten as

$$P\left(\mathbf{T} \mid w_1^n\right) \approx \frac{P\left(\mathbf{L}_N\right)}{P\left(\mathbf{L}_1\right)} \prod_{i=2}^{N} P\left(\mathbf{R}_{i-1} \mid \mathbf{L}_i\right). \tag{2}$$

Furthermore, since $P(\mathbf{L}_1)$ is the priori probability of the input sentence, it is the same for all competing forests and, thus, can be ignored without affecting the ranking order of those competing forests. Therefore, once we can estimate the probabilities of $P(\mathbf{L}_N)$ and $P(\mathbf{R}_{i-1} | \mathbf{L}_i)$, we can determine the ranking order of the competing forests. The probability $P(\mathbf{L}_N)$ can be simplified by using a trigram model as follows:

$$P\left(\mathbf{L}_N = A_1, \cdots, A_n\right) \approx \prod_{A_j \in \mathbf{L}_N} P\left(A_j \mid A_{j-2}, A_{j-1}\right), \tag{3}$$

where $A_1, \cdots, A_n$ are the n symbols in the phrase-level $\mathbf{L}_N$. For example, in Figure 2, the priori probability $P(\mathbf{L}_4)$ will be approximated as the product of $P(\text{N3} \mid \$\ \$)$[5], $P(\text{AUX} \mid \$\ \text{N3})$ and $P(\text{ADTC} \mid \text{N3 AUX})$. This trigram model implicitly assumes that the information of those two left partial parses is more relevant in determining the current partial parse. Take the sentence "Include files provided to you" as an example. If the first two partial parses are parsed to be [v Include] [N3 files], the partial parse of "provided to you" should be a relative clause with a gap rather than a verb phrase.

On the other hand, the computation of $P(\mathbf{R}_{i-1} \mid \mathbf{L}_i)$ is somewhat more complicated. Let $\rho_{i-1,j} = <r\,;\,t>$ denote the j-th action in $\mathbf{R}_{i-1}$, let and $A_{i,j}$ denote the j-th symbol in $\mathbf{L}_i$, where r represents the production rule applied by $\rho_{i-1,j}$, and t indicates that the

5 The symbol $ represents the sentence boundary marker.

left-hand-side symbol of r is the t-th symbol in $\mathbf{L}_i$ (i.e., $A_{i,t}$). For example, in Figure 2, the production rules "N3 → N* n" and "ADTC → P* N3" are applied to build $\mathbf{L}_3$ into $\mathbf{L}_4$. Therefore, there are two actions $\rho_{3,1}$ = <N3 → N* n; 1> and $\rho_{3,2}$ = <ADTC → P* N3;3> in $\mathbf{R}_3$, where the arguments "1" and "3" indicate that N3 (the left-hand-side symbol of "N3 → N* n") is the first element and ADTC (the left-hand-side symbol of "ADTC → P* N3") is the third element in $\mathbf{L}_4$, respectively. Hereafter, all the level indices (i and i-1) will be dropped for expression conciseness, if not causing misunderstanding.

Traditionally, the stochastic context-free grammar obtains the likelihood value of a set of production rules by multiplying the probabilities of those rules together. However, recent researches [Briscoe and Carroll, 1993; Chiang, Lin and Su, 1995] have shown that the contextual information should be taken into account if the training data is enough. For example, in our task, the probability of deriving a pronoun from a noun phrase (i.e., applying the rule "N3 → pron") is about 0.12. But if the noun phrase is at the beginning of the sentence and is followed by an auxiliary, the probability of deriving a pronoun from a noun phrase is significantly increased to 0.28. Therefore, we derive the conditional probability $P(\mathbf{R}_{i-1} \mid \mathbf{L}_i)$ in Equation (2) by incorporating the contextual information as follows. Assume that there are m actions in $\mathbf{R}_{i-1}$ and n symbols in $\mathbf{L}_i$; then,

$$P(\mathbf{R}_{i-1} \mid \mathbf{L}_i) = P(\rho_1^m \mid A_1^n) = \prod_{j=1}^{m} P(\rho_j \mid \rho_1^{j-1}, A_1^n) \approx \prod_{\rho=\langle r;t\rangle \in \mathbf{R}_{i-1}} P(r \mid A_{t-1}^{t+1}), \tag{4}$$

where the action ρ = <r ; t> is assumed to be dependent on the its local context A_{t-1} and A_{t+1} . For instance, in Figure 2, the probability $P(\mathbf{R}_3 \mid \mathbf{L}_4)$ is approximated as the product of P(N3 → N* n|\$ N3 AUX) and P(ADTC → P* N3|AUX ADTC \$).

According to Equations (2) - (4), the likelihood of a forest is then approximated as follows:

$$P(\mathbf{T} \mid w_1^n) \approx \frac{1}{P(\mathbf{L}_1)} \times \prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho=\langle r;t\rangle \in \mathbf{R}_i} P(r \mid A_{t-1}^{t+1}), \tag{5}$$

where A_{t-1}^{t+1} is the short-hand notation for "A_{t-1}, A_t, A_{t+1}" in $\mathbf{L}_{i+1}$. As mentioned before, the probability factor $P(\mathbf{L}_1)$ can be ignored without changing the ranking order of forests because it is the same for all competing forests. Therefore, we obtain the following score function to rank the forests:

$$S_{\mathrm{FS}}(\mathbf{T}) \equiv \prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho=\langle r;t \rangle \in \mathbf{R}_i} P(r \mid A_{t-1}^{t+1}), \qquad (6)$$

where the subscript "FS" in $S_{\mathrm{FS}}(\cdot)$ means "forest selection".

3.3 The PLB Parsing Mechanism

The PLB parser parses an input sentence by building a set of phrase-levels for the sentence. Each possible sequence of phrase-levels represents a particular forest of partial parses. Since many different action sets can be applied to a given phrase-level to build it into a higher level, the number of possible phrase-level sequences increases exponentially as the phrase-levels are built upward. Because it is infeasible to exhaustively generate all possible phrase-level sequences, the beam search strategy is adopted here to find forests with higher likelihood values.

To efficiently carry out the beam search, a score function which can rank the candidates of each phrase-level in a very short time is required. However, the score function in Equation (6) spends too much computational time on ranking the candidates of a phrase-level as explained below. Suppose that the parser is going to build a particular phrase-level into its higher level. According to Equation (6), all possible candidates of that phrase-level must be expanded because the contextual information of all possible new phrase-levels is required so that every candidate can be scored. However, it is quite time-consuming to expand all these candidates. One way to overcome this problem is to adopt another time-saving score function to rapidly find the potential candidates for the given phrase-level. If this scoring function can score the candidates based on the contextual information in the old phrase-level, not in the new just expanded phrase-levels, the potential candidates for the new phrase-level, then, can be ranked even before they are fully constructed. Therefore, the time required to fully expand the uninteresting candidates can be saved. Afterwards, we only need to re-score the forests which are retained in the search beam using Equation (6) to find the final best forest.

$$\begin{array}{rl} & \mathbf{L}_4=\{\ \texttt{N3}\qquad \texttt{AUX}\qquad \texttt{ADTC}\qquad \} \\ \textbf{Transform}\Uparrow & \\ & \mathbf{C}_3=\{\ [\texttt{N* n}]\quad [\texttt{AUX}]\quad [\texttt{P* N3}]\ \} \\ \textbf{Segment}\Uparrow & \\ & \mathbf{L}_3=\{\ \texttt{N* n}\qquad \texttt{AUX}\qquad \texttt{P* N3}\quad \} \end{array}$$

Figure 3 *Segmenting and transforming phrase-levels.*

To derive the time-saving score function, the process of building a phrase-level $\mathbf{L}_i$ into a higher phrase-level $\mathbf{L}_{i+1}$ is remodeled as the process of first segmenting $\mathbf{L}_i$ into segments and then transforming these segments into $\mathbf{L}_{i+1}$. Recall the forest shown in Figure 2. The phrase-level $\mathbf{L}_3$={`N* n AUX P* N3`} is built into $\mathbf{L}_4$={`N3 AUX ADTC`} by applying the production rules "`N3` → `N* n`" and "`ADTC` → `P* N3`". This process can be decomposed into two steps. First, segment $\mathbf{L}_3$ into three segments: "`[N* n]`", "`[AUX]`" and "`[P* N3]`". Second, transform these three segments into $\mathbf{L}_4$={`N3 AUX ADTC`}. Figure 3 gives a graphical illustration of these two steps. Based on this viewpoint, a forest with i phrase-levels can alternatively be represented by a sequence of unsegmented and segmented phrase-levels as $\{\mathbf{L}_1, \mathbf{C}_1, \cdots, \mathbf{L}_{i-1}, \mathbf{C}_{i-1}, \mathbf{L}_i\}$, where $\mathbf{C}_j$ denotes the segmented phrase-level of $\mathbf{L}_j$. As a result, the likelihood of a partial tree with i phrase-levels is computed as

$$P(\mathbf{L}_1,\mathbf{C}_1,\cdots,\mathbf{L}_{i-1},\mathbf{C}_{i-1},\mathbf{L}_i \mid w_1^n) = \prod_{j=2}^{n} P(\mathbf{L}_j,\mathbf{C}_{j-1} \mid \mathbf{L}_1,\mathbf{C}_1,\cdots,\mathbf{L}_{j-2},\mathbf{C}_{j-2},\mathbf{L}_{j-1})$$
$$\approx \prod_{j=2}^{n} P(\mathbf{L}_j,\mathbf{C}_{j-1} \mid \mathbf{L}_{j-1}). \tag{7}$$

The approximation in the above equation is based on the assumption that the segmentation and transformation results (i.e., $\mathbf{C}_{j-1}$ and $\mathbf{L}_j$) only depend on the previous phrase-level $\mathbf{L}_{j-1}$. According to Equation (7), the "beam search" score function $S_{BS}(\cdot)$[6] is defined as follows so that the score of a forest with i phrase-levels can be evaluated rapidly:

[6] The performance may be further improved if the evaluation score $S_{FS}(\cdot)$ in Equation (6) is replaced by the Log-linear interpolation of $S_{FS}(\cdot)$ and $S_{BS}(\cdot)$.

$$S_{\mathrm{BS}}\left(\mathbf{L}_1,\mathbf{C}_1,\cdots,\mathbf{L}_{i-1},\mathbf{C}_{i-1},\mathbf{L}_i\right) \equiv S_{\mathrm{lex}}\left(\mathbf{L}_2,\mathbf{C}_1 \mid \mathbf{L}_1\right)\times\prod_{j=3}^{n} S_{\mathrm{syn}}\left(\mathbf{L}_j,\mathbf{C}_{j-1} \mid \mathbf{L}_{j-1}\right), \tag{8}$$

where $S_{\mathrm{lex}}(\mathbf{L}_2, \mathbf{C}_1 \mid \mathbf{L}_1)$ denotes the lexical score of the part-of-speech sequence of $\mathbf{L}_2$, and $S_{\mathrm{syn}}(\mathbf{L}_j, \mathbf{C}_{j-1} \mid \mathbf{L}_{j-1})$ denotes the syntactic score corresponding to the *j*-th phrase-level $\mathbf{L}_j$. The lexical and syntactic scores are provided by the lexical and syntactic modules respectively. The following sections describe these two modules in detail.

3.3.1 Lexical Module

The lexical module is basically a statistical tagger [Church, 1989] which finds the most likely part-of-speech sequence for the input sentence. The likelihood of a part-of-speech sequence $\mathbf{L}_2$ for the input word sequence $\mathbf{L}_1$ is computed according to the widely-used trigram model [Church, 1989; Lin, Chiang and Su 1995] as follows:

$$\begin{aligned} P\left(\mathbf{L}_2,\mathbf{C}_1 \mid \mathbf{L}_1\right) &= P\left(c_1^n \mid w_1^n\right) = \frac{P\left(c_1^n\right)}{P\left(w_1^n\right)}P\left(w_1^n \mid c_1^n\right) \\ &\approx \frac{1}{P\left(w_1^n\right)}\prod_{j=1}^{n} P\left(c_j \mid c_{j-2},c_{j-1}\right)P\left(w_j \mid c_j\right), \end{aligned} \tag{9}$$

where n is the number of words in the input sentence, w_j is the *j*-th input word, and c_j denotes the part of speech for w_j. Since the probability $P\left(w_1^n\right)$ is a constant, it can be ignored without affecting the ranking order of the likelihood probabilities of these competing part-of-speech sequences. Therefore, the score function for the lexical module is defined as

$$S_{\mathrm{lex}}\left(\mathbf{L}_2,\mathbf{C}_1 \mid \mathbf{L}_1\right) \equiv \prod_{j=1}^{n} P\left(c_j \mid c_{j-2},c_{j-1}\right)P\left(w_j \mid c_j\right). \tag{10}$$

3.3.2 Syntactic Module

The syntactic module is responsible for ranking the phrase-level candidates which are one level higher than the given phrase-level. The likelihood of a phrase-level candidate

$\mathbf{L}_i$ for the given phrase-level $\mathbf{L}_{i-1}$ is computed as

$$P(\mathbf{L}_i, \mathbf{C}_{i-1} \mid \mathbf{L}_{i-1}) = P(\mathbf{L}_i \mid \mathbf{C}_{i-1}, \mathbf{L}_{i-1}) P(\mathbf{C}_{i-1} \mid \mathbf{L}_{i-1}) = P(\mathbf{L}_i \mid \mathbf{C}_{i-1}) P(\mathbf{C}_{i-1} \mid \mathbf{L}_{i-1}). \tag{11}$$

Let $A_1, \cdots, A_n$ be those n symbols in $\mathbf{L}_{i-1}$, and let $\alpha_1, \cdots, \alpha_m$ be those m segments in $\mathbf{C}_{i-1}$; then, $P(\mathbf{C}_{i-1} \mid \mathbf{L}_{i-1})$ is approximated as

$$\begin{aligned} P(\mathbf{C}_{i-1} \mid \mathbf{L}_{i-1}) &= P(\alpha_1^m \mid A_1^n) = \prod_{j=1}^{m} P(\alpha_j \mid \alpha_1^{j-1}, A_1^n) \approx \prod_{j=1}^{m} P(\alpha_j \mid \alpha_{j-2}, \alpha_{j-1}) \\ &\approx \prod_{j=1}^{m} P(\alpha_j \mid \Gamma_{R2}(\alpha_{j-2}\alpha_{j-1})), \end{aligned} \tag{12}$$

where $\Gamma_{R2}(\alpha_{j-2}, \alpha_{j-1})$ denotes the rightmost two symbols in "$\alpha_{j-2}\, \alpha_{j-1}$"[7]. Taking Figure 3 as an example, the score $P(\mathbf{C}_3 \mid \mathbf{L}_3)$ is approximated as P([N* n] | \$, \$) × P([AUX] | N*, n) × P([P* n3] | n, AUX). Likewise, $P(\mathbf{L}_i \mid \mathbf{C}_{i-1})$ is further simplified as

$$\begin{aligned} P(\mathbf{L}_i \mid \mathbf{C}_{j-1}) &= P(A_1^n \mid \alpha_1^m) = \prod_{j=1}^{m} P(A_j \mid A_1^{j-1}, \alpha_1^m) \approx \prod_{j=1}^{m} P(A_j \mid \alpha_{j-1}, \alpha_j, \alpha_{j+1}) \\ &\approx \prod_{j=1}^{m} P(A_j \mid \Gamma_{R1}(\alpha_{j-1}), \alpha_j, \Gamma_{L1}(\alpha_{j+1})), \end{aligned} \tag{13}$$

where $\Gamma_{R1}(x)$ and $\Gamma_{L1}(x)$ denote the rightmost symbol and the leftmost symbol of x, respectively. For example, in the case of Figure 3, $P(\mathbf{L}_4 \mid \mathbf{C}_3)$ is approximated as P(N3 | \$, [N* n], AUX) × P(AUX | n, [AUX], P*) × P(ADTC | AUX, [P* n3], \$) . Therefore, according to Equations (11)- (13), the syntactic score function is defined as

$$S_{\text{syn}}(\mathbf{L}_j, \mathbf{C}_{j-1} \mid \mathbf{L}_{j-1}) \equiv \prod_{j=1}^{m} \{ P(\alpha_j \mid \Gamma_{R2}(\alpha_{j-2}\alpha_{j-1}))\, P(A_j \mid \Gamma_{R1}(\alpha_{j-1}), \alpha_j, \Gamma_{L1}(\alpha_{j+1})) \}, \tag{14}$$

where α_j is the j-th segment in $\mathbf{C}_{i-1}$, and A_j is the j-th symbol in $\mathbf{L}_i$.

[7] Since the segment sequence "$\alpha_{j-2}\, \alpha_{j-1}$" is on the left side of α_j, the rightmost two symbols in "$\alpha_{j-2}\alpha_{j-1}$" are adjacent to α_j.

3.4 Simulation Results and Discussion

As mentioned above, the PLB parsing mechanism uses the score function $S_{BS}(\cdot)$ in Equation (8) to rapidly rank the candidates of phrase-levels and remove the less likely ones during constructing the forests. Therefore, only the forests with higher likelihood values are generated. Finally, the score function $S_{FS}(\cdot)$ in Equation (6) is used to select the best one from these generated forests. The performance of PLB parsing in the testing set is shown in Table 3 for various beam widths. Because the probability that the correct forest is included in the search space (i.e., the including rate) increases as the beam width increases, the accuracy rate shown in Table 3 rises until the beam width reaches a size of 20. Afterwards, the including rate saturates, and the effect of the increasing ambiguous competitors due to the increasing beam width dominates the performance. These introduced ambiguities not only slightly degrade the accuracy rate but also significantly increase the computational time. Similar phenomena have also been observed in other applications that adopt the beam search strategy [Chiang, Lin and Su, 1996]. Considering both the accuracy rate and computational time, a beam width of 20 is recommended for PLB parsing in this task.

The results of the baseline system with LLF heuristics (selecting the longest left-most phrase first) are also listed in Table 3 for comparison. It is obvious that the PLB approach significantly outperforms the baseline system. Even when a very small beam width is used, the PLB approach achieves better results than the baseline system in terms of the precision and recall rates for brackets as well as in terms of the forest accuracy. Since the search space is cut down via a probabilistic score function during parsing, the PLB approach can rapidly select the most likely combinations of partial parses for ill-formed sentences. Therefore, the PLB approach with a small beam width can parse sentences faster than the baseline system can. However, the above results cannot lead us to the conclusion that the PLB approach can save more time than other systems with LLF heuristics do. The reason is that the LLF heuristics can be implemented by a LR parser, which is usually more efficient than the CYK parser used in the baseline system. However, since the PLB approach yields better results than the LLF heuristic approach does within one second, the LLF heuristics is no more attractive even if it can be implemented using a faster parser.

Table 3 also shows that the improvement in the bracket precision rate achieved by using the PLB approach is better than that achieved in the bracket recall rate. For instance, compared with using the LLF heuristics, the PLB approach with a beam width of 20 improves the bracket precision rate by 10.12% (from 69.37% to 79.49%) while it only improves the bracket recall rate by only 2.66% (from 78.73% to 81.39%). To further explore the reason for the above phenomenon, more detailed data are given in Table 4. Compared with the PLB approach, the LLF approach yields significantly more brackets (3794 versus 3423). This is because a longer phrase tends to have a deeper structure, which will consist of more brackets. Since more brackets are generated with fewer correct brackets (2632 versus 2721), the LLF approach achieves significantly lower bracket precision rate than the PLB approach does.

Table 3. *The performance of PLB parsing in the testing set for various beam widths.*

	Beam width	Bracket and its label		Forest accuracy	Parsing time (sec./sent.)
		Precision	Recall		
PLB	3	78.52%	79.27%	24.5%	0.46
	5	78.22%	80.56%	25.5%	0.66
	10	78.78%	80.74%	26.0%	1.17
	20	79.49%	81.39%	27.5%	2.51
	50	80.08%	80.92%	26.5%	9.75
	100	80.11%	80.98%	27.0%	35.93
LLF	—	69.37%	78.73%	16.5%	2.16

Table 4. *The detailed results for the baseline system and the PLB approach in the testing set.*

	Number of brackets		Precision	Recall
	Total	Matched		
Treebank	3343	—	—	—
LLF	3794	2632	69.37%	78.73%
PLB	3423	2721	79.49%	81.39%

Selecting the longest phrase also causes the baseline system to inaccurately partition the ill-formed sentences. As shown in Table 5, there are 605 fragments in those 200 ill-formed sentences. However, the baseline system partitions these ill-formed sentences into only 355 fragments, which is too few. This is due to the fact that the baseline system does not consider the contextual information when partial parses are assembled. It always prefers a longer partial parse to a shorter one, and, consequently, partitions a sentence into as few fragments as possible. The baseline system, thus, has a very low recall rate for fragments. On the other hand, the PLB approach can more accurately partition ill-formed sentences because it uses statistical contextual information. The number of fragments that the PLB approach generates is almost equal to the number of correct fragments. Besides, with the aid of contextual information, the PLB approach yields more matched fragments than the baseline system does. Therefore, the PLB approach has a significantly higher recall rate for fragments than the baseline system has (57.9% vs. 29.6%).

Table 5. *The performance in partitioning ill-formed sentences in the testing set using LLF and PLB approaches.*

	Number of fragments		Precision	Recall
	Total	Matched		
Treebank	605	—	—	—
LLF	355	179	50.4%	29.6%
PLB	656	350	53.4%	57.9%

In summary, the proposed PLB approach outperforms the baseline system to a great extent because it uses statistical contextual information. With a beam width of 20, the PLB approach significantly improves the precision rate for brackets in the forests from 69.37% to 79.49%. The recall rate for brackets is also improved from 78.73% to 81.39%. Although the above experiment was performed on ill-formed sentences, the proposed model can also be used to parse well-formed sentences. However, since parsing well-formed sentences is not the focus of this paper, it will not be further investigated here.

4. Error Recovery with Modification Actions

The above level-synchronous approach does not fix any errors in ill-formed sentences. However, an unfixed error may affect neighboring words and prevent them from being recognized by the parser. The wicked effect of these errors may even extend to other processing stages following syntactic parsing. To reduce the effects caused by such errors, the parser should recover as many errors as possible before they slip into subsequent stages.

To fix the errors in a phrase-level, the modification actions of insertion, deletion and substitution are incorporated into the parsing process. These actions are realized by three modification productions: $X \rightarrow \varepsilon$ (which denotes insertion of a symbol X), $\varepsilon \rightarrow X$ (which denotes deletion of a symbol X) and $Y \rightarrow X$ (which denotes replacement of the symbol X with the symbol Y). A modification action consists of a rule argument and two associated position arguments, such as $\tilde{\rho} = \langle \tilde{r}; u, v \rangle$, where $\tilde{r}$ stands for the applied modification rule, u and v indicate that this modification action is applied between the u -th and the v-th symbols in the modified phrase-level.

Take the input "*Printer Buffers are by DRAM*" as an example. This sentence misses a verb after the auxiliary "*are*". To correct such an error, a verb should be inserted so that $\mathbf{L}_2$={n n be p n} can be modified to obtain $\tilde{\mathbf{L}}_2$ ={n n be v p n}. In this case, the corresponding modification action set $\tilde{\mathbf{R}}_2$ will consist of one modification action $\tilde{\rho} = \langle \tilde{r} : \mathrm{v} \rightarrow \varepsilon; u = 3, v = 5 \rangle$. In that modification action, the rule argument $\tilde{r} : \mathrm{v} \rightarrow \varepsilon$ indicates insertion of a verb, and the position arguments $u = 3$ and $v = 5$ indicate that the inserted verb will be placed between the third and the fifth symbols in the modified phrase-level $\tilde{\mathbf{L}}_2$.

After having incorporated with modification actions, the parsing process will apply the modification actions and the normal reduction actions in turn. A modified parse tree $\tilde{\mathbf{T}}$ of N phrase-levels can then be represented as

$$\tilde{\mathbf{T}} = \{\mathbf{L}_1, \tilde{\mathbf{R}}_1, \tilde{\mathbf{L}}_1, \mathbf{R}_1, \mathbf{L}_2, \cdots, \mathbf{L}_{N-1}, \tilde{\mathbf{R}}_{N-1}, \tilde{\mathbf{L}}_{N-1}, \mathbf{R}_{N-1}, \mathbf{L}_N\} ,$$

where $\mathbf{L}_i$ is the i-th phrase-level, $\tilde{\mathbf{R}}_i$ is the set of modification actions adopted to modify $\mathbf{L}_i$, $\tilde{\mathbf{L}}_i$ is the result obtained after applying $\tilde{\mathbf{R}}_i$ to $\mathbf{L}_i$, and $\mathbf{R}_i$ is the set of normal actions applied to build $\mathbf{L}_{i+1}$ from $\tilde{\mathbf{L}}_i$. After some derivations and approximations (please see Appendix B), the score of a modified parse tree is defined as

$$S_{\mathrm{MT}}(\tilde{\mathbf{T}}) \equiv \prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho=<r;t> \in \mathbf{R}_i} P(r \mid A_{t-1}^{t+1}) \\ \times \prod_{\tilde{\mathbf{R}}_i = \phi} P(\tilde{\mathbf{R}}_i = \phi) \times \prod_{\tilde{\mathbf{R}}_i \neq \phi} \left\{ P(\tilde{\mathbf{R}}_i \neq \phi) \times \prod_{\tilde{\rho}=<\tilde{r};u,v> \in \tilde{\mathbf{R}}_i} P(\tilde{r} \mid \tilde{A}_u^v) \right\}, \quad (15)$$

where the subscript "MT" in $S_{\mathrm{MT}}(\cdot)$ means "modified tree", the notation $\tilde{\mathbf{R}}_i = \phi$ indicates that $\tilde{\mathbf{R}}_i$ is an empty set (i.e., no modifications have been applied to modify $\mathbf{L}_i$), $\tilde{\mathbf{R}}_i \neq \phi$ denotes that it is not an empty set and includes some modification action rules, and $\tilde{A}_u^v$ is the short-hand notation for symbols from the u-th position to the v-th position in $\tilde{\mathbf{L}}_i$. The first two product terms, which are related to the normal productions, are the same as those in Equation (6). The third product term $\prod_{\tilde{\mathbf{R}}_i=\phi} P(\tilde{\mathbf{R}}_i=\phi)$ is related to those phrase-levels which do not need to be modified. The last product term $\prod_{\tilde{\mathbf{R}}_i\neq\phi}\left\{P(\tilde{\mathbf{R}}_i\neq\phi)\times\prod_{\tilde{\rho}=<\tilde{r};u,v>\in\tilde{\mathbf{R}}_i}P(\tilde{r}\mid\tilde{A}_u^v)\right\}$ accounts for the modification actions, where the sequence $\tilde{A}_u^v$ contains the left context (i.e., $\tilde{A}_u$) and the right context (i.e., $\tilde{A}_v$) of the applied modification rule $\tilde{r}$. For example, to recover the error in the ill-formed sentence "*Printer Buffers are by DRAM*" by inserting a verb after the auxiliary "*are*", a modification set $\tilde{\mathbf{R}}_2$ ={ $\tilde{\rho}=\langle\tilde{r}:\mathtt{v}\rightarrow\varepsilon;u=3,v=5\rangle$ } can be applied to modify $\mathbf{L}_2$ ={`n n be p n`} to obtain $\tilde{\mathbf{L}}_2$ ={`n n be v p n`}. In this case, the modification rule "$\mathtt{v}\rightarrow\varepsilon$" is considered depending on the third symbol (the left context) and the fifth symbol (the right context) in $\tilde{\mathbf{L}}_2$. As a result, for this nonempty modification set $\tilde{\mathbf{R}}_2$, the corresponding probability term $\prod_{\tilde{\rho}=<\tilde{r};u,v>\in\tilde{\mathbf{R}}_2}P(\tilde{r}\mid\tilde{A}_u^v)$ is set to be the term P($\mathtt{v}\rightarrow\varepsilon$ |`be v n`).

5. Two-stage Strategy to Find Potential Modification Actions

One serious problem in doing error-recovery is that the possible moditcation actions are much more than the possible normal production actions. It is infeasible to first rank all possible alternatives and then retain the more likely ones, as we did in Section 3. Therefore, a two-stage strategy is proposed solving this problem. Instead of blindly trying all possible modification actions, this strategy only tries to fit the forest of partial parses into an S-production rule (i.e., one whose left-hand-side symbol is the start symbol) during the first stage, which is a low cost approach. If the forest of partial parses cannot be well fitted into any S-production rule, the sentence is then passed to the second stage to recover the part-of-speech errors, which requires much more computational cost. Therefore, without attacking every case with a full-blown approach, this practical strategy tries the simple approach first (which works in most cases) and only passes the unresolved ill-formed sentences to the second stage. These two stages are described in detail in the following sections.

5.1 Fitting the Partial Parses

The errors in ill-formed sentences can be divided into two types: isolated errors and clingy errors. Isolated errors are those that do not hinder the parser from correctly parsing other phrases. Taking the sentence "*Printer buffers are made by DRAM*" as an example, if the noun phrase "*Printer buffers*" is missing, the other words can still be parsed into a verb phrase. Therefore, such an error is an isolated error. On the other hand, if the word "*made*" is missing, this error will hinder a parser from parsing the other three words "*are by DRAM*" into a verb phrase. Thus, this error is a clingy error. (It clings to other words.)

Isolated errors can be recovered by fitting the partial parses [Jensen, Miller, and Ravin, 1983]. In the past, the fitting procedure was usually guided by heuristic rules, such as preferring some head phrases, preferring the widest phrase, etc. However, acquiring these heuristic rules is expensive, and it is also difficult to maintain consistency among a large number of rules. Therefore, the heuristic approach is hard to scale up. Furthermore, the heuristic rules are usually system-specific and are not easily re-used by other systems. The probabilistic approach is, thus, proposed in this paper fixing these isolated errors.

Because most of the partial parses are constituents of S-production rules, the forest of partial parses is first tried to fit into an S-production rule. For example, once the incomplete sentence "*are made by DRAM*" is parsed into a verb phrase, we can recover it by fitting the verb phrase into the right-hand-side of the S-production "S → N3 V2". In this case, a modified full parse can be obtained by inserting a noun phrase in front of the verb phrase, as shown in Figure 4.

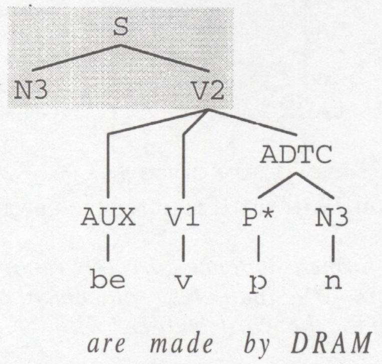

Figure 4 *Fitting a verb phrase to a full parse according to the S-production rule "S→N3 V2".*

In our experiment, it was observed that most of the isolated errors could be fixed with two modification actions. Besides, it is obvious that if too many modification actions are allowed during fitting S-production rules, every ill-formed sentence can be fitted into an S-production, including the ill-formed sentences of clingy errors, which should be fixed in the bottom phrase-level. (See next section for details.) Therefore, in the current system, at most two modification actions are tried during fitting a forest of partial parses into the right-hand side of a S-production rule. Different modification actions can fit the forest into different S-production rules and construct different full parse trees. Those full parse trees are then ranked based on the score function $S_{MT}(\cdot)$ in Equation (15). If the forest of partial parses cannot be fitted into a full tree by means of one or two modification actions, it is assumed that the errors in this ill-formed input are not isolated. Then, the partial parses are passed to the second stage, where the clingy errors are fixed.

5.2 Recovering Part-of-Speech Errors

It is noticed that many clingy errors come from the parts of speech in the second phrase-level. Therefore, in the second stage, attempts are made to recover the errors originated from the second phrase-level. For example, as shown in Figure 5, the word "sufflce", which is a typo of the word "suffice", is initially regarded as an unknown word and, thus, is regarded as a noun by default. To recover such an error, the part of speech for the unknown word "sufflce" should be changed from noun to verb.

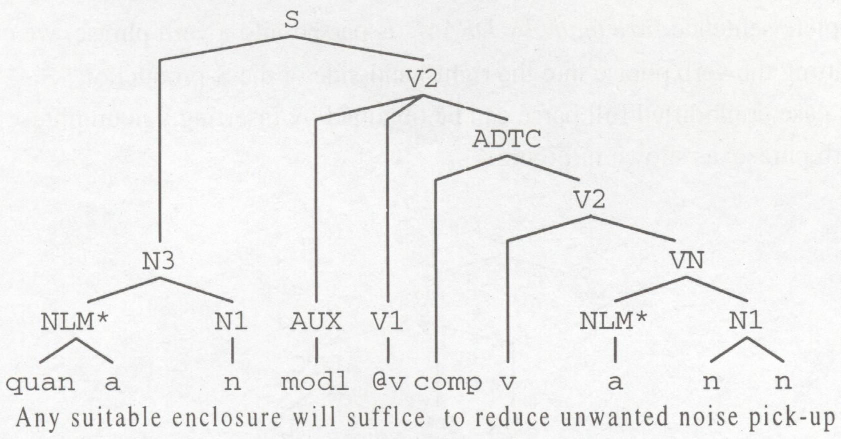
S
V2
ADTC
V2
N3
VN
NLM*
N1
AUX
V1
NLM*
N1
quan
a
n
modl
@v
comp
v
a
n
n
Any suitable enclosure will sufflce to reduce unwanted noise pick-up

Figure 5 *An ill-formed sentence correctly recovered in the second stage, where "@V" indicates replacement of the part of speech for the word "sufflce" with "v".*

Since an enormous number of modifications can be applied to modify a part-of-speech sequence, it is infeasible to try all of them. To be practical, currently, only the modifications with one insertion, deletion or substitution are permitted. Furthermore, since the number of ill-formed sentences in our training set is rather limited, we cannot get reliable statistical parameters from them. Therefore, the statistical information of these three score functions, described below, is acquired from well-formed training data.

Using the trigram formulation, the likelihood of a part-of-speech sequence, $c_1^n = c_1, c_2, \cdots, c_n$ can be approximated as $\prod_{j=1}^{n} P(c_j \mid c_{j-2}, c_{j-1})$, where c_j denotes the j-th part of speech. Therefore, the score for inserting a part of speech "x" in front of the i-th part of speech is defined as

$$S_{\mathrm{INS}}(i, x; c_1^n) \equiv P(x \mid c_{i-2}, c_{i-1}) \times P(c_i \mid c_{i-1}, x) \times P(c_{i+1} \mid x, c_i) \times \prod_{\substack{j=1 \\ j \neq i,\, i+1}}^{n} P(c_j \mid c_{j-2}, c_{j-1}). \tag{16}$$

In other words, if we insert a part of speech "x" in front of the i-th part of speech in the sequence $c_1, c_2, \cdots, c_n$, we can obtain a new part-of-speech sequence $c_1, \cdots, c_{i-2}, c_{i-1}, x, c_{i+1}, c_{i+2}, \cdots, c_n$, and then assign the trigram score of this new sequence as the score of this insertion action. According to this score function, we can find the most probable modification using one insertion action. Currently, only the top 5 insertion actions are applied to modify parts of speech. In the same way, the scores for deleting the i-th part of speech and substituting the i-th part of speech with "x" are defined, respectively, as follows:

$$S_{\mathrm{DEL}}(i; c_1^n) \equiv P(c_{i+1} \mid c_{i-2}, c_{i-1}) \times P(c_{i+2} \mid c_{i-1}, c_{i+1}) \times \prod_{\substack{j=1 \\ j \neq i,\, i+1,\, i+2}}^{n} P(c_j \mid c_{j-2}, c_{j-1});$$

$$S_{\mathrm{SUB}}(i, x; c_1^n) \equiv P(x \mid c_{i-2}, c_{i-1}) \times P(c_{i+1} \mid c_{i-1}, x) \times P(c_{i+2} \mid x, c_{i+1}) \times \prod_{\substack{j=1 \\ j \neq i,\, i+1,\, i+2}}^{n} P(c_j \mid c_{j-2}, c_{j-1}). \tag{17}$$

Again, only the top 5 deletion actions and the top 5 substitution actions are applied to modify the parts of speech. These 15 modified part-of-speech sequences are then parsed to find most likely full parse trees. Finally, the full parse trees are ranked based on the score function $S_{\mathrm{MT}}(\cdot)$ defined in Equation (15).

5.3 Experimental Results and Discussion

To obtain those probability parameters, two training sets were used in the following experiments. The first one, consisting of 8,727 well-formed sentences and their correct parse trees (as described in Section 2.2), was used to estimate the parameters relative to the normal production actions. The second one, consisting of 300 ill-formed sentences and their full parse trees manually annotated with the required modification actions, was used to estimate the parameters relative to the modification actions in Equation (15). As for the testing set, 200 ill-formed sentences were manually parsed into full parse trees using correct modification actions, so that they could be used to test the performance of the proposed error recovery mechanism.

Table 6 lists the experimental results[8] of parsing the 200 ill-formed testing sentences. The first row (PLB) corresponds to the performance of parsing ill-formed sentences without error recovery. The second row (ER1) gives the results of error recovery up to the first stage (i.e., fitting the partial parses only). The last row (ER2) shows the results of error recovery up to the second stage (i.e., fitting the partial parses and also recovering part-of-speech errors).

Table 6. *Performance of parsing ill-formed sentences without and with error recovery in the testing set.*

	Bracket and its label		Parse tree	
	Precision	Recall	Accuracy	Fitting rate
PLB	79.49%	68.49%	—	—
ER1	80.02%	70.59%	25.0%	43.0%
ER2	80.69%	76.60%	35.0%	76.0%

The second row in Table 6 shows that, by fitting the partial parses into the S-production rules, 25% of the ill-formed sentences can be correctly parsed and fitted into full parse trees. The last column indicates that 43% of the ill-formed sentences can be parsed into full parse trees by fitting their partial parses with one or two modification actions. In other words, 18% (the result of subtracting 25% from 43%) of the ill-formed sentences are parsed into incorrect full parse trees.

The last row of Table 6 shows that, using the two-stage error recovery mechanism, 35% of the ill-formed sentences can be correctly parsed into full parse trees. That is, 10% (the result of subtracting 25% from 35%) of the ill-formed sentences are correctly parsed by recovering the part-of-speech errors. After the second stage, 76% of the ill-formed sentences can be fitted into well-formed syntactic structures.

[8] The bracket recall rate listed in Table 6 for PLB should be interpreted differently from that listed in Table 3. The data listed in Table 3 is evaluated based on the brackets of the forests in the original testing set. However, the data listed here is evaluated based on the brackets of the annotated full parse trees obtained by parsing the ill-formed sentences in the testing set using correct modifications. After the errors have been recovered, the testing sentence can be more deeply parsed and, consequently, the number of brackets in its parse tree increases.

Upon carefully inspecting the results, we find that, in the first stage, the improvement in the accuracy rate for parse trees is significant, but the improvement in the bracket recall rate is small. The bracket recall rate is significantly improved only in the second stage. This phenomenon is due to the fact that the errors recovered in the first stage are isolated errors. This kind of error does not hinder the parser from correctly parsing other words. On the contrary, the errors recovered in the second stage are not isolated. They seriously affect the partial parses of other words. Therefore, recovering such errors can significantly improve the bracket recall rate.

In both the first stage and the second stage, some ill-formed sentences are parsed into incorrect full parse trees. The number of incorrect brackets thus increases in both stages. When the precision rate for brackets is computed, the increasing incorrect brackets compensate for the increasing correct brackets, which result from correctly parsing ill-formed sentences. As a result, there is almost no improvement in the bracket precision rates in either the first stage or the second stage.

Although the above experiments were performed based on the English text, we believe that this proposed approach can be applied to other languages as no language-specific assumption is made in the model. Also, nothing that would prevent this model from being applied to other languages has been observed so far.

6. Error Analysis

Although 35% of the ill-formed sentences can be recovered and correctly parsed into full parse trees, there are still many errors which remain unresolved. By inspecting the unrecovered sentences (65% of the test sentences), three different types of errors can be identified and their proportions are listed in Table 7.

About 36% of the unrecoverable sentences have multiple errors. For example, the sentence "*The tutorial explains how and why to use the tool*" can not be covered by our grammar. Our grammar only covers the noun clause with only one question word (i.e., "*why*") followed by an infinitive (i.e., "*to use the tool*"). The desired modification for this sentence is to delete the parts of speech of the words "how and". To recover such errors, multiple modification actions should be employed to modify the parts of speech in the second stage. Because multiple modifications are not allowed in our current system, this sentence is an unrecoverable one.

Table 7. *Error types and their proportions.*

Error type	Proportion
Multiple errors	36%
Syntactic errors	35%
Incorrect modifications	29%

Since the number of different ways to modify a part-of-speech sequence with multiple modification actions is very large, a fast searching method to find the most likely combinations of modification actions is required (if we still want to recover the multiple errors). Besides, applying multiple modification actions, such as two insertions or two deletions, will significantly change the number of parts of speech. However, a full parse tree with fewer parts of speech is usually more likely to have a higher score than one with more parts of speech. Therefore, the normalization issue must be considered to fairly score the full parse trees having different numbers of parts of speech.

Besides the sentences of multiple errors, about 35% of the unresolved errors are caused by syntactic ambiguity. Most of those syntactic errors result from incorrect attachment of prepositional phrases. To eliminate such errors, purely syntactic information is not enough; higher level language knowledge, such as semantic knowledge, should be incorporated.

The final portion of the unrecovered errors result from incorrectly applying modification actions. This accounts for 29% of the errors. Such errors usually occur in the second stage, where modification actions are applied to modify the parts of speech. About half of the errors occur in the situations that the correct modification actions are not included in the top 15 modification actions. To deal with this problem, the discrimination power of the score function, which is used to select the most likely modification actions, should be enhanced. The other half of the errors result from selecting the full parse tree relative to the undesired modification actions. For example, the correct modification action for the ill-formed sentence "*An may appear in the display* " is to insert a noun after the word "*An*". However, the output tree is derived from the undesired modification action, which replaces the part of speech of the word "*An*" with a pronoun. This is because the incorrectly modified full tree, which has fewer parts of speech, is considered to be better than the correctly parsed full tree, which has more parts of speech, from the syntactic point of view. Therefore, such errors can be regarded as being related to "syntactic ambiguity"; and discourse information might be needed to eliminate this kind of error.

In summary, further improvements should be made in the following areas. First, semantic knowledge should be incorporated to eliminate errors resulting from syntactic ambiguity and some errors caused by incorrectly applying modification actions. Second, the discrimination power and speed of the score function for searching potential modification actions should be enhanced. Third, the normalization issue should be considered in order to deal with ill-formed sentences with multiple errors.

7. Conclusion

Parsing ill-formed sentences usually encounters more serious ambiguity problem than parsing the grammatical sentences does. The ambiguities in an ill-formed sentence consist of various partial tree forests, each of which is a combination of partial parses, which jointly generate the input sentence. Since the number of possible forests is very large, it is usually infeasible to enumerate all of them. In the past, the heuristic rule of preferring a larger phrase was used to limit the number of partial parses. However, this heuristic rule, although simple to implement, fails to achieve satisfactory performance because the longest phrase is not always the correct phrase.

This paper has presented a Phrase-Level-Building (PLB) parsing mechanism for handling ill-formed input. In this framework, a parse tree is represented as a set of phrase-levels so that the wide-scope contextual information can be used to efficiently narrow down the search space and accurately choose the desired forest of partial parses. Compared with the baseline system, which uses a stochastic context-free grammar and the "longest leftmost phrase first'" heuristics, the proposed PLB approach improves the precision rate for brackets from 69.37% to 79.49%. The recall rate for brackets is also improved from 78.73% to 81.39%.

Since partial parsing without fixing errors only provides coarse and limited syntactic information about ill-formed sentences, the proposed level-synchronous parsing algorithm is further generalized to recover errors so that more complete syntactic information can be provided. A two-stage strategy has also been proposed efficiently finding the most probable modification actions to recover these errors. The experimental results show that the enhanced parser can correctly recover 35% of ill-formed sentences. The recall rate for brackets is significantly improved from 68.49% to 76.60% while the precision rate for brackets is improved slightly from 79.49% to 80.69%. Although 35% of the sentences can be recovered and correctly parsed into full parse trees, many errors still remain. Further improvements should be made in three areas: incorporating semantic knowledge to eliminate syntactic ambiguity of partial parses, enhancing the discrimination power and speed of the score function to allow more modification actions, and dealing with the normalization problem resulting from multiple errors.

Acknowledgements

This paper is a partial result of both Project NSC-84-2221-E-007-013 sponsored by the National Science Council, R.O.C., and Project 3P11200 conducted by ITRI under the sponsorship of the Ministry of Economic Affairs, R.O.C. The authors would also like to thank the Behavior Design Corporation for providing the English grammar and the corpus.

References

Abney, S. P., "Rapid Incremental Parsing with Repair," in *Proc. of the 6th New OED Conference*: Electronic Text Research, 1990, pp. 1-9.

Abney, S. P., "Parsing by Chunks," in *Principle-Based Parsing: Computation and Psycholinguistics*, Robert C. Berwick, Steven P. Abney, and Carol Tenny (Eds.), Kluwer Academic Publishers,1991, pp. 257-278.

Aho, A. V. and J. D. Ullman, *The Theory of Parsing, Translation, and Compiling*, Vol.1, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

Black E., S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini and T. Strzalkowski, "A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars," in *Proc. of the February 1991 DAPAR Speech and Natural Language Workshop*, pp. 306-311.

Briscoe, T. and J. Carroll, "Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars," *Computational Linguistics*, Vol. 19, No. 1, 1993, pp. 25-59.

Chiang, T.-H., Y.-C. Lin and K.-Y. Su, "Robust Learning, Smoothing and Parameter Tying on Syntactic Ambiguity Resolution," *Computational Linguistics*, Vol. 21, No. 3, 1995, pp. 321-349.

Chiang, T.-H., Y.-C. Lin and K.-Y. Su, "On Jointly Learning the Parameters in a Character-Synchronous Integrated Speech and Language Model," *IEEE Trans. on Speech and Audio Processing*, Vol. 4, No. 3, 1996, pp. 167-189.

Chomsky, N., "On Certain Formal Properties of Grammars," *Information and Control*, Vol. 2, 1959, pp. 137-167.

Church, K. W., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," in *Proc. of ICASSP*, 1989, pp. 695-698.

Doran, C., D. Egedi, B. A. Hockey, B. Srinivas and M. Zaidel, "XTAG System - A Wide Coverage Grammar for English," in *Proc. of the 15th International Conference on Compu-*

tational Linguistics, 1994, pp. 922-928.

Fujisaki, T., F. Jelinek, J. Cocke, E. Black and T. Nishino, "A Probabilistic Parsing Method for Sentence Disambiguation," in *Proc. of the International Workshop on Parsing Technologies*, 1989, pp. 85-94.

Good, I. J., "The Population Frequencies of Species and the Estimation of Population Parameters," *Biometrika*, 40, 1953, pp. 237-264.

Hobbs, J. R., D. E. Appelt, J. Bear, and M. Tyson, "Robust Processing of Real-World Natural-Lanugage Texts," in *Proc. of the Third Conference on Applied Natural Language Processing*, 1992, pp. 186-192.

Hopcroft, J. and J. D. Ullman, *Formal Language and Their Relation to Automata*, Addison-Wesley, 1976.

Hutchins, W. J., *Machine Translation: Past, Present, Future*, West Sussex, England: Ellis Horwood Limited, 1986.

Jensen, K., G. E. Heidorn L. A. Miller, and Y. Ravin., "Parse Fitting and Prose Fixing: Getting a Hold on Ill-formedness," *American Journal of Computa-tion-al Linguistics*, Vol. 9, No. 3-4, 1983, pp. 147-160.

Kwasny, S. C. and N. K. Sondheimer, "Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems," *American Journal of Computational Linguistics*, Vol. 7, No. 2, 1981, pp. 99-108.

Lee, K. J., C. J. Kweon, J. Seo, and G. C. Kim., "A Robust Parser Based on Syntactic Information," in Proc. of *the 7th Conference of European Chapter of the Association for Computational Linguistics*, 1995, pp. 223-228.

Lin, Y.-C., T.-H. Chiang and K.-Y. Su, "The Effects of Learning, Parameter tying and Model Refinement for Improving Probabilistic Tagging," *Computer Speech and Language*, Vol. 9, 1995, pp. 37-61.

Lyon, G., "Syntax-Directed Least-Errors Analysis for Context-Free Languages," *Communications of the ACM*, Vol. 17, No. 1, 1974, pp. 3-14.

Magerman, D. M., "Statistical Decision-Tree Models for Parsing," in *Proc. of the 33rd Annual Meeting of the Association for Computa-tional Linguistics*, 1995, pp. 276-283.

McDonald, D. D., "An Efficient Chart-Based Algorithm for Partial-Parsing of unrestricted Texts," in Proc. of the Third Conference on Applied Natural Language Processing, 1992, pp. 193-200.

Mellish, C. S., "Some Chart-Based Techniques for Parsing Ill-Formed Input," in *Proc. of the 27th Annual Meeting of the Association for Computa-tional Linguistics*, 1989, pp. 102-109.

Meteer, M. and H. Gish, "Integrating Symbolic and Statistical Approaches in Speech and Natural Language Applications," in *Proc. of the Workshop on the Balancing Act Combining Symbolic and Statistical Approaches to Language*, 1994, pp. 69-75.

Ng, S.-K. and M. Tomita, "Probabilistic LR Parsing for General Context-Free Grammars," in *Proc. of the Second International Workshop on Parsing Technologies*, 1991, pp. 154-163.

Seneff, S., "Robust Parsing for Spoken Language System," in *Proc. of ICASSP*, 1992, pp. 189-192.

Skut, W. and T. Brants, "Chunk Tagger - Statistical Recognition of Noun Phrases," in *Proc. of ESSLLI-98 Workshop on Automated Acquistion of Syntax and Parsing*, 1998.

Su, K.-Y., T.-H. Chiang, and Y.-C. Lin, "A Unified Probabilistic Score Function for Integrating Speech and Language Information in Spoken Language Processing," in *Proc. of ICSLP*, 1990, pp. 901-904.

Su, K.-Y. and J.-S. Chang, "Some Key Issues in Designing MT System," *Machine Translation*, Vol. 5, No. 4, 1990, pp. 265-300.

Tomita, M., "An Efficient Augmented-Context-Free Parsing Algorithm," *Computational Linguistics*, Vol. 13, No. 1-2, 1987, pp. 31-46.

Weischedel, R. M. and N. K. Sondheimer, "Meta-Rules as a Basis for Processing Ill-Formed Input," *American Journal of Computational Linguistics*, Vol. 9, No. 3-4, 1983, pp. 161-177.

Appendix A: Descriptions of Grammar Symbols

Terminals:

`a`	Adjective
`be`	Verb "be" used as an auxiliary
`comp`	Complementizer
`modl`	Modal
`n`	Noun
`p`	Preposition
`quan`	Quantifier
`v`	Verb

Nonterminals

`ADTC`	Adjunct used to modify a verb phrase
`AUX`	Auxiliary phrase
`N*`	A list of nouns (or noun phrases) in recursive form
`1`	Noun phrase (level 1)
`N3`	Noun phrase (level 3)
`NLM*`	A list of left modifiers for a noun phrase (in recursive form)
`P*`	A list of prepositions (in recursive form)
`S`	Sentence (the start symbol of the grammar)
`V1`	Verb phrase (level 1)
`V2`	Verb phrase (level 2)
`VN`	The noun phrase which is an argument of a verb phrase

Appendix B: Derivation of the Score Function for Modified Parse Trees

As stated in Section 4, a modified parse tree $\tilde{\mathbf{T}}$ of N phrase-levels can be represented as

$$\tilde{\mathbf{T}} \equiv \{\mathbf{L}_1, \tilde{\mathbf{R}}_1, \tilde{\mathbf{L}}_1, \mathbf{R}_1, \mathbf{L}_2, \cdots, \mathbf{L}_{N-1}, \tilde{\mathbf{R}}_{N-1}, \tilde{\mathbf{L}}_{N-1}, \mathbf{R}_{N-1}, \mathbf{L}_N\},$$

where $\mathbf{L}_i$ is the i-th phrase-level, $\tilde{\mathbf{R}}_i$ is the set of modification actions used to modify $\mathbf{L}_i$, $\tilde{\mathbf{L}}_i$ is the result after applying $\tilde{\mathbf{R}}_i$ to modify $\mathbf{L}_i$, and $\mathbf{R}_i$ is the set of normal actions applied to build $\mathbf{L}_{i+1}$ from $\tilde{\mathbf{L}}_i$. Similar to Equation (1), the likelihood of a modified parse tree $\tilde{\mathbf{T}}$ of N phrase-levels is derived as

$$\begin{aligned} P(\tilde{\mathbf{T}} \mid w_1^n) &= P(\mathbf{L}_1, \tilde{\mathbf{R}}_1, \tilde{\mathbf{L}}_1, \mathbf{R}_1, \mathbf{L}_2, \cdots, \mathbf{L}_{N-1}, \tilde{\mathbf{R}}_{N-1}, \tilde{\mathbf{L}}_{N-1}, \mathbf{R}_{N-1}, \mathbf{L}_N \mid \mathbf{L}_1) \\ &= P(\mathbf{L}_N, \mathbf{R}_{N-1}, \tilde{\mathbf{L}}_{N-1}, \tilde{\mathbf{R}}_{N-1} \mid \mathbf{L}_{N-1}, \mathbf{R}_{N-2}, \tilde{\mathbf{L}}_{N-2}, \tilde{\mathbf{R}}_{N-2}, \cdots, \mathbf{L}_2, \mathbf{R}_1, \tilde{\mathbf{L}}_1, \tilde{\mathbf{R}}_1, \mathbf{L}_1) \\ &\quad \times P(\mathbf{L}_{N-1}, \mathbf{R}_{N-2}, \tilde{\mathbf{L}}_{N-2}, \tilde{\mathbf{R}}_{N-2} \mid \mathbf{L}_{N-2}, \mathbf{R}_{N-3}, \tilde{\mathbf{L}}_{N-3}, \tilde{\mathbf{R}}_{N-3}, \cdots, \mathbf{L}_2, \mathbf{R}_1, \tilde{\mathbf{L}}_1, \tilde{\mathbf{R}}_1, \mathbf{L}_1) \\ &\quad \times \cdots \times P(\mathbf{L}_2, \mathbf{R}_1, \tilde{\mathbf{L}}_1, \tilde{\mathbf{R}}_1 \mid \mathbf{L}_1) \\ &= \prod_{i=2}^{N} P(\mathbf{L}_i, \mathbf{R}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \tilde{\mathbf{R}}_{i-1} \mid \mathbf{L}_{i-1}, \mathbf{R}_{i-2}, \tilde{\mathbf{L}}_{i-2}, \tilde{\mathbf{R}}_{i-2}, \cdots, \mathbf{L}_2, \mathbf{R}_1, \tilde{\mathbf{L}}_1, \tilde{\mathbf{R}}_1, \mathbf{L}_1). \end{aligned} \tag{18}$$

Assume that the modification action $\tilde{\mathbf{R}}_{i-1}$ and the normal action $\mathbf{R}_{i-1}$, which will be applied to build $\mathbf{L}_{i-1}$ into $\mathbf{L}_i$, only depend on $\mathbf{L}_{i-1}$. Then, the above equation can be approximated as

$$\begin{aligned} P(\tilde{\mathbf{T}} \mid w_1^n) &= \prod_{i=2}^{N} P(\mathbf{L}_i, \mathbf{R}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \tilde{\mathbf{R}}_{i-1} \mid \mathbf{L}_{i-1}, \mathbf{R}_{i-2}, \tilde{\mathbf{L}}_{i-2}, \tilde{\mathbf{R}}_{i-2}, \cdots, \mathbf{L}_2, \mathbf{R}_1, \tilde{\mathbf{L}}_1, \tilde{\mathbf{R}}_1, \mathbf{L}_1) \\ &\approx \prod_{i=2}^{N} P(\mathbf{L}_i, \mathbf{R}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \tilde{\mathbf{R}}_{i-1} \mid \mathbf{L}_{i-1}) = \prod_{i=2}^{N} \left\{ P(\mathbf{L}_{i-1}, \tilde{\mathbf{R}}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1} \mid \mathbf{L}_i) \frac{P(\mathbf{L}_i)}{P(\mathbf{L}_{i-1})} \right\} \\ &= \frac{P(\mathbf{L}_N)}{P(\mathbf{L}_1)} \prod_{i=2}^{N} P(\mathbf{L}_{i-1}, \tilde{\mathbf{R}}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1} \mid \mathbf{L}_i) = \frac{P(\mathbf{L}_N)}{P(\mathbf{L}_1)} \prod_{i=2}^{N} P(\tilde{\mathbf{R}}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1} \mid \mathbf{L}_i). \end{aligned} \tag{19}$$

Note that the last equality in the above equation holds because $\mathbf{L}_{i-1}$ is uniquely determined by $\tilde{\mathbf{R}}_{i-1}$, $\tilde{\mathbf{L}}_{i-1}$, $\mathbf{R}_{i-1}$ and $\mathbf{L}_i$. The last probability term in the above equation can be further derived as

$$\begin{aligned} P(\tilde{\mathbf{R}}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1} \mid \mathbf{L}_i) &= P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1}, \mathbf{L}_i)\, P(\tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1} \mid \mathbf{L}_i) \\ &= P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1}, \mathbf{L}_i)\, P(\mathbf{R}_{i-1} \mid \mathbf{L}_i). \end{aligned} \tag{20}$$

Again, the last equality in the above equation holds because $\tilde{\mathbf{L}}_{i-1}$ is uniquely determined by $\mathbf{R}_{i-1}$ and $\mathbf{L}_i$. Furthermore, assuming that the modification action $\tilde{\mathbf{R}}_{i-1}$ only depends on $\tilde{\mathbf{L}}_{i-1}$, we can approximate $P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1}, \mathbf{L}_i)$ as $P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1})$. Consequently, we can obtain

$$P(\tilde{\mathbf{R}}_{i-1}, \tilde{\mathbf{L}}_{i-1}, \mathbf{R}_{i-1} \mid \mathbf{L}_i) \approx P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1})\, P(\mathbf{R}_{i-1} \mid \mathbf{L}_i)\,. \tag{21}$$

The probability factor $P(\mathbf{R}_{i-1} | \mathbf{L}_i)$ can be further approximated by Equation (4). The other probability factor $P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1})$, which is related to the modification actions, is computed based on whether $\tilde{\mathbf{R}}_{i-1}$ is an empty set or not. If $\tilde{\mathbf{R}}_{i-1}$ is an empty set, then $P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1})$ is expressed as

$$P\left(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1}\right) = P\left(\tilde{\mathbf{R}}_{i-1} = \phi \mid \tilde{\mathbf{L}}_{i-1}\right). \tag{22}$$

For simplicity, we assume that the even $\tilde{\mathbf{R}}_{i-1} = \phi$ is independent of the event $\tilde{\mathbf{L}}_{i-1}$. Then $P\left(\tilde{\mathbf{R}}_{i-1} = \phi \mid \tilde{\mathbf{L}}_{i-1}\right)$ can be set to $P\left(\tilde{\mathbf{R}}_{i-1} = \phi\right)$. Similarly, if $\tilde{\mathbf{R}}_{i-1}$ is not an empty set, then $P(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1})$ is computed as

$$\begin{aligned} P\left(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{L}}_{i-1}\right) &= P\left(\tilde{\mathbf{R}}_{i-1}, \tilde{\mathbf{R}}_{i-1} \neq \phi \mid \tilde{\mathbf{L}}_{i-1}\right) \\ &= P\left(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{R}}_{i-1} \neq \phi, \tilde{\mathbf{L}}_{i-1}\right) P\left(\tilde{\mathbf{R}}_{i-1} \neq \phi \mid \tilde{\mathbf{L}}_{i-1}\right) \\ &\approx P\left(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{R}}_{i-1} \neq \phi, \tilde{\mathbf{L}}_{i-1}\right) P\left(\tilde{\mathbf{R}}_{i-1} \neq \phi\right). \end{aligned} \tag{23}$$

Once it is decided that a phrase-level is to be modified, which actions will be applied is assumed to depend on the contextual information in the modified phrase-level. Therefore, the term $P\left(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{R}}_{i-1} \neq \phi, \tilde{\mathbf{L}}_{i-1}\right)$ is further derived as follows:

$$\begin{aligned} P\left(\tilde{\mathbf{R}}_{i-1} \mid \tilde{\mathbf{R}}_{i-1} \neq \phi, \tilde{\mathbf{L}}_{i-1}\right) &= P\left(\tilde{\rho}_1^m \mid \tilde{A}_1^n\right) \\ &= \prod_{j=1}^{m} P\left(\tilde{\rho}_j \mid \tilde{\rho}_1^{j-1}, \tilde{A}_1^n\right) \approx \prod_{\tilde{\rho}=\langle \tilde{r};u,v\rangle \in \tilde{\mathbf{R}}_{i-1}} P\left(\tilde{r} \mid \tilde{A}_u^v\right), \end{aligned} \tag{24}$$

where $m \neq 0$, and the modification action $\tilde{\rho} = \langle \tilde{r}; u, v\rangle$ is assumed to depend on the its local context $\tilde{A}_u, \cdots \tilde{A}_v$, (i.e., the symbols from the u-th position to the v-th position in $\tilde{\mathbf{L}}_i$). The above parameter $P(\tilde{r} \mid \tilde{A}_u^v)$ is estimated from the modified phrase-levels obtained by applying modification actions.

According to Equations (18) - (24), the likelihood of a modified parse tree is, thus, approximated as

$$\begin{aligned} P\left(\tilde{\mathbf{T}} \mid w_1^n\right) \approx & \frac{1}{P(\mathbf{L}_1)} \times \prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho=<r;t> \in \mathbf{R}_i} P(r \mid A_{t-1}^{t+1}) \\ & \times \prod_{\tilde{\mathbf{R}}_i=\phi} P(\tilde{\mathbf{R}}_i = \phi) \times \prod_{\tilde{\mathbf{R}}_i \neq \phi} \left\{ P(\tilde{\mathbf{R}}_i \neq \phi) \times \prod_{\tilde{\rho}=<\tilde{r};u,v> \in \tilde{\mathbf{R}}_i} P(\tilde{r} \mid \tilde{A}_u^v) \right\}. \end{aligned} \tag{25}$$

Since $P(\mathbf{L}_1)$ is the priori probability of the input sentence, it is the same for all competing modified parse trees and, thus, can be ignored without changing the ranking order of the likelihood values of the competing forests. Therefore, the scoring function $S_{\mathrm{MT}}(\cdot)$ used to rank the modified parse trees is defined as follows:

$$\begin{aligned} S_{\mathrm{MT}}\left(\tilde{\mathbf{T}}\right) \equiv & \prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho=<r;t> \in \mathbf{R}_i} P(r \mid A_{t-1}^{t+1}) \\ & \times \prod_{\tilde{\mathbf{R}}_i=\phi} P(\tilde{\mathbf{R}}_i = \phi) \times \prod_{\tilde{\mathbf{R}}_i \neq \phi} \left\{ P(\tilde{\mathbf{R}}_i \neq \phi) \times \prod_{\tilde{\rho}=<\tilde{r};u,v> \in \tilde{\mathbf{R}}_i} P(\tilde{r} \mid \tilde{A}_u^v) \right\}, \end{aligned} \tag{26}$$

where the subscript "MT" in $S_{\mathrm{MT}}(\cdot)$ means "modified tree".