A Level-synchronous Approach to Ill-formed Sentence Parsing

Yi-Chung Lin* and Keh-Yih Su†

* Advanced Technology Center, Computer and Communication Research Lab., Industrial Technology Research Institute, Hsinchu, Taiwan 310, R.O.C.
lyc@e0sun3.ccl.irti.org.tw

† Department of Electrical Engineering, National Tsing-Hua University
Hsinchu, Taiwan 300, R.O.C.
kysu@bdc.com.tw

Abstract

In this paper, a *Phrase-Level-Building* (PLB) mechanism is proposed to parse ill-formed sentences. By decomposing a syntactic tree into phrase-levels, this mechanism regards the task of parsing a sentence as a task of building the phrase-levels for the sentence. During parsing, a level-synchronous scoring function is used to remove less likely phrase-levels. As a result, instead of enumerating all possible parses, the PLB parser only generates the more likely *tree groups*, each of which is a set of partial parses jointly deriving the input. Whenever all active phrase-levels in the search beam cannot be further reduced by any grammar rules, the process of building phrase-levels is stopped and a probabilistic scoring function is used to select the best tree group. With this approach, the best tree group is selected within a wider scope (i.e., the whole sentence), and thus generates better result. Compared with the baseline system using the stochastic context-free grammar and the "leftmost longest phrase first" heuristics (which operates in a narrow scope), the proposed PLB approach improves the precision of brackets in the tree group from 69.37% to 79.49%. The recall of brackets is also improved from 78.73% to 81.39%.

1 Introduction

Natural language parsing plays an important role in various applications of natural language processing (NLP), 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 relationship of the words in an input sentence according to a given grammar which formally specifies the allowable syntactic structures in the language. In real applications, to correctly parse a sentence, a parser often encounters the problems resulted from the ambiguities in syntactic structure and the ill-formedness of the inputs.

Ambiguous syntactic structures are generated due to the implied ambiguity from language usage or due to the over-generation from the given grammar. The number of syntactic ambiguities of a sentence depends on the grammar. In practical applications, a sentence usually has thousands of ambiguities, and, on some occasions, the number of ambiguities may be greater than millions. To give a correct interpretation for the input sentence, a natural language parser must be able to choose the correct syntactic structure from such ambiguities. In the past, many algorithms have been proposed to resolve this problem and significant improvements have been observed (Briscoe and Carroll, 1993; Chiang, Lin, and Su, 1995).

The other problem in natural language parsing is the ill-formedness of inputs. An ill-formed sentence is the sentence that cannot be fitted into any well-formed syntactic structures generated by the grammar. The major sources of ill-formed inputs are (1) incorrect sentences resulted from typographical errors, OCR scanning etc, (2) unknown words which are not contained by the system dictionary and (3) the insufficient coverage of the grammar. Compared with the topic of syntactic disambiguation, the problem of ill-formed inputs is less investigated and is often ignored in experiment works. However, ill-formed inputs are inevitable in real applications because the incorrect sentences always exist in the real world and it is impossible to limit users using only the predefined artificial grammar and built-in vocabulary. Therefore, we focus on the problem of handling ill-formed sentences here.

In this paper, the *Phrase-Level-Building* (PLB) parsing mechanism is proposed to attack the ambiguity problem of ill-formed inputs by consulting the contextual information. In this framework, a parse tree is modeled as a set of *phrase-levels*. By decomposing a syntactic tree into phrase-levels, this mechanism regards the task of parsing a sentence as a task of building the phrase-levels for the sentence. A phrase-level here refers a set of terminals and nonterminals constructed at a particular snap shot of the parsing process. For example, the parser may construct a noun phrase "[N3 *Printer buffers*]" and a verb phrase "[V2 *are made by DRAM*]" for the sentence *"Printer buffers are made by DRAM"*. In this case, the phrase-level at this particular time consists of "[N3 *Printer buffers*]" and "[V2 *are made by DRAM*]". During parsing, a fast level-synchronous search mechanism is used to remove less likely phrase-levels. As a result, only the tree groups with large likelihood values are generated by the PLB parser. Whenever all active phrase-levels in the search beam cannot be further reduced by any grammar rules, the process of building phrase-levels is stopped and a probabilistic scoring function is used to select the best tree group. With this approach, the best tree group is selected within a wider scope (i.e., the whole sentence), and thus generates better result. Compared with the baseline system using the stochastic context-free grammar and the "leftmost longest phrase first" heuristics (which operates in a narrow scope), the proposed PLB approach improves the precision of brackets in the tree group from 69.37% to 79.49%. The recall of brackets is also improved from 78.73% to 81.39%.

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 alone or with other system-dependent heuristics to further constraint the possible partial parses while parsing the ill-formed sentences. On the other hand, in some systems, natural language sentences are parsed by a left-corner parser (such as the LR parser), which uses 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. Thus, these systems usually parse the ill-formed input with the heuristics of selecting the leftmost longest phrase and then starting to parse from the subsequent word again. To make a comparative study, a baseline system is built to evaluate the performances of these two heuristic rules, *leftmost longest phrase first* (LLF) and *longest phrase first* (LF), on handling ill-formed sentences.

The baseline system consists of two components: a Cocke-Younger-Kasami (CYK) parser and a partial parse assembler. According to the Chomsky normal form (CNF) grammar (Chomsky, 1959), which is converted from a normal context-free grammar (CFG), the CYK parser efficiently parse the ill-formed sentence to all its possible partial parses; and every combination of the partial parses which covers all terminals will form a tree group for the ill-formed sentence. Then, according to the adopted heuristic rule (LF or LLF), the partial parses are assembled by the partial parse assembler, in which the partial parses cover the same input words are ranked by the stochastic context-free grammar (SCFG) (Fujisaki et al., 1989; Ng and Tomita, 1991) with the probability parameters smoothed by the Good-Turing method (Good, 1953; Katz, 1987).

2.1 Evaluation Method

The performances of different approaches are measured by three factors: bracket precision, bracket recall and tree group accuracy. The parse tree in Figure 1 is used as an example to explain how to calculate the precision and recall for brackets. This parse tree has nine

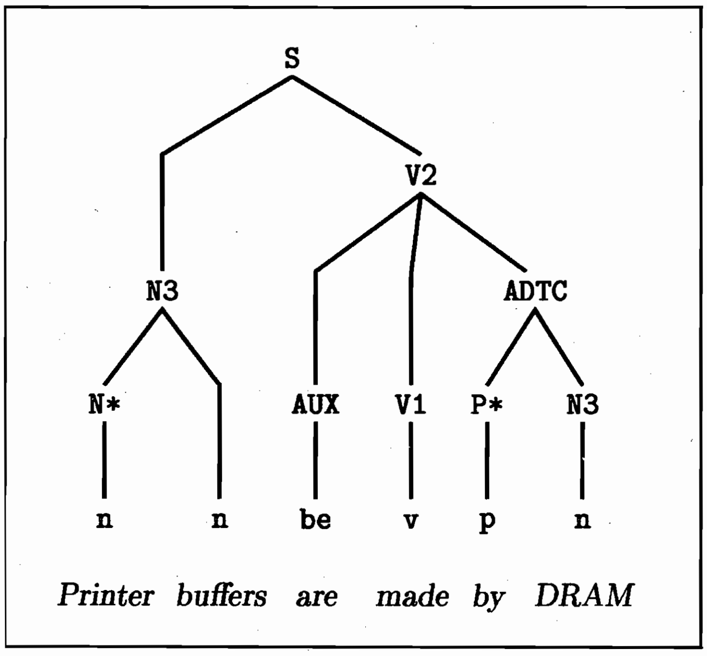

Figure 1: An example of the parse tree.

brackets shown below,

[N* *Printer*], [N3 *Printer buffers*], [AUX *are*],
[V1 *made*], [P* *by*], [N3 *DRAM*], [ADTC *by DRAM*],
[V2 *are made by DRAM*], [S *Printer buffers are made by DRAM*].

Each of the brackets corresponds to the application of a production (shown in a left-to-right depth-first traversal sequence).

The precision rate and the recall rate are computed as follows.

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

and

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

It should be noticed that in many applications the grammar of a system to be measured may differ from the one used to parse the treebank (i.e. the database of correct parse tree). Therefore, the labels of the brackets are not taken into account in computing the precision or recall of brackets in most cases. However, in this task, both the system and the treebank use the same grammar. Thus, the labels of brackets are also taken into account while computing the precision and recall for brackets. In other words, we will regard two brackets as being "matched" only when they have the same label.

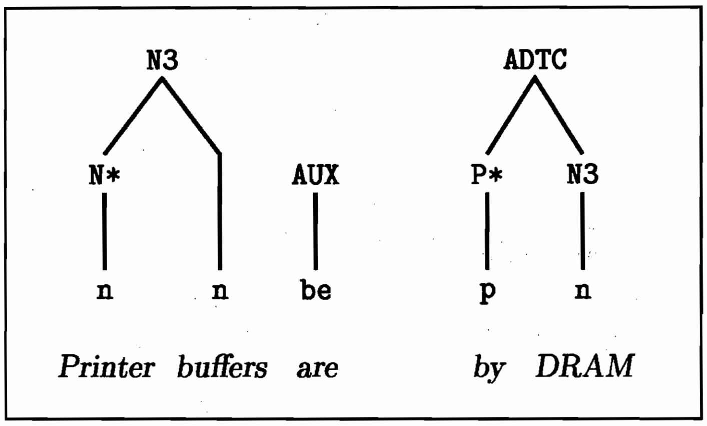

Figure 2: An example of the tree group.

Since the performance of a robust parser strongly depends on whether the parser can accurately partition the inputs or not, the factors "fragment precision" and "fragment recall" are also measured to reflect the performance of a robust parser on partitioning the ill-formed sentences. Here, fragments are the brackets which are not enclosed by any other brackets, i.e. the outmost brackets. For example, for the ill-formed sentence "*Printer buffers are by DRAM*" with the tree group in Figure 2, its three fragments are

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

On the other hand, since an ill-formed sentence cannot be parsed to a full parse tree, the conventional parse tree accuracy is replaced with the tree group accuracy, which is computed as

$$\text{tree group accuracy} = \frac{\text{number of exactly matched tree groups}}{\text{number of sentences}},$$

where "exactly matched tree groups" means that all the tree groups consist of the same partial parses.

2.2 Simulation Results and Discussions

In the baseline system, 8,727 well-formed sentences, collected from computer manuals, and their correct parse trees are used as the training data. The average length of these sentences is about 13 words. All the training sentences are parsed by a context-free grammar 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 tree groups are used as the testing data. The average length of the testing sentences is about 13 words.

	Bracket and its label		Tree group 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

Table 1: The performances of the baseline system with "longest phrase first" (LF) and "leftmost longest phrase first" (LLF) heuristics.

Table 1 lists the simulation results with different heuristic rules. The precision and recall of the labeled brackets are given in the first and second columns. The third column shows the accuracy rate of tree group and the last column gives the average processing time for parsing a sentence with a "SUN SPARC station ELC". The experiment results show that LLF slightly outperforms LF. This is because the LF, compared with the LLF heuristics, is more likely to grab the words belonging to the neighboring phrases. In fact, due to preferring a larger partial parse than a smaller one, the LF heuristics always

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

Table 2: The performances of LLF and LF on fragments.

partitions a sentence into as few fragments as possible. As shown in Table 2, the number of fragments generated by using the LLF heuristics is only 355, which is much smaller than that of the correct fragments. But, the number of fragments generated by using the LF heuristics is even smaller. In other words, assembling partial parses with the LF heuristics produces more inadequate partitions than with the LLF heuristics.

In fact, both the LLF and LF heuristic rules use rather coarse knowledge to assemble partial parses. They always append the largest partial parses, either the leftmost one or a global one, to the tree group, regardless of the context of the partial parse. Therefore, the performance is not really satisfactory. In the next section, a Phrase-Level-Building parsing algorithm is proposed to parse the ill-formed inputs by consulting more contextual information.

3 PLB Parsing

In this section, a *Phrase-Level-Building* (PLB) parsing algorithm is proposed to parse an ill-formed sentence using contextual information in wider scope. This algorithm treats the parsing process as the procedure of building a set of *phrase-levels*. During parsing, a fast level-synchronous search mechanism is used to cut down the search space. Instead of using heuristics, the final parse trees are ranked by a probabilistic scoring function which makes use of the contextual information in the phrase-levels. The details of this algorithm is described in the following sections.

3.1 Phrase-Levels of a Parse Tree

The basic idea of PLB parsing is to model a syntactic tree as a set of phrase-levels. Figure 3 is an example to show the relations between a syntactic tree and its phrase-levels. As shown in this figure, the syntactic tree for the sentence "*Printer buffers are made by DRAM*" is decomposed into six phrase-levels. The lowest one, $\mathbf{L}_1$, corresponds to the input

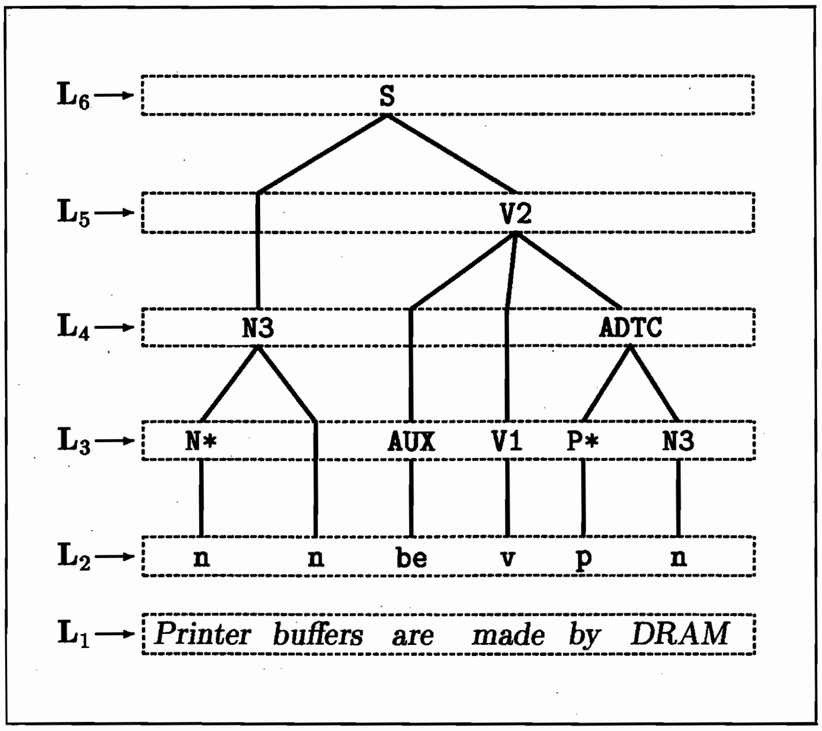

Figure 3: A syntactic tree and its phrase-levels.

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) which are obtained by applying some grammar rules on the grammar symbols of the previous phrase-level. As a result, the parsing process can be considered as the procedure of building the phrase-levels from $\mathbf{L}_1$ to $\mathbf{L}_6$ in a bottom-up manner.

Every phrase-level in a parse tree is obtained by applying some production rules on the phrase-level immediately preceding the current phrase-level. In Figure 3 $\mathbf{L}_3$ has six grammar symbols and is denoted as $\mathbf{L}_3 = \{$ `N* n AUX V1 P* N3` $\}$. By applying the productions "`N3` $\rightarrow$ `N* n`" and "`ADTC` $\rightarrow$ `P* N3`" on the leftmost two and the rightmost two grammar symbols respectively, $\mathbf{L}_3$ is built up to $\mathbf{L}_4$, which has four grammar symbols and is denoted as $\mathbf{L}_4 = \{$ `N3 AUX V1 ADTC` $\}$. As a result, the parse tree in Figure 3 can be represented as $\mathbf{T} = \{\mathbf{L}_1, \mathbf{R}_1, \mathbf{L}_2, \mathbf{R}_2, \cdots, \mathbf{L}_5, \mathbf{R}_5, \mathbf{L}_6\}$, where $\mathbf{R}_i$ denotes a sequence of actions which are applied to build $\mathbf{L}_i$ up to $\mathbf{L}_{i+1}$. For example, to build $\mathbf{L}_4$ from $\mathbf{L}_3$ in Figure 3, $\mathbf{R}_3$ contains two actions, which corresponds to applying the productions "`N3` $\rightarrow$ `N* n`" and "`ADTC` $\rightarrow$ `P* N3`" on the leftmost two and the rightmost two grammar symbols in $\mathbf{L}_3$. The detailed definition of action will be described in the following section.

3.2 Scoring a Parse Tree

The likelihood of the parse tree of N phrase-levels can be derived as follows.

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

In the above equation, the prior probability $P(\mathbf{L}_N)$ is introduced by applying the Bayesian formula because $P(\mathbf{L}_N)$ is regarded as useful information to assemble partial parses to a tree group. For example, the likelihood of the tree group in Figure 2 is considered related to the likelihood of a nonterminal sequence "`N3 AUX ADTC`".

Note that $P(\mathbf{R}_{i-1}, \mathbf{L}_{i-1} | \mathbf{L}_i) = P(\mathbf{R}_{i-1} | \mathbf{L}_i)$ because $\mathbf{L}_{i-1}$ is uniquely determined by $\mathbf{R}_{i-1}$ and $\mathbf{L}_i$. Therefore, the Equation 1 can be written as

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

Since $P(\mathbf{L}_1)$ is the prior probability of the input sentence, it is the same for all competing parse trees and can be ignored without changing the ranking order of the likelihood values of the competing parse trees. Suppose there are n symbols $\{A_1, \cdots, A_n\}$ in the phrase-level $\mathbf{L}_N$, the probability $P(\mathbf{L}_N)$ can be approximated by a trigram model as follows.

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

The probability term $P(\mathbf{R}_{i-1}|\mathbf{L}_i)$ in Equation (2) accounts for the actions which are applied to build $\mathbf{L}_i$ from $\mathbf{L}_{i-1}$. Alternatively, it could be regarded as the probability of applying the rewriting rules in $\mathbf{R}_{i-1}$ at the phrase-level $\mathbf{L}_i$ from a top-down point of view. Before deriving $P(\mathbf{R}_{i-1}|\mathbf{L}_i)$, the notations for the actions of $\mathbf{R}_{i-1}$ will be defined first. Let $\{\rho_1, \cdots, \rho_m\}$ denote the m actions of $\mathbf{R}_{i-1}$. The j-th action ρ_j will be denoted as $\rho_j = \langle r; t\rangle$, where the rule argument r denotes the rule applied by ρ_j; and the position argument t is the index of the reduced symbol in $\mathbf{L}_i$. For example, to build $\mathbf{L}_4$ from $\mathbf{L}_3$ in Figure 3, the production rules " `N3` $\rightarrow$ `N* n` " and " `ADTC` $\rightarrow$ `P* N3` " are applied respectively. In this case, the corresponding actions are $\rho_1 = \langle r = \texttt{N3} \rightarrow \texttt{N* n}; t = 1\rangle$ and $\rho_2 = \langle r = \texttt{ADTC} \rightarrow \texttt{P* N3}; t = 4\rangle$. The reduced symbols are `N3` and `ADTC`, which are the 1st and 4th symbols in $\mathbf{L}_4$ respectively. Therefore, the position arguments t of these two actions are 1 and 4 respectively. Figure 4 gives a more clear illustration for the relationship of those phrase-levels and actions.

$\mathbf{L}_4$	$= \{$ `N3`	`AUX V1`	`ADTC`	$\}$
$\mathbf{R}_3$	$= \{ \rho_1$		ρ_2	$\}$
$\mathbf{L}_3$	$= \{$ `N* n`	`AUX V1`	`P* N3`	$\}$

$$\rho_1 = \langle\, \texttt{N3} \rightarrow \texttt{N* n} \;;\, 1 \,\rangle$$
$$\rho_2 = \langle\, \texttt{ADTC} \rightarrow \texttt{P* N3} \;;\, 4 \,\rangle$$

Figure 4: Two phrase-levels and their corresponding actions.

With these notations, the conditional probability $P(\mathbf{R}_{i-1}|\mathbf{L}_i)$ can be derived as follows. Let $A_1^n \equiv \{A_1, \cdots, A_n\}$ denote the n symbols in $\mathbf{L}_i$ and $\rho_1^m \equiv \{\rho_1, \cdots, \rho_m\}$ denote the m actions in $\mathbf{R}_{i-1}$. Then, the conditional probability $P(\mathbf{R}_{i-1}|\mathbf{L}_i)$ can be approximated as

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

where we assume that the action $\rho_j = \langle r; t\rangle$ depends on the its local context $A_{t-1}, \cdots, A_{t+1}$.

According to Equations (2)-(4), the likelihood of a parse tree is approximated as follows.

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

Note that, in the above equation, the notation $A_{i+1,t-1}^{i+1,t+1}$ denotes the sequence "$A_{i+1,t-1}$ $A_{i+1,t}$ $A_{i+1,t+1}$", which represents the $(t-1)$-th symbol, the t-th symbol and the $(t+1)$-th symbol in $\mathbf{L}_{i+1}$ respectively. These three symbols are the local context of an action $\rho = \langle r; t \rangle$ in $\mathbf{R}_i$. As mentioned before, the probability $P(\mathbf{L}_1)$ can be ignored while ranking the likelihoods of parse trees because it is the same for all competing parse trees. Therefore, the parse tree scoring function $S_{\mathrm{PT}}(\cdot)$ is defined as follows.

$$S_{\mathrm{PT}}(\mathbf{T}|w_1^n) \equiv \prod_{A_j \in \mathbf{L}_N} P(A_j | A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho=\langle r;t\rangle \in \mathbf{R}_i} P\left(r | A_{i+1,t-1}^{i+1,t+1}\right) \qquad (6)$$

The parameters in this scoring function are smoothed by the Good-Turing smoothing method.

3.3 The PLB Parsing Mechanism

The PLB parser parses an input sentence as building the phrase-levels for the sentence. The building process is illustrated in Figure 5, where we assume there are only two ambiguities for every phrase-level candidate while expanding it up to a higher level. As shown in Figure 5, up to the 4th phrase-level, there are eight different *partial trees* (i.e. tree groups), each of which consists of four phrase-levels and is represented by one of the eight paths. Since the number of paths increases exponentially, it is infeasible to exhaustively travel all possible paths during parsing. Thus, the beam search is adopted to find the most likely paths.

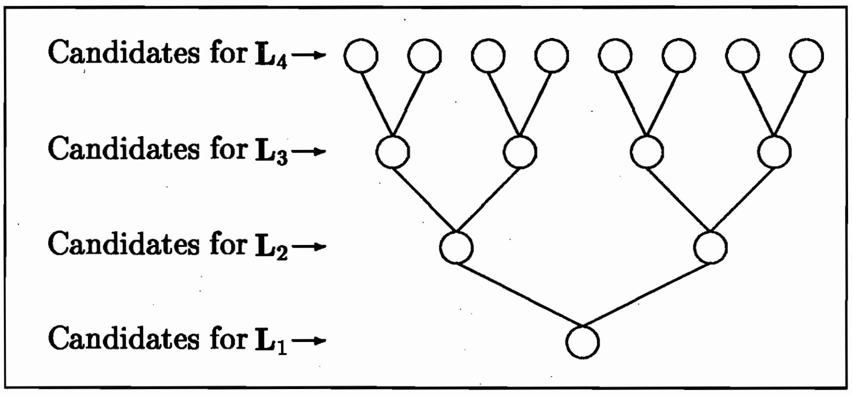
Candidates for $\mathbf{L}_4$
Candidates for $\mathbf{L}_3$
Candidates for $\mathbf{L}_2$
Candidates for $\mathbf{L}_1$

Figure 5: An illustration of building the phrase-levels up.

To efficiently carry out the beam search, we require a scoring function which can rank the candidates of every phrase-level in very short time. However, the scoring function in Equation (6) will spend too much computation time in ranking the candidates of a phrase-level because all possible candidates of the phrase-level must be expanded and scored. Thus, a time-saving scoring function is proposed in this section to rapidly find the potential candidates of a phrase-level, and Equation (6) will be used only after the final level is reached. In other words, two different scoring functions are used during the parsing process and during the final best tree selection process, respectively.

To make the derivation of the scoring function more clear, another representation form of parse trees is introduced in the following. From another point of view, the process of building a phrase-level $\mathbf{L}_i$ up to a higher phrase-level $\mathbf{L}_{i+1}$ can be considered as segmenting $\mathbf{L}_i$ into segments and then transforming these segments into $\mathbf{L}_{i+1}$. For example, as shown in Figure 3, building $\mathbf{L}_3 = \{$ `N* n AUX V1 P* N3` $\}$ up to $\mathbf{L}_4 = \{$ `N3 AUX V1 ADTC` $\}$ is equivalent to segmenting $\mathbf{L}_3$ to four segments as $\{$ `[N* n] [AUX] [V1] [P* N3]` $\}$ and then transforming these four segments to $\mathbf{L}_4 = \{$ `N3 AUX V1 ADTC` $\}$. Figure 6 gives an illustration of such segmentation and transformation, where the notation $\mathbf{C}_3$ denotes the segmented phrase-level obtain by segmenting $\mathbf{L}_3$. Therefore, during parsing, a partial tree of i phrase-levels can be represented by a sequence of unsegmented and segmented phrase-levels as $\{\mathbf{L}_1, \mathbf{C}_1, \mathbf{L}_2, \mathbf{C}_2, \cdots, \mathbf{L}_{i-1}, \mathbf{C}_{i-1}, \mathbf{L}_i\}$.

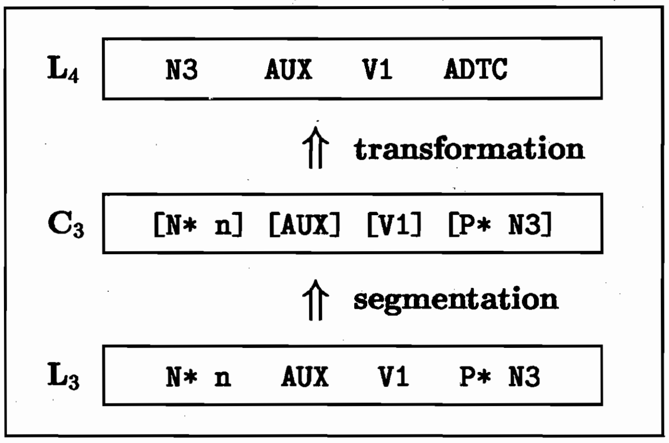

Figure 6: Parsing by segmentation and transformation.

Based on the above representation, the likelihood of a partial tree of i phrase-levels is computed as

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

The approximation in the above equation is based on the assumption that the segmentation and transformation results (i.e. $\mathbf{L}_j$ and $\mathbf{C}_{j-1}$) only depend on the previous phrase-level (i.e. $\mathbf{L}_{j-1}$). According to the above equation, the scoring function $S_{\text{BS}}(\cdot)$ is defined as follows to evaluate the score of a partial tree of i-th phrase-levels, where the subscript BS in $S_{\text{BS}}(\cdot)$ denotes "beam search".

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

where $S_{\text{lex}}(\mathbf{L}_2, \mathbf{C}_1 | \mathbf{L}_1) = P(\mathbf{L}_2, \mathbf{C}_1 | \mathbf{L}_1)$ denotes the lexical score of the part-of-speech sequence of $\mathbf{L}_2$; $S_{\text{syn}}(\mathbf{L}_j, \mathbf{C}_{j-1} | \mathbf{L}_{j-1}) = P(\mathbf{L}_j, \mathbf{C}_{j-1} | \mathbf{L}_{j-1})$ 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 give the details of these two modules.

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.

$$P(\mathbf{L}_2, \mathbf{C}_1 | \mathbf{L}_1) = P(c_1^n | w_1^n) = P(w_1^n | c_1^n) \frac{P(c_1^n)}{P(w_1^n)} \approx \frac{1}{P(w_1^n)} \prod_{j=1}^{n} P(w_j | c_j) P(c_j | c_{j-2}, c_{j-1}), \tag{9}$$

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 the j-th input word. Since the probability $P(w_1^n)$ is a constant, it can be ignored without changing the ranking order of the likelihood probabilities of those competing part-of-speech sequences. Therefore, the scoring function $S_{\text{lex}}(\cdot)$ for the lexical module is defined as

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

where w_j is the j-th input words in $\mathbf{L}_1$ and c_j is the j-th part-of-speech in $\mathbf{L}_2$.

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 follows.

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

Let $A_1, \cdots, A_n$ be the n symbols in $\mathbf{L}_{i-1}$ and $\alpha_1, \cdots, \alpha_m$ be the m segments in $\mathbf{C}_{i-1}$. Then, the first probability term on the right-hand side of Equation (11) is approximated as

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

where $\Gamma_{\text{R2}}(\alpha_{j-2}\alpha_{j-1})$ denotes the rightmost two symbols of $\alpha_{j-2}\alpha_{j-1}$.

The last probability term on the right-hand side of Equation (11) is derived as

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

where $\Gamma_{\text{R1}}(x)$ and $\Gamma_{\text{L1}}(x)$ denote the the rightmost symbol and the leftmost symbol in x respectively.

According to Equations (11)-(13), the syntactic scoring function $S_{\text{syn}}(\mathbf{L}_i, \mathbf{C}_{i-1}|\mathbf{L}_{i-1})$ is defined as follows.

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

where α_j is the j-th segment in $\mathbf{C}_{i-1}$ and A_j is the j-th symbol in $\mathbf{L}_i$. The parameters used in Equation (10) (the lexical scoring function) and Equation (14) (the syntactic scoring function) are smoothed by the Good-Turing smoothing method. Using this scoring function, the syntactic module can rapidly rank the possible candidates for a given phrase-level.

3.4 Simulation Results and Discussions

The PLB parsing mechanism uses the scoring function $S_{\text{BS}}(\cdot)$, Equation (8), to rapidly rank the candidates of phrase-levels and remove the less likely ones during constructing the tree groups. Therefore, only the tree groups with high probability are generated. Then, the scoring function $S_{\text{PT}}(\cdot)$, Equation (6), is used to select the best one from the generated tree groups. The performances of the PLB parsing in the testing set with various beam widths are listed in Table 3. In general, the accuracy rates and the parsing time increases while the beam width increases. The accuracy rates almost saturate after the beam width exceeds 20. On the other hand, the parsing time rapidly increases when the beam width is greater than 20. Therefore, the beam width of 20 is recommended for the PLB parsing in this task.

The results of the baseline system with LLF heuristics (selecting the leftmost longest phrase first) are also listed in Table 3 for comparison. It is obvious that the PLB approach significantly outperforms the baseline system. Even using a very small beam width, the PLB approach achieves better results than the baseline system in terms of the precision and recall of brackets as well as on the accuracy rate of the whole tree group. Since the search space is cut down via a probabilistic scoring function during parsing, the PLB approach can rapidly select the most possible combination of the partial parses for ill-formed inputs. Therefore, the PLB approach with a small search beam width can parse the inputs faster than the baseline system. However, current experiments still cannot claim that the PLB approach is more time-saving than other systems with LLF heuristics, because the LLF heuristics can be implemented by left-corner parsers which are theoretically more efficient than the CYK parser used in the baseline system. But, since the PLB approach can obtain better results than the LLF heuristics within one second, the LLF heuristics is no more attractive even if it could be implemented by a faster parser.

	Beam width	Bracket and its label		Tree group 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 3: The performances of PLB parsing in the testing set with various beam widths.

Table 3 shows that the improvement on bracket precision rate achieved by the PLB approach is better than the improvement on bracket recall rate. For instance, compared to the LLF heuristics, the PLB approach with 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 2.66% (from 78.73% to 81.39%). To further explore the reason, more detailed data are given in Table 4. It indicates that there are 3,343 brackets in the hand-parsed treebank (i.e. the testing set of the 200 ill-formed sentences). The second row shows that there are 3,794 brackets in the parse trees assembled by the LLF heuristics. However, among these 3,794 brackets, only 2,632 brackets (i.e. 69.37%) are correct. Such a low precision rate results from the fact that the heuristics of selecting the longest phrase, either the leftmost one or the global one, usually selects undesirable partial parses. On the contrary, the PLB approach assembles the partial parses according to the statistical information and, consequently, selects the desirable configuration in more cases.

Selecting the longest phrase also causes the baseline system to inadequately partition

	Number of brackets		Precision (%)	Recall (%)
	Total	Matched		
Treebank	3,343	—	—	—
LLF	3,794	2,632	69.37	78.73
PLB	3,423	2,721	79.49	81.39

Table 4: The detailed results of the baseline system and the PLB approach.

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

Table 5: The performances of LLF and PLB on fragments.

the ill-formed sentences. As shown in Table 5, there are 605 fragments in the 200 ill-formed sentences. However, the baseline system partitions these ill-formed sentences into only 355 fragments, which is much smaller than the number of that they should be. This is due to the fact that, while assembling partial parses, the baseline system does not consider the contextual information. It always prefers a larger partial parse than a smaller one, and consequently partitions a sentence into as few fragments as possible. Thus, the baseline system has a very low recall rate for fragments. On the other hand, due to the use of statistical contextual information, the PLB approach can more accurately partition the ill-formed sentences. It partitions the 200 ill-formed sentences into almost the same number of fragments as that they should be. Besides, the number of matched fragments generated by the PLB approach is much larger than that generated by the baseline system. Therefore, the PLB approach has a significantly higher recall rate for fragments than the baseline system (57.9% v.s. 29.6%).

In summary, by using the statistical contextual information, the proposed PLB approach outperforms the baseline system to a great extent. With the beam width of 20, the PLB approach significantly improves the precision of brackets in the tree group from 69.37% to 79.49%. The recall of brackets is also improved from 78.73% to 81.39%.

4 Conclusions

Parsing the ill-formed input usually suffers from the ambiguity problem more deeply than parsing the grammatical sentences. The ambiguities of an ill-formed sentence include all possible tree groups, each of which is a combination of the partial parses jointly generating the input sentence. Since the number of possible tree groups is very large, it is infeasible to do disambiguation by enumerating all of them. In the past, the heuristics of preferring a larger phrase is used (or with other heuristics) 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 presents a Phrase-Level-Building (PLB) parsing mechanism to resolve the ambiguity problem of ill-formed inputs. In this framework, a parse tree is modeled as a set of phrase-levels for being explored in a wider scope. By decomposing a syntactic tree into phrase-levels, this mechanism regards the task of parsing a sentence as a task of building the phrase-levels from the sentence. During parsing, a level-synchronous scoring function is used to remove less likely phrase-levels. As a result, instead of enumerating all possible tree groups, the PLB parser only generates the more likely ones. Whenever all active phrase-levels in the search beam cannot be further reduced by the grammar rules, the process of building phrase-levels is stopped and a probabilistic scoring function is used to select the best tree group. Compared with the baseline system using stochastic context-free grammar and the "leftmost longest phrase first" heuristics, the proposed PLB approach improves the precision rate of brackets in the tree group from 69.37% to 79.49%. The recall rate of brackets is also improved from 78.73% to 81.39%.

Acknowledgement

This paper is a partial result of the project No. 3P11200 conducted by ITRI under sponsorship of the Ministry of Economic Affairs, R.O.C.

References

Briscoe, Ted and John Carroll. 1993. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59.

Chiang, Tung-Hui, Yi-Chung Lin, and Keh-Yih Su. 1995. Robust learning, smoothing, and parameter tying on syntactic ambiguity resolution. *Computational Linguistics*, 21(3).

Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control*, 2:137–167.

Church, Kenneth Ward. 1989. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of ICASSP*, pages 695–698, Glasgow, May 23-26.

Fujisaki, T., F. Jelinek, J. Cocke, E. Black, and T. Nishino. 1989. A probabilistic parsing method for sentence disambiguation. In *Proceedings of the International Workshop on Parsing Technologies*, pages 85–94, Pittsburgh, Pennsylvania, USA, 28–31 Aug.

Good, I. J. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

Hobbs, Jerry R., Douglas E. Appelt, John Bear, and Mabry Tyson. 1992. Robust processing of real-world natural-language texts. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 186–192, Trento, Italy, 31 Mar. – 3 Apr.

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

Jensen, K., G. E. Heidorn L. A. Miller, and Y. Ravin. 1983. Parse fitting and prose fixing: Getting a hold on ill-formedness. *American Journal of Computational Linguistics*, 9(3–4):147–160, July–December.

Katz, S. M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoustic, Speech, Signal Processing*, ASSP-34(3):400–401, March.

Lin, Y.-C., T.-H. Chiang, and K.-Y. Su. 1995. The effects of learning, parameter tying and model refinement for improving probabilistic tagging. *Computer Speech and Language*, 9:37–61.

McDonald, David D. 1992. An efficient chart-based algorithm for partial-parsing of unrestricted texts. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 193–200, Trento, Italy, 31 Mar. – 3 Apr.

Mellish, Chris S. 1989. Some chart-based techniques for parsing ill-formed input. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 102–109, Vancouver, British Columbia, Canada, 26–29 June.

Meteer, Marie and Herbert Gish. 1994. Integrating symbolic and statistical approaches in speech and natural language applications. In *Proceedings of the Workshop on The Balancing Act Combining Symbolic and Statistical Approaches to Language*, pages 69–75, Las Cruces, New Mexico, USA, 1 July.

Ng, See-Kiong and Masaru Tomita. 1991. Probabilistic LR parsing for general context-free grammars. In *Proceedings of the Second International Workshop on Parsing Technologies*, pages 154–163, Cancun, Mexico, 13–15 Feb.

Seneff, Stephanie. 1992. Robust parsing for spoken language system. In *Proceedings of the 1992 International Conference on Acoustics, Speech and Signal Processing*, pages 189–192, San Fransisco, California, USA, 23–26 Mar.

Su, K.-Y., T.-H. Chiang, and Y.-C. Lin. 1990. A unified probabilistic score function for integrating speech and language information in spoken language processing. In *Proceedings of the 1990 International Conference on Acoustics, Speech and Signal Processing*, pages 901–904, Kobe, Japan, Nov. 19-22.

Su, Keh-Yih and Jing-Shin Chang. 1990. Some key issues in designing MT systems. *Machine Translation*, 5(4):265–300.