002

TR-IT-0076

A Statistical Approach to Parsing Ill-Formed Input

Pierre Hudry Yasuharu Den

1994.10

Abstract

In this report we argue for the use of statistical models in the handling of ill-formed input, coupled to a global consideration of syntactic structure. We describe an algorithm based on an already existing language model, Bayesian Language Inference. The algorithm was implemented and tested on artificially altered tagged data taken from the ATR Dialogue Database. Partial conclusions are drawn from these results, focusing on remaining problems and potential future developments.

ATR Interpreting Telecommunications Research Laboratories

Contents

1 Introduction

This report describes an algorithm used for parsing spontaneous speech and attempting to deal in an efficient way with different types of ill-formedness: false starts, filled pauses, substitutions, omissions... The approach proposed to recover from these frequently occuring errors is based on a stochastic language model called Bayesian Language Inference (BLI). After a review of previous work on parsing ill-formed input, a general overview of the BLI algorithm will be given, emphasizing characteristics that make it an appropriate tool capable of partially dealing with ill-formed input. This paper will focus on how partial information yielded by the BLI method can be reorganized in order to parse sentences displaying simple kinds of ill-formedness. This will lead to the description of a statistical processing system capable of analysing ill-formed input with mathematically sound consideration of full syntactic context. Preliminary experiments were conducted on artificially altered tagged corpus, allowing correct detection of errors with an 85% success rate, while full recovery from artificially introduced errors was achieved in 65% of the cases. The partial conclusions that can be drawn from the analysis of these results will be reported, with an emphasis on remaining problems and future search issues.

2 Parsing Ill-Formed Input

2.1 What is "ill-formed input" ?

The collection and analysis of native speaker conversations has allowed linguists and engineers to deepen their understanding of spontaneous speech. Characteristics of natural conversations not found in formal or read speech have been divided into a certain number of different categories [9][10]. From these attempts to list the various structures specifically found in spontaneous speech, it is possible to distinguish the following phenomena:

- structural differences between speech and writing; for instance non-applicability of the traditional notion of sentence to the analysis of natural conversational data, replaced by alternative units of analysis like "information units" [12] or "utterance chunks" [11] (an overview of these differences can be found in [26])
- starts and stops due to the very nature of spontaneous speech; introduction of a certain number of structures that make no significant contribution to the conversation [10]: these structures include filled pauses, false starts, breaks, omissions, repetitions, etc...

Although these phenomena do not usually weaken human understanding of natural language, they do cause significant problems to automated syntactic analysis systems. Moreover, other errors introduced by speech recognizer failure or incorrect part-of-speech tagging are likely to come up and widen the gap between the utterance as intended by the speaker and the actual input given to a parser.

In the following, we will assume that this input is a string of symbols delivered by a part-of-speech tagger, allowing some symbols to be considered "unrecognized", the other symbols being taken from a list of grammatical categories (for the list of Japanese grammatical categories actually used in the experiments, refer to Annex 1). Each string of symbols will be referred to as a *sentence*. The input will be considered *ill-formed* whenever the intended *sentence* (as defined by human analysis) differs from the actual input *sentence* yielded by the part-of-speech tagger. These differences between intended sentence and actual input sentence can be classified into three elementary categories:

- insertion of a symbol
- substitution of a symbol for another symbol
- deletion of a symbol

Insertion phenomena of *unrecognized* symbols are typically encountered in the case of "filled pauses": these are sounds that speakers make to fill silence whenever taking time to consider a structure, lexical item or conversational direction. In Japanese, they can take the form of "etto" or "ano ne", in English usually "um", "ah" or "er". Insertion phenomena of *recognized* symbols occur in what is usually referred to as "false starts": initially uttered material is replaced by a following utterance. In the special case of repetitions the replacement material is strictly identical to the original. But when differing replacement material is actually used to correct the utterance, the ill-formedness introduced is referred to as a "self-repair". Filled pauses and false starts, usually discussed in the literature as disfluencies, may of course cause the insertion of one or more symbols.

Substitutions of a symbol for another symbol typically occur in cases where speech recognition either fails to identify a constituent or just mistakes a constituent for another one. In the first case, the intended symbol will be replaced by some "unrecognized" category; in the second case it will just be unaccurately replaced by another symbol category. Errors coming from the speaker can also be responsible for this type of ill-formedness, in the case of slips of tongue for example. These errors don't occur frequently, and according to estimates found in [5], disfluencies are about 100 times as frequent as slips of tongue.

Finally deletion of a symbol will be encountered whenever a speaker deletes material from an utterance. This type of deletion is typical of the differences between spoken and written language. In the case of spoken Japanese for instance, particles used to identify the grammatical nature of a preceding sub-sequence are frequently omitted. In such situations, the deleted material can in some way be recovered, usually by inference from semantic context.

These three categories of ill-formedness occur very frequently in spontaneous speech. It is reported in [3] that self-repairs are made at an average of once every 5 seconds in dialogues taken from radio talk shows. According to estimates found in [17], about 25% of utterances are ill-formed by any criterion. And although it is generally believed that speakers would make adjustments in speech when talking in a machine translation environment, analysis of conversational data collected in such environment (the ATR environment for Multimodal Interaction, see [16]) by Laurel Fais [10] has shown that ill-formed input would still prove to be very frequent. Therefore no system attempting to deal with spontaneous speech should overlook these phenomena.

2.2 Previous Work on Parsing Ill-formed Input

Before going any further, it is necessary to point out that the framework we have defined so far is purely syntactic. The input sentences given to our parser are a list of grammatical symbols, and only syntactic information is made available. But other works taking into account semantics and world knowledge have been conducted on the subject, using abduction-based inference schemes [13][6][7]. Work reported in [32] also involves integration of prosodic information. But although we are convinced that the ultimate solution to the problem of processing ill-formed input simply has to take into account multiple sources of information, it seems nonetheless important to explore the limits of recovery strategies based on syntax alone.

Such recovery strategies fall into two main categories: pattern-matching techniques and adaptations of chart-based parsing algorithms. The first category includes works found in [28], [35] and [32]. The basic idea behind these techniques is to apply a pattern-matching filter to the input sentence, yielding an initial set of possible repairs (for instance looking for syntactic anomalies, such as "a the" or "to from"). When the parser fails to find a syntactic structure, a repair is selected from the set and the modified sentence is parsed again. Although these simple techniques have proven to be quite successful when seeking to determine only one type of error, they cannot easily be extended to generally deal with several types of errors. This limitation derives from the fact that the detection of possible sources of ill-formedness is made independently from the parsing process and therefore fails to use any structural information (for instance about well-formed subtrees). Moreover, the computational costs introduced by the full re-initialization of the parsing process is another significant drawback of these pattern-matching techniques. But apart from feasibility considerations, the complete lack of structural analysis in itself severely limits the possibilities of such techniques and it seems necessary to consider other strategies.

The second type of recovery strategy considered to deal with ill-formed input is directly adapted from chart-based algorithms. In [29] and [19], a generalized LR parsing algorithm is used, allowing potential symbols to be skipped or inserted anywhere. Although this type of method provides the necessary structural analysis not found in pattern matching techniques, chart-based methods are known to suffer from a left-right bias making it difficult to take into account the right portion of the input sentence. In order to overcome this difficulty, island-driven chart-parsing [33] and combinations of both bottom-up and top-down parsing [25][34] are two ideas which led to further exploration in the field. However, a remaining problem in this approach is the explosion in the number of possible parses. And although this number can be reduced by using beam-search or similar heuristics, the problem of finding the exact parse among hundreds and sometimes thousands of possible ones makes it necessary to rely on a accurate scoring function. In fact even when processing text, in most cases, a parser will produce far more than one analysis. But because of the further widening of the search space one expects to find when handling ill-formed input, it seems likely that hand-crafted scoring functions similar to the ones used in broad-coverage parsers should definitely be left aside.

We have therefore considered alternative solutions to the problem of "finding the best parse". Among these solutions, statistical methods seemed most appropriate, because the use of statistical models in the field of natural language processing has recently led to such dramatic improvements in the performance of parsing systems [24][2]. While allowing automatic training of stochastic grammars, these models also provide the quantitative analysis needed in the disambiguation process. This mathematically sound analysis makes the use of statistical models quite relevant to the task of parsing ill-formed input. However simple local models like n-gram models [14] and probabilistic context-free grammars [15] [18], or even the more complex tree-adjoining grammar formalisms [30] only give us general information about how likely a structure is to appear *anywhere* in a given sentence. Rule expansion at a given node only depends on the portion of input spanned by this node (inside context), and doesn't consider the remaining part of the input (outside context). Therefore were these simple stochastic models to be adapted to handle types of ill-formedness found in spontaneous speech, they would not display the full consideration of context that was already lacking in chart-based methods. In other words they suffer from an inside-outside bias similar to the left-right bias of chart-based methods.

2.3 What we want from a system parsing ill-formed input

From this discussion, it appears that the characteristics to be expected from a parsing system capable of dealing with simple kinds of ill-formedness are:

- an error detection process *combined* to the parsing process, in order to allow the full use of structural information (unlike pattern matching which separately considers a detection phase and a parsing phase)
- a taking into account of the whole input sentence, in order to consider both left and right, inside and outside context (unlike traditional chart-based methods or simple stochastic models)
- use of powerful mathematical tools in the disambiguation process, in order to accurately select the best among a great number of parses (unlike all non-statistical methods which set their parameters by hand)

We will describe in the following a parsing algorithm that possesses exactly these characteristics. Our approach will be based on an already existing language model, called Bayesian Language Inference (BLI): it is one of the more powerful statistical models that have been proposed within recent years. In the next section, we will give a general overview of the BLI algorithm, before moving on to see how it can be adapted to deal with ill-formed input.

3 Bayesian Language Inference

BLI is a stochastic language model developped at *ATR* by Helmut Lucke. It uses the theory of belief propagation in causal trees [27], a general inference model used for combining different sources of information. Since our focus will be on emphasizing characteristics that make it an appropriate tool capable of handling ill-formed input, some originally essential features of BLI will not be mentionned. For those readers who are already familiar with BLI, we refer in particular to:

- the automatic segmentation of the input sequence
- the purely statistical nature of the algorithm
- the continuous training of the parameters

Obviously, a description of the algorithm is best to be found in [20] or [21]. However we *will* give a general overview of BLI, both for the sake of completeness and to introduce notations that will prove useful later on.

The BLI language model is based on a context-free language formalism i.e. a set of terminal and non-terminal symbols and a set of rewriting rules. Terminal symbols (W_m) belong to the same list of categories that make up the input sentence (adverb, noun, etc...) and non-terminal symbols (G_i) correspond to grammatical categories used to identify sentence fragments (noun-phrase, verb-phrase, etc...). In the so-called Chomsky Normal Form [4], the grammar is written so that each rewriting rule describes how a single non-terminal symbol can be rewritten as either a string of two non-terminal symbols ($G_i \longrightarrow G_j G_k$) or as a single terminal symbol ($G_i \longrightarrow W_m$). Figure 1 shows how a typical parse tree can be derived through repeated application of these rules. The squares correspond to the terminal symbols that form the sentence, the circles represent non-terminal symbols, and the binary parse tree shows consecutive applications of the rewriting rules.

In a probabilistic framework each rule has a certain probability of being used. A probabilistic context-free grammar can thus be described as follows:

- a list of N_t terminal symbols $(W_m)_{1 \leq m \leq Nt}$
- a list of N_{nt} non-terminal symbols $(G_i)_{1 \leq i \leq Nnt}$
- a tensor (A_{ijk}) denoting the probability that $G_i \longrightarrow G_j G_k$
- a matrix (B_{im}) denoting the probability that $G_i \longrightarrow W_m$

The rewriting rules probabilities must also satisfy the stochastic constraints:

$$\forall i, \quad \sum_{jk} A_{ijk} + \sum_{m} B_{im} = 1$$

Given these notations we can move on to giving a more detailed explanation of the parsing process. The algorithm is actually divided into three tasks: segmentation, structure and assignment tasks. We will limit our description to the later two, the structure task and assignment task. Assuming then that the observation sequence has already been divided into segments (that we earlier called *sentences*), the BLI algorithm first determines the parse tree

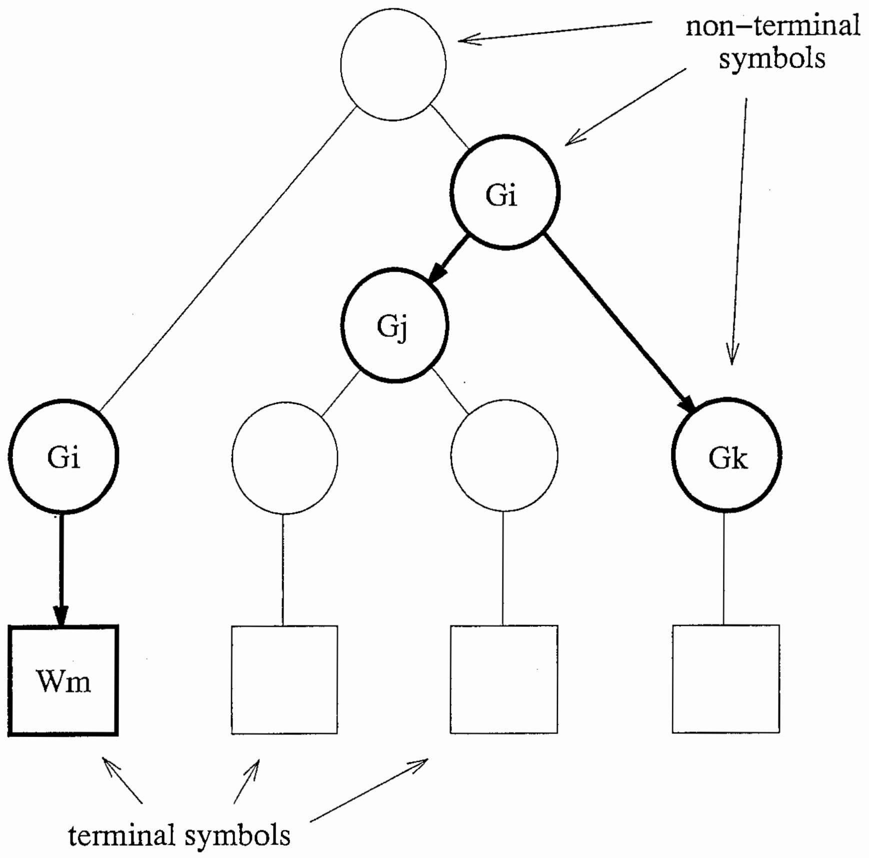


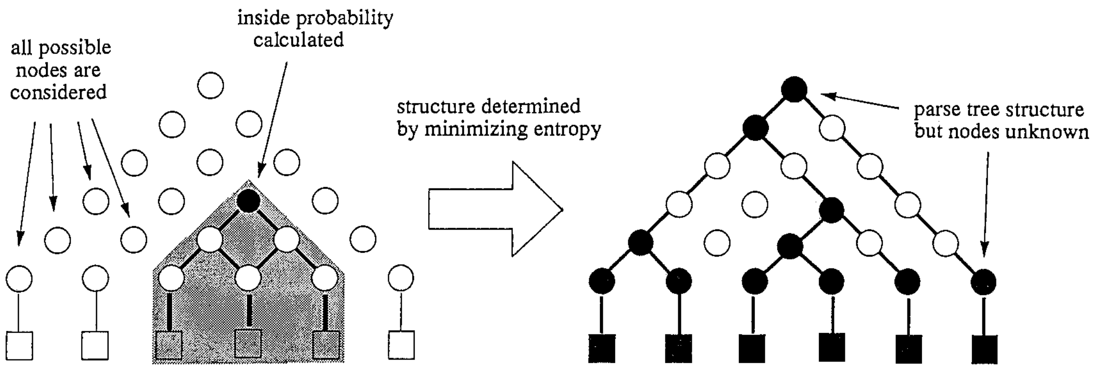
Figure 1: Parse tree

topology for that given segment, but without making assumptions about the nature of each node. In other words, the structure of the parse tree is determined, but without actually deciding upon the identity of the non-terminal symbols at each node. This is done by conjecturing the existence of all possible nodes spanning any sub-sequence of the sentence, as is shown in Figure 2. Recursive calculations of the probability that each node rewrites as the segment it spans are then performed in a fashion similar to the Inside-Outside algorithm [1] (in fact this probability corresponds exactly to Baker's inside probabilities). To put this into mathematical form, for each sub-sequence $W_{to} \ldots W_{tp}$, the BLI algorithm conjectures a node $n(to, tp)$ spanning this segment. Λ is then defined as the probability that node $u = n(to, tp)$ rewrites as the sequence $W_{to} \ldots W_{tp}$:

$$\Lambda(u) = \Lambda(to, tp) = P(e_u^- | u)$$

The following recursive formula is then applied bottom-up to calculate all the Λs:

$$\Lambda(to, tp)_i = \sum_{tq} \sum_{jk} A_{ijk} \Lambda(to, tq)_j \Lambda(tq, tp)_k$$



inside probability calculated
all possible nodes are considered
structure determined by minimizing entropy
parse tree structure but nodes unknown

Figure 2: The structure problem

Using prior parameters obtained through training, division points in the parse tree are then determined, starting from the top node down to the lower nodes. The division point at each node is determined by minimizing the sum of the entropies of its two potential daughter nodes. Then the division process is recursively applied to each of the two daughter nodes until terminal nodes are reached. At this point, the structure of the parse tree is known but the identity of the non-terminal nodes remains unknown.

Once this blank structure has been determined, the BLI method proceeds with the assignment task: assigning appropriate non-terminal symbols to the nodes of the tree. For each node u, the input is divided in two parts (see Figure 3):

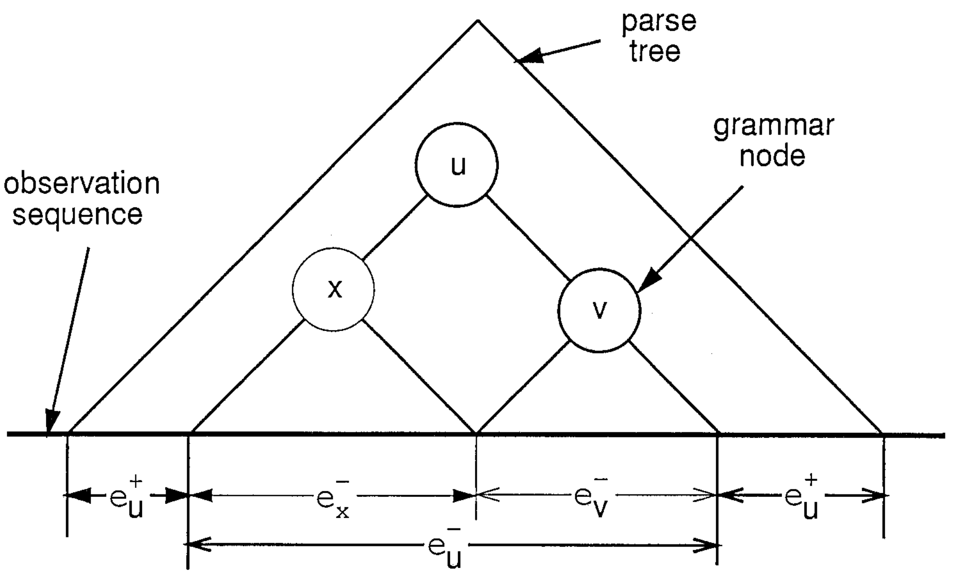


Figure 3: Definition of inner and outer evidence

- inner evidence e_u^-, part of the input actually spanned by node u
- outer evidence e_u^+, remaining part of the evidence

e stands for the entire observation sequence. In this probabilistic framework, the assignment task for each node u is simply that of determining the non-terminal symbol of highest probability given the global evidence e. This is achieved by finding the vector $\mathrm{BEL}(u) = P(u|e)$, the ith component of this vector being defined as the probability that node u corresponds to non-terminal symbol G_i, given the entire input sequence. In order to calculate $\mathrm{BEL}(u)$, two auxiliary vectors are also defined:

$$\lambda(u) = P(e_u^-|u), \quad \pi(u) = P(e_u^+|u)$$

In the framework provided by [27], the parse structure is actually regarded as a Bayesian network, λ and π as the "evidential" and "causal" support of a node. In a more intuitive way, we can say that $\lambda(u)$ provides us with information about the nature of u based on *inner* evidence, whereas $\pi(u)$ provides us with the same type of information, but based on *outer* evidence. The difference between Λs and λs come from the fact that in the later case, the probabilities are calculated given the tree structure. The relations between probabilities of adjacent nodes are provided by the following equations:

$$\begin{aligned} \lambda(u)_i &= \sum_{jk} A_{ijk}\lambda(v)_j\lambda(x)_k && (1)\\ \pi(v)_j &= \alpha\sum_{ik} A_{ijk}\pi(u)_i\lambda(x)_k && (2)\\ \pi(x)_k &= \beta\sum_{ij} A_{ijk}\pi(u)_i\lambda(v)_j && (3) \end{aligned}$$

where α and β are normalization constants. All the lambdas and pis can be determined recursively using only local calculations and the belief vector is then given by:

$$\mathrm{BEL}(u) = \frac{\lambda(u)\pi(u)}{\lambda(u)\cdot\pi(u)}$$

where ab is componentwise vector product, and $a \cdot b$ is the dot product.

These equations can be understood in the following way (see Figure 3): $\lambda(u)$ is determined using only lambda values of daughter nodes v and x. In other words, the inside evidence e_u^- is divided into e_v^- and e_x^- and λs can be calculated bottom-up. On the other hand, determining $\pi(v)$ is done using π values of mother node u and λ values of sister node x, the outside evidence e_v^+ being divided into e_u^+ and e_x^-. Once all λs have been determined, the π vectors can be determined from top to bottom nodes. The final equation yielding $\mathrm{BEL}(u)$ only means that the probability that node u stands for a given non-terminal symbol is obtained through the combination of two sources of information: inner evidence given by $\lambda(u)$ and outer evidence given by $\pi(u)$. Full syntactic context, divided into inner and outer evidence, is therefore considered.

Although pursuing our description of the BLI algorithm beyond this point would mean going into details that can already be found in the original papers, some additional features should at least be briefly mentionned for the flexibility they allow and their potential usefulness in future concerns. The first is the possibility to work on unsegmented data, allowing the continuous processing of an infinite symbol source. The second feature concerns a possible interface with a speech recognizer which instead of producing symbols would produce lattices of symbols to be directly interpreted as BLI lambda vectors. The third and last feature definitely worth mentionning is of course the training of parameters in BLI, which is one issue we have left aside.

4 Adapting BLI to Parse Ill-formed Input

The main idea behind this work is that whereas the BLI method only uses λ and π vectors as auxiliary means to calculate BEL, the information these vectors contain is particularly useful as it is in parsing ill-formed input. We will now describe how recovery from the three different types of ill-formedness can be achieved by combining the information provided by inner and outer evidence. But before that, we need to introduce some additional notations: let's call S the set of possible input sentences, i.e. the set of finite strings formed of elements from (W_m). We also consider W_f the subset of sentences for which the BLI method as we have described it succeeds in yielding a parse; I_f is further defined as $S \backslash W_f$, subset of sentences for which it is unsuccessful. Let $s_o \in W_f$ be the sentence originally intended by the speaker and s the actual input sentence obtained after whatever deletions, insertions and substitutions occur. s is generally expected to belong to I_f but, in some cases, the modified utterance will be parsed successfully as a well-formed sentence, with a meaning not intended by the speaker. In [28], this is called "change to well-formed" and is found to occur in over 10% of the test sentences. How to cope with such input sentences remains a problem, but we will chiefly be concerned here with utterances such that $s_o \in W_f$ and $s \in I_f$. For such an utterance, parsing will be divided in structure task and assignment task just like what is done in BLI.

4.1 The structure problem

Suppose then that the BLI method *doesn't* succeed in finding a parse, the first general problem to adress is determining the tree structure best associated to input sentence s. Rather than using the BLI algorithm and in case of failure applying some special recovery algorithm giving us the most probable tree structure, we have tried to adapt the original structure task algorithm so that it could deal with both well and ill-formed sentences. There are two things we expect from such an adapted version of the structure task:

- first of all, it should be capable of yielding results identical to the ones obtained by using the BLI algorithm in the case where s belongs to W_f
- second, in the case where s belongs to I_f, it should choose a tree structure taking into account the existence of well-formed subtrees and their different probabilities

To achieve this, we introduced some noise in the probabilities attached to the different rewriting rules. These modified rules are only to be used in the structure task, and *not* in the assignment task. The original probabilistic context-free grammar was altered according to the different types of ill-formedness handled: for instance, to deal with the case of insertion, rules like $G_i \longrightarrow G_j\star$ or $G_i \longrightarrow \star G_k$ were introduced with extremely low probabilities (where $\star$ can be any non-terminal symbol). Thus if the input is well-formed, results yielded by the calculations do not differ from those obtained with the BLI method. And in the case of ill-formed input, the tree structure chosen will be one using the greatest number of well-formed subtrees. Our version of the structure task therefore fulfills the requirements we described earlier.

It is important to note that, when ill-formedness occurs, several parse structures can possibly lead to identification and correction of errors, and thus be said to be correct. For instance, still in the case of insertion, the ill-formed segment can be either attached left or right at any point in the well-formed parse tree. Having chosen a tree structure, the branch responsible for ill-formedness will then be detected and discarded in the second phase, the assignment task. This makes the structure problem a much simpler one to solve, several solutions being allowed. It is therefore not necessary for the time being to try and go beyond the simple method we have described here since a more refined analysis will be provided in the assignment task. Let's examine how, given the blank tree structure yielded in this first phase, it is possible to proceed with an adapted version of the assignment task.

4.2 The assignment problem

Once we have the topology of the parse tree spanning the observation sequence, the problem of assigning grammatical categories to each node remains to be dealt with. Using the *unaltered* rewriting rules probabilities, we proceed in the same way as the BLI method: all λ vectors (probabilities resting on inside evidence) are recursively calculated bottom-up, leading to the top node spanning the entire input sentence. The π vectors (probabilities resting on outside evidence) can then be calculated. In the case of well-formed input (parsing s_o), no problem will be encountered and, after all πs and λs have been calculated, the algorithm will determine BEL vectors and assign to each non-terminal node its most appropriate grammatical category G_i. However, in the case of ill-formed input (parsing $s \in I_f$), the calculations of πs and λs will lead to zero probability vectors at some nodes, causing BEL $= 0$. This prevents us from assigning categories to most non-terminal nodes. However, the partial information made available by non-zero probability vectors can be analysed at a finer-grain level, allowing recovery from ill-formedness. We proceed in three steps:

- detection of the region where the ill-formedness is present
- identification of the type of ill-formedness encountered
- recovery from the ill-formedness identified in previous step

To illustrate our method, we will consider the case of a single error (although multiple errors can in theory be handled) and describe these three operations in detail, starting with the detection of ill-formed region.

The region where the ill-formedness lies can be determined in a very straightforward way. Our criteria for identifying this region is to find a mother node u with daughter nodes v and x such that (see Figure 3):

$$\lambda(u) = 0; \quad \lambda(v) \neq 0; \quad \lambda(x) \neq 0$$

The ancestors of u, all the way to the top node, will therefore have zero λ probabilities. However, given the recursive formulas used to calculate the probabilities from outer evidence, the π vectors of those ancestor nodes can be accurately determined, including $\pi(u)$. Identification of ill-formedness encountered in the specified region will then be made possible by directly comparing λ and π vectors for this group of three nodes, and analysing coherence between their different values.

For instance, let's study the case of insertion (Figure 4) of a symbol or group of symbols

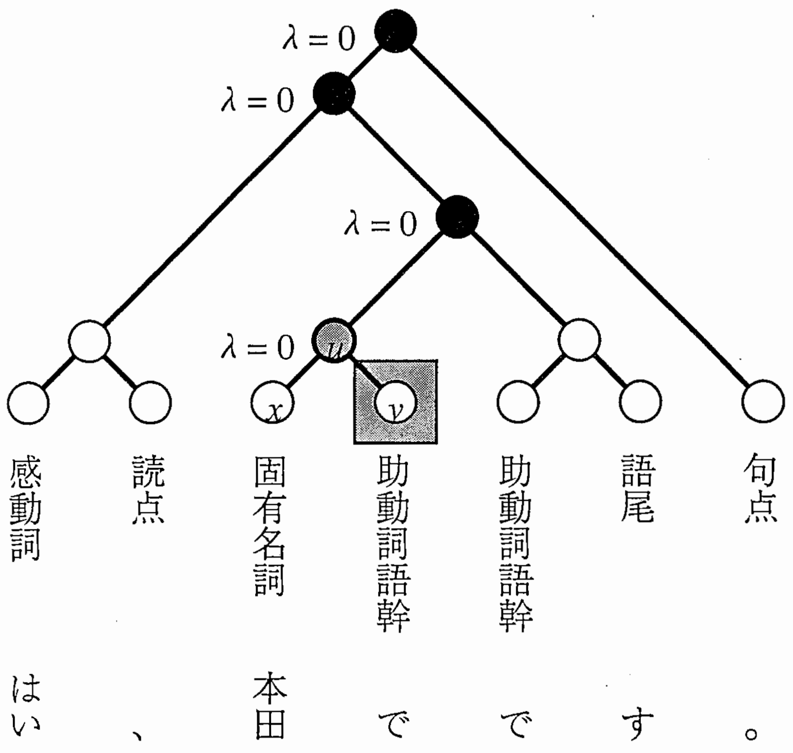


Figure 4: Insertion occuring at node v

spanned by node v. Because of the ill-formedness, the inner portion of the evidence, including the inserted segment, will cause $\lambda(u) = 0$, failing to bring any information. But the parser *will* give an accurate analysis of the outer portion of the evidence. Since, in the case of insertion at node v, nodes u and x are actually one same node, the insertion can be detected by directly comparing $\lambda(x)$ and $\pi(u)$. If those two vectors are "sufficiently close" to one another, we can hypothesize an insertion at node v. The portion of the input sentence responsible for the ill-formedness can then be eliminated and the parser proceed with calculations of λ and π vectors. Although several definitions of "close" can be thought of, we selected the following criteria for identifying insertion at node v:

$$\lambda(x) \cdot \pi(u) > \theta_i$$

where θ_i is a threshold value.

In the case of substitution of a symbol for another symbol (Figure 5), occuring for instance

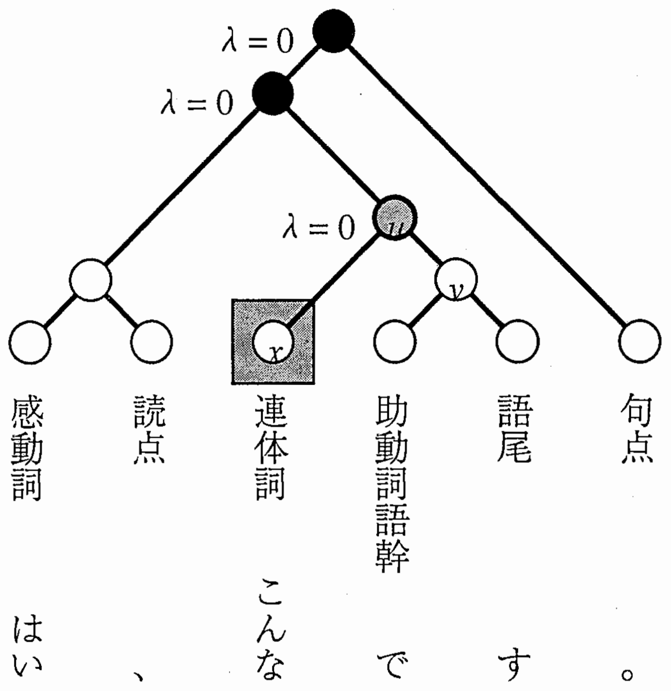


Figure 5: Substitution at node v

at node v, the ill-formedness will cause $\lambda(v)$ to bring false information about inner evidence at node v. Therefore, informations provided by inner evidence at node v ($\lambda(v)$) and outside evidence at node u ($\pi(u)$) are likely to contradict each other. Therefore $\pi(x)$, calculated from $\lambda(v)$ and $\pi(u)$, is likely to give us no information at all. We therefore selected the following criteria for identifying substitution at node v:

$$\pi(x) \cdot \pi(x) < \theta_s$$

When the substitution is detected at node v, inner evidence being useless, the node will be assigned the symbol G_i such that $\pi(v)_i$ has the greatest value. This only means that we fully rest on outer evidence in the assignment task, putting aside any information provided by unreliable inner evidence.

Finally in the case of deletion of a symbol occuring for instance between node u and node v, an extra node w should be considered between u and v, and the outside evidence at this node gives:

$$\pi(w) = \pi(v)$$

Calculating $\pi(y)$ from $\pi(w)$ and $\lambda(v)$, a deletion will actually occur when the outside evidence at hypothesized node y is very strong. Thus our criteria for identifying substitution between node x and v will be:

$$\pi(y) \cdot \pi(y) > \theta_d$$

If a deletion is detected, the structure will be modified accordingly by introducing a new branch in the tree and recalculating λs and πs in appropriate regions.

On a more general level, we can also point out that the computational cost of the operations we have described is not any higher than the one involved in parsing well-formed input. This can be explained by the fact that an ill-formed section of input will generate a great number of zero λ and π probability vectors, which all lead to trivial calculations. This apparent reduction in cost will of course be compensated later on in the recalculation of probability vectors from the new non-zero λs and πs. But the important point here is that dealing with ill-formed input using this method, unlike the other approaches we earlier mentionned, does not induce tremendous additional computational costs.

5 Experimental Evaluation

We carried out preliminary experiments to examine the validity of the above approach. We will now describe these experiments and report results of interest.

5.1 Description of the Experiment

In our experiment, a set of 245 sentences were taken from a Japanese corpus the ATR Dialogue Database (ADD) [8]. These sentences ranged in length from 2 to 8 symbols. They were artificially altered by randomly inserting or substituting one symbol in the original well-formed input sentence. The lists of terminal symbols (50) and non-terminal symbols (90) can be found in the annex and were directly derived from the corpus. Since training wasn't the focus of this work, we chose to take as probabilities the direct frequency counts estimated from the corpus. Although these probabilities are directly estimated from the data, they are *not* the optimal parameters as far as the BLI method is concerned. They do however provide us with likely values for the rewriting rules probabilities. We asked the algorithm to react to ill-formed input by detecting the symbol responsible for the modification and identifying the type of modification involved.

For each of the sentences, we randomly inserted the three following categories: kandōshi, gomi and kakujōshi. The two later were chosen because they are strongly constrained grammatical symbols, and therefore should be easily detected as source of error when leading to ill-formed input. This gave us a set of 735 sentences. The parser had to detect whether a substitution or an insertion had occured, and recover from the ill-formedness. Out of these 735 sentences, 24% were parsed sucessfully by the BLI method. This is much higher than the 10% of "change to well-formed" reported in [28], but since our errors were introduced randomly this isn't too surprising. Of the remaining 555 sentences, 527 were given a structure similar to the original structure, the inserted portion being added to that structure in a "recoverable" way. For 330 of these 527 sentences, the exact parse tree was recovered, identical to the one given to the unaltered input sentence. And out of the total 555 sentences, whether identified as substitution or insertion, the inserted symbol was detected as the source of ill-formedness for 487 sentences.

For each of the sentences, we then randomly chose a symbol and changed it to either kandooshi, gomi and kakujooshi. This gave us another set of 735 sentences. The parser again had to detect whether a substitution or an insertion had occured, and recover from the ill-formedness. Out of these 735 sentences, 20% were parsed sucessfully by the BLI method. Of the remaining 587 sentences, 468 were given a structure identical to the original structure. For 403 of these 468 sentences, the exact parse tree was recovered, identical to the one given to the unaltered input sentence and the original symbol was found to belong to the lattice of symbols yielded by analysis of outer evidence. And out of the total 555 sentences, whether identified as substitution or insertion, the substituted symbol was detected as the source of ill-formedness for 483 sentences. These results for insertion and substitution can be found in Figure 6.

	Detected and Recovered	**Detected**
Insertion	**59 %**	**88 %**
Substitution	**69 %**	**82 %**
Total	**64 %**	**85 %**

Figure 6: Experimental results

5.2 Discussion

As far as finding an appropriate structure, our method does not suffer from its extreme simplicity: a recoverable tree structure is found in more than 87% of the cases. We do have hope however of refining these techniques to increase this percentage. Nonetheless, since even when a different tree structure is found it is always possible to identify the exact source of ill-formedness, we think the center of focus should be the second task, the assignment problem. Once the correct structure is found, the exact parse tree can be recovered in 63%

of the cases in the case of insertion and 86% of the cases for substitution (for a global success rate of 74%). In the remaining 26% of the cases, the parser identifies what was a substitution for an insertion or vice-versa. Since we only used very crude methods to determine whether insertion or substitution was occuring, we think one of the priorities in the future should be refining the θ parameters used in this choice, especially when conducting experiments involving the third type of ill-formedness we have isolated, deletions. However, a perfect success rate seems unconceivable since insertion of a symbol somewhere could very well lead to a very likely substitution, just like it can lead to well-formed input for instance.

Other additional remarks can also be made about the difference of results between insertion and substitution. It is not surprising that the structure task get lesser results in the case of identifying substitutions (80% success rate) because there is only one possible structure from which it is possible to recover, the exact structure obtained in the case of well-formed input. However, as we mentionned earlier, in the case of insertion (95% success rate) the ill-formed segment can be either attached left or right at any point in the well-formed parse tree. The flexibility allowed in this task is enough to account for this difference. Not surprisingly, those results are inverted in the assignment task. When the exact structure is found, the parser succeeds in identifying a substitution and submitting the original symbols among the list of possible options in 86% of the cases, whereas the success rate for input suffering from insertion drops to 63% in that second task.

Since our experiments were conducted on small sentences with at most 8 symbols, we studied the recovery and detection rates as a function of the number of symbols (Figure 7). These results show that an increase in the number of symbols doesn't seem to reduce the efficiency of our method and therefore the length of the sentences parsed hasn't proved to be a limitation here.

6 Conclusion and Future Directions

The preliminary experiments were led on small sentences, and artificially altered data. Besides only one error was introduced at one time and we left out the cases of deletions and unrecognized symbols. Thus we certainly can't claim at this point having proved anything. But these experiments did give a certain concreteness to the ideas and point to a number of unresolved problems:

- training the parameters involved in selecting type of ill-formedness involved
- taking advantage of other useful features of the BLI model (segmentation...)
- dealing with multiple errors, leading to experiments with real data

The training of the parameters is an issue we have left aside but which should be considered at some point: how should we estimate the rewriting rules probabilities ? and the parameters involved in the alteration of these rules in the structure task ? how about the θ parameters used in selecting one type of ill-formedness over another ? All the experiments were conducted with very rough estimates: using the *exact* probabilities yielded by the BLI

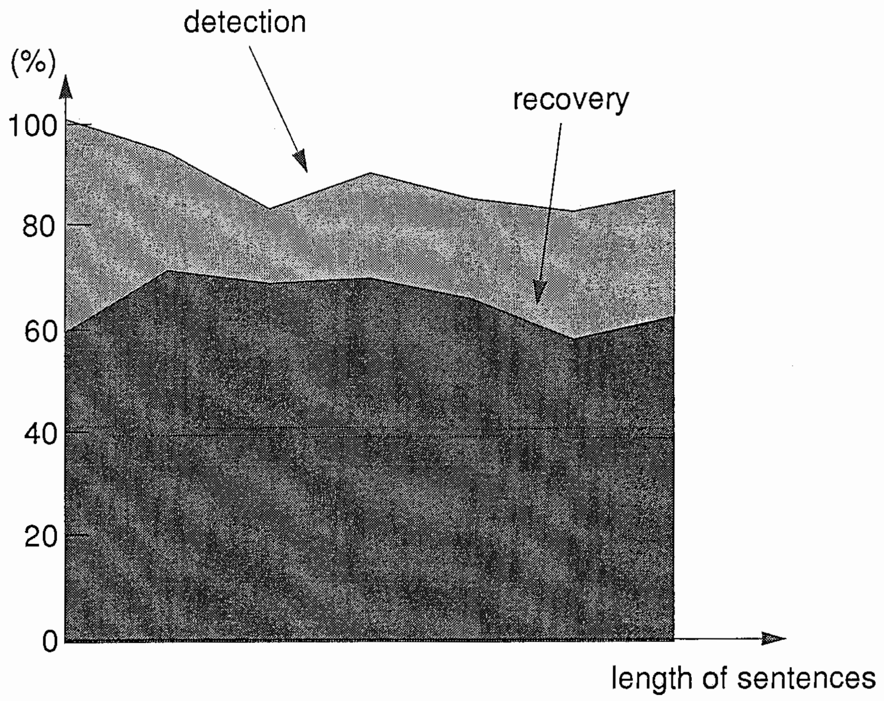


Figure 7: Recovery and detection rates depending on length of sentences

method, we tried to combine these probabilities in an intuitive way to detect any incoherence caused by ill-formed input. But as we have seen, no fine tuning of the parameters was made. However, although these parameters were rough estimates, preliminary experiments have shown rather promising results. Therefore it seems that the efficiency of this method could be greatly improved by including some kind of training.

Moreover, other features of BLI give this method additional potential: we have already mentionned the possibility to process unsegmented text but, equally important, is the possibility to interface BLI with a speech recognizer. In this discussion we have considered that the input sequence was a string of grammatical symbols taken from a list of categories, including an unknown category. But this is only a special case where the probability vectors at *terminal* nodes are of the form $\lambda = (0 \ldots 0, 1, 0 \ldots 0)$, the non-zero value corresponding to the terminal symbol identified (unrecognized symbols being such that $\lambda = 0$). But in a general case where a part-of-speech tagger would give us as input lattices of symbols, the previous method could also work very well. And hopefully resolve at the syntactic levels ambiguities present at the recognition level.

About dealing with multiple errors, we considered using some prior π probabilities trained from the data and depending upon the number of symbols spanned. The problem in the case of multiple errors is that with the method we described, πs can't always be calculated from the top node down to the portion of ill-formed input. Whenever two daughter nodes have zero λ probabilities, this is not possible anymore. Therefore the use of prior π probabilities has estimates used in such cases seemed most appropriate. Such priors are already used in the structure task in BLI, which doesn't necessarily consider the top node to be actually spanning a sentence. The most simple type of prior should depend at least upon the number of symbols it covers, but relative position in the tree, nature of spanned symbols and so on could also be considered.

This taking into account of multiple errors should lead to experiments with real data, which is the only way to really test any approach in natural language processing. In this work, we randomly inserted or substituted symbols in the input sentence but it is obvious that ill-formedness in spontaneous speech doesn't occur just *anywhere*. Recent linguistic findings tend to show that on the contrary, such errors are highly systematic [5]. Some works have in fact tried using features specific to spontaneous speech as an advantage in machine translation. In [31], work was conducted on how filled pauses and pauses could actually be used for segmenting Japanese utterances. Similarly we think the method we used might prove to be more efficient with real ill-formed data than with artificially altered data.

Conclusion

We have argued in the above for the use of statistical methods coupled with a global consideration of context in the parsing of ill-formed input. Such a method was proposed, adapted from an already existing language model which posessed these two characteristics. The description of this method was given and experiments were conducted: the results of these experiments, although quite promising, are not sufficient yet to draw any definitive conclusion on the validity of the above approach. However, at this point, the method we considered offers a wide range of future developments, capable of testing the method's potential and its limitations.

References

[1] Baker, J.K., Trainable Grammars for Speech Recognition, Speech Communication Papers for the 97th Meeting of the Acoustical Society of America, pp. 547–550, 1979.

[2] Black, E., Parsing English by Computer: The State of the Art, Proceedings of the ATR International Workshop on Speech Translation, 1993.

[3] Blackmer, E.R., and Mitton, J.L., Theories of Monitoring and the Timing of Repairs in Spontaneous Speech, *Cognition* 39, pp. 173–194, 1991.

[4] Chomsky, N., On Certain Formal Properties of Grammars, *Information and Control*, No.2, pp. 137–167, 1959.

[5] Clark, H.H., Wasow, T., Repeated Words in Spontaneous Speech - A Preliminary Report. Center for the Study of Language and Information, Stanford University, 1994.

[6] Den, Y., A Study on a Spoken Dialogue Grammar, *Proceedings of the SIG-SLUD Workshop sponsored by the Japanese Society for Artificial Intelligence*, pp. 33–40, 1993.

[7] Den, Y., Generalized Chart Algorithm: An Efficient Procedure for Cost-Based Abduction, *Proceedings of 32nd Annual Meeting of the ACL*, pp. 218–225, 1994.

[8] Ehara, E., et al. Contents of the ATR Dialogue Database, ATR Technical Report TR-I-0186. Kyoto, Japan: ATR Interpreting Telephony Laboratories, 1990.

[9] Fais, L., Non-grammatical Phenomena in Real English Conversation, ATR Technical Report TR-IT-0007. Kyoto, Japan: ATR Interpreting Telecommunications Laboratories, 1993.

[10] Fais, L., Structures in Spontaneous English Conversation, ATR Technical Report TR-IT-0040. Kyoto, Japan: ATR Interpreting Telecommunications Laboratories, 1994.

[11] Gumperz, J.J., Sociocultural Knowledge in Conversational Inference, in M. Saville-Troike, (ed.), *Georgetown University Round Table* (Washington D.C.), pp. 191–211, 1977.

[12] Halliday, M.A.K., *Explorations in the Functions of Language* (London), 1973.

[13] Hobbs, J.R., Stickel, M.E., Appelt, D.E., and Martin, P., Interpretation as Abduction, *Artificial Intelligence* 63, pp. 69–142, 1993.

[14] Jelinek, F., Up from trigrams!, Proceedings Eurospeech, pp. 1037–1040, 1991.

[15] Jelinek, F., Lafferty, J.D., and Mercer, R.L., Basic Methods of Probabilistic Context Free Grammars, Continuous Speech Recognition Group IBM T.J. Watson Research Center, 1991.

[16] Loken-Kim, K., Fumihiro, Y., Kazuhiko, K., Fais, L., and Ryo, F., EMMI-ATR environment for multi-modal interactions, ATR Technical Report TR-IT-0018. Kyoto, Japan: ATR Interpreting Telecommunications Laboratories, 1993.

[17] Labov, W., *Sociolinguistic patterns*, Oxford: Basil Blackwell, 1972.

[18] Lari, K., Young, S.J., Applications of Stochastic Context-free Grammars Using the Inside-outside Algorithm, *Computer Speech and Language*, No.5, pp. 237–257, 1991.

[19] Lavie, A., Tomita, M., GLR* An Efficient Noise-skipping Parsing Algorithm for Context Free Grammars, *IWPT'93*, pp. 123–133, 1993.

[20] Lucke, H., A Method for Inferring Stochastic Context-free Grammars Using the Theory of Bayesian Causal Trees, Proccedings of the Institute of Electronics, Information and Communication Engineers, Technical Report of ASJ Speech Commitee, SP92–113, pp. 79–86, 1992.

[21] Lucke, H., Bayesian Belief Networks as a Tool for Stochastic Parsing, submitted to *Speech Communication*, 1993.

[22] Lucke, H., Inference of Stochastic Context-Free Grammar Rules from Example Data using the Theory of Bayesian Belief Propagation, *Proceedings of Eurospeech '93*, pp. 1195–1198, 1993.

[23] Lucke, H., Reducing the Computational Complexity for Inferring Stochastic Context-free Grammar Rules from Example Text, Proceedings ICASSP'94, pp. 353–356, 1994.

[24] Marcus, M., Statistical Natural Language Processing: Current Trends and Future Directions, Proceedings of the ATR International Workshop on Speech Translation, 1993.

[25] Mellish, C., Some Chart-Based Techniques for Parsing Ill-Formed Input, *Proceedings of 27th Annual Meeting of the ACL*, pp. 102–109, 1989.

[26] Niyi Akinnaso, F., On the Differences Between Spoken and Written Language, *Language and Speech*, Vol. 25, Part 2, pp. 97–125, 1982.

[27] Pearl, J., *Probabilistic Reasoning in Intelligent Systems - Networks of Plausible Inference*, Morgan and Kaufmann, 1987.

[28] Sagawa, Y., Ohnishi, N., Sugie, N., A Parser Coping with Self-Repaired Japanese Utterances and Large Corpus-Based Evaluation, pp. 593–597, 1993.

[29] Saito, H., Tomita, M., Parsing Noisy Sentences, Proceedings of the 12th Intrnational Conference on Computational Linguistics COLING'88, pp. 561–565, 1988.

[30] Schabes, Y., Stochastic Lexicalized Tree-adjoining Grammars, Proceedings of the 16th Intrnational Conference on Computational Linguistics COLING'92, pp. 426–432, 1992.

[31] Seligman, M., Boitet, C., A Whiteboard Architecture for Automatic Speech Translation. *Proceedings of the International Symposium on Spoken Dialogue '93*, pp. 243-246, 1993.

[32] Shriberg, E., Bear, J., Dowding, J., Automatic Detection and Correction of Repairs in Human-Computer Dialog, Proceedings of the DARPA Speech and Natural Language Workshop, 1992.

[33] Stock, O., R. Falcone, and P. Insinnamo, Bidirectional Charts: A Potential Technique for Parsing Spoken Natural Language Sentences, *Computer Speech and Language*, No. 3, pp. 219–237, 1989.

[34] Theeramunkong, T., and Tanaka, H., A Parallel Chart-based Parser for Analysing Ill-formed Input, *SIG-PPAI Japanese Society for Artificial Intelligence Technical Report*, 1993.

[35] Woszczyna, M., et al. JANUS 93: Towards Spontaneous Speech Translation, to appear Proceedings ICASSP-94, 1994.

A List of Terminal Symbols

感動詞	数詞
読点	連体助詞
固有名詞	人名
助動詞語幹	副助詞
語尾	補助動詞
終助詞	形容詞
疑問符	受身助動詞語幹
副詞	副詞的名詞
句点	引用助詞
サ変名詞	形容名詞
格助詞	日時
本動詞	文副詞
準体助詞	連体詞
複合語	通貨記号
係助詞	記号
接頭辞	丸括弧
助動詞	引用符
接続詞	パーセント記号
代名詞	感嘆符
補助動詞語幹	アンダーバー
接続助詞	中黒
普通名詞	等号
並立助詞	使役助動詞語幹
住所名	ピリオド
接尾辞	かぎ括弧

B List of Non-Terminal Symbols

symbol_G251	アンダーバー
symbol_G252	サ変名詞
symbol_G253	テ形補助動詞
symbol_G254	テ形補助動詞語幹
symbol_G255	パーセント記号
symbol_G256	ピリオド
symbol_G257	引用助詞
symbol_G258	引用符
symbol_G259	格助詞
symbol_G260	感嘆符
symbol_G261	感動詞
symbol_G262	丸括弧
かぎ括弧	記号

疑問符
句点
区画番地
係助詞
形容詞
形容名詞
固有名詞
後置詞句
語尾
使役助動詞語幹
受身助動詞語幹
終助詞
住所
住所名
住所要素
準体助詞
助動詞
助動詞語幹
人名
数詞
数詞連体詞句
数量詞
姓名
接続詞
接続助詞
接頭辞
接尾辞
節
態の助動詞
態の動詞
態の動詞句
代名詞
中黒
通貨記号
等号
動詞
動詞句
読点
日時
番地連体詞句
普通名詞
副詞
副詞句
副詞節
副詞的名詞
副助詞
複合区画番地
複合語
複合数詞
複合日時
複合番地要素
文
文副詞
並立助詞
補助動詞
補助動詞語幹
本動詞
名詞句
名詞節
連体詞
連体詞句
連体修飾節
連体助詞
連用修飾