Lexicalized Tree Automata-based Grammars for Translating Conversational Texts

Kiyoshi YAMABANA Shinichi ANDO Kiyomi MIMURA

Computer & Communication Media Research, NEC Corporation

4-1-1, Miyazaki, Miyamae-ku, Kawasaki 216-8555, JAPAN

k-yamabana@ct.jp.nec.com

s-ando@cw.jp.nec.com

k-mimura@da.jp.nec.com

Abstract

We propose a new lexicalized grammar formalism called Lexicalized Tree Automata-based Grammar, which lexicalizes tree acceptors instead of trees themselves. We discuss the properties of the grammar and present a chart parsing algorithm. We have implemented a translation module for conversational texts using this formalism, and applied it to an experimental automatic interpretation system (speech translation system).

1 Introduction

Achieving both broad coverage for general texts and better quality for texts from a restricted domain has been an important issue in practical natural language processing. Conversational language is a typical domain this problem has been notable, since they often include idioms, colloquial expressions and/or extra-grammatical expressions while a majority of utterances still obey a standard grammar.

Furuse and Iida (1994) proposed an approach to spoken-language translation based on pattern matching on the surface form, combined with an example-based disambiguation method. Since the grammar rules are simple patterns containing surface expressions or constituent boundaries, they are easy to write, and domain-specific knowledge can be easily accumulated in the grammar. On the other hand, relationships between two trees are not easy to describe, especially when they are separated apart on a larger tree. This might become an obstacle in expanding a domain-specific grammar into a general grammar with a wide coverage.

Brown (1996) approached to this problem employing a multi-engine architecture, where outputs from Transfer Machine Translation (MT), Knowledge-based MT and Example-based MT are combined on the chart during parsing. Ruland et al.

(1998) employs a multi-parser multi-strategy architecture for robust parsing of the spoken language, where the results from different engines are combined on the chart using probability-based scores. A difficult part with these hybrid architectures is that it is not easy to properly compare and combine the results from different engines designed on different principles. In addition, these methods will require much computational power, since multiple parsers have to be run simultaneously.

A third approach, such as Takeda (1996), is grammar-based. In this approach, a method is provided to associate a grammar rule to a word or a set of words in order to encode their idiosyncratic syntactic behaviour. An associated grammar rule can be seen as a kind of example if it is described mostly by the surface level information. As is apparent from this description, this approach is an application of strong lexicalization of a grammar (Schabes, Abeillé and Joshi, 1988).

This approach allows coexistence of general rules and surface-level patterns in a uniform framework. Combination of both types of rules is naturally defined. These advantages are a good reason to employ strongly lexicalized grammars as the basic grammar formalism. However, we feel there are some points to be improved in the current strongly lexicalized grammar formalisms.

The first point is the existence of globally defined special tree operation, which requires a special parsing algorithm. In a strongly lexicalized grammar formalism, each word is associated with a finite set of trees anchored by that word. The tree operations usually include substitution of a leaf node by another tree, corresponding to expansion of a nonterminal symbol by a rewriting rule in CFG. However, if the tree operation is limited to substitution, the resulting grammar, namely Lexicalized Tree Substitution Grammar (LTSG), cannot even reproduce the trees obtained from non-lexicalized context free grammars. This will be obvious from the fact that for any LTSG, there is a

constant such that, in any trees built by the grammar, the distance of the root node and the nearest lexical item is less than that constant, while this property does not always hold for CFG. Tree Insertion Grammar (TIG), introduced by Schabes et al. (1995), had to be equipped with the insertion operation in addition to substitution, so that it can be strongly equivalent to an arbitrary CFG. The insertion operation is a restricted form of the adjoining operation in the Lexicalized Tree Adjoining Grammar (LTAG) (Joshi and Schabes, 1992).

Thus, a special tree operation other than substitution is inevitable to strongly lexicalized grammars. It is needed to grow an infinite number of trees from a finitely ambiguous set of initial trees representing the extended domain of locality (EDOL) of the word.

However, such special tree operation requires a specially devised parsing algorithm. In addition, the algorithm will be operation-specific and we have to devise a new algorithm if we want to add or modify the operation at all. Our first motivation was to eliminate the need for globally defined special tree operations other than substitution whenever possible, without losing the existence of EDOL.

Another point is the fact that lexicalization is applied only to trees, not to the tree operations. For example, in LTAG, initial tree sets anchored to a word is not enough to describe the whole set of trees anchored by that word, since initial trees are grown by adjunction of auxiliary trees. Since an auxiliary tree is in the EDOL of another word, the former word has limited direct control over which auxiliary tree can be adjoined to certain node. For detailed control, the grammar writer has to give additional adjoining restrictions to the node, and/or detailed attribute-values to the nodes that can control adjunction through node operations such as unification.

In short, we would like to define a lexicalized grammar such that 1) tree operation is substitution only, 2) it has extended domain of locality, and 3) tree operations as well as trees are lexicalized whenever possible. In the next section, we propose a grammar formalism that has these properties.

2 Lexicalized Tree Automata-based Grammars

In this section we introduce Lexicalized Tree Automata-based Grammar (LTA-based Grammar) and present its parsing algorithm.

First, we define some basic terminologies. A grammar is *strongly lexicalized* if it consists of 1) a finite set of structures each associated with a lexical item; each lexical item will be called the anchor of the corresponding structure, and 2) an operation or operations for composing the structures (Schabes, Abeillé and Joshi, 1988).

In the following, the word “*tree automaton*” (TA) will be used as a generic term for an automaton that accepts trees as input. It can be a finite tree automaton, a pushdown tree automaton, or any tree-accepting automaton having a state set, state transitions, initial and final states, and optional memories associated with states. Although our argument below does not necessarily require understanding of these general TAs, definitions and properties of finite and pushdown TAs can be found in Gécseg and Steinby (1997) for example.

2.1 Definition of LTA-based Grammars

The basic idea of an LTA-based grammar is to associate a tree automaton to each word that defines the set of local trees anchored to the word, instead of associating the trees themselves. The lexicalized tree automaton (LTA) provides a finite representation of a possibly non-finite set of local trees. This differs from other lexicalized grammars as LTAG, where non-finiteness of local trees is introduced through a global tree operation such as adjunction of auxiliary trees.

We define a lexicalized tree automata-based grammar as follows. Let Σ be a set of terminal symbols (words), and NT be the set of nonterminal symbols disjoint from Σ . Let T_w be a set of trees (elementary trees) associated with a word w in Σ . A tree in T_w has nodes either from Σ or from NT , and its root and one of its leaves are marked by a distinguished symbol *self* in NT . Let A_w be the tree automaton lexicalized to the word w , which accepts a subset of trees obtained by repeatedly joining two trees in T_w at the special nodes labelled *self*, one at the root of a tree and the other at a foot of another tree. From this definition, A_w can be identified with a string automaton; its alphabets are the trees in T_w , and a string of the elementary trees are identified with the tree obtained by joining the elementary trees in a bottom-up manner. S_w is a set of nonterminal symbols associated with the word w . They are assigned to the root of a tree when the tree is accepted by A_w .

For each word w , the set $A_w = \{T_w, A_w, S_w\}$ is the set of local trees associated with w . The structure is described by A_w and T_w , the symbol at the root node

is from S_w , and *self* in the foot is identified with w . We denote the family of A_w as $A = \{A_w\}$ for w in Σ . A lexicalized tree automata-based grammar G is defined to be the tree algebra whose trees are the set union of A_w for all w in Σ , and the basic tree operation being the substitution, that is, joining two trees at the root node and a foot node when they have the same nonterminal in NT other than *self*.

2.2 Some Remarks

1 Strictly speaking, the definition above does not satisfy the definition of strongly lexicalized grammars that the structures associated to a word must form a finite set, since the tree set accepted by the automaton may be an infinite set. However, since a finite device, namely an automaton, describes this possibly infinite set, we will classify the proposed formalism as a strongly lexicalized grammar.

2 We defined the lexicalized tree automata using string automata where the alphabets are trees. The latter is obtained by linearizing the constituent trees along the spine of the tree. Because the LTA can be any tree automaton as long as it accepts all and only the (possibly infinite) tree set headed by a word, LTA are essentially tree automata. These equivalent two pictures (the tree automata picture of a tree grammar and the string automata picture employed in the definition) will be used interchangeably in this paper.

3 The grammar G can also be defined by a tree automaton T that accepts all and only the trees of the grammar as follows: First we regard NT as the set of states of T . Its initial states are Σ , and the final states are also NT . S_w is regarded as the set of final states of A_w . The set of initial states of A_w are the set of nonterminal symbols that appear in T_w and w . The LTAs are combined into T through the common state set NT . The recognition of a tree t proceeds in a bottom-up manner, beginning at the leaf nodes that are initial states for G and for some A_w . When a subtree of T has been recognized by an LTA A_w , its root node is in a state s from S_w . If s is

Figure 1. General form of X-bar theory

an initial state of another LTA A_w , the recognition can proceed. The tree t has been successfully recognized if the recognition step arrives at the root node.

2.3 Examples

Adjunction in the X-bar Theory

We demonstrate how the proposed formalism handles the simplest case of an infinite set of the local trees. The example is repeated adjunction at the bar level 1 of the X-bar theory. Figure 1 shows a general scheme of the X-bar theory. X' at the bar level 1 can be paired with some adjunct arbitrary times before it grows to the phrase level, XP.

Figure 2 shows how this scheme is realized in the LTA-based grammar formalism. Figure 2 (a) shows the tree set associated with the word. It consists of three trees, corresponding to the bar levels. T_1 is for the complement, T_2 for adjunction, and T_3 for the specifier. (b) shows the tree automaton associated with this word in the (tree-alphabet) string automaton representation. It first accepts T_1 , then T_2 arbitrary times, finally T_3 to arrive at the final state. This sequence is identified with the trees in (d), obtained by concatenating T_1 through T_3 in a bottom-up manner. When the LTA arrives at the final state, the root node is given a nonterminal symbol from the set in (c), which is XP.

Tree Adjoining Language

Figure 3 shows a LTAG that generates a strictly context sensitive language $a^n b^n c^n d^n$. The unique initial tree T in (a) is adjoined repeatedly by the unique auxiliary tree A in (b) at the root node labeled S . The root and foot of A is labeled S , but adjunction to them is inhibited by the index NA. (c) shows a tree obtained by adjoining A once to T . Generally a string $a^n b^n c^n d^n$ is obtained as the yield

Figure 2. LTA-representation of the tree in Figure 1.

Figure 3: LTAG for $a^n b^n e c^n d^n$

Figure 4: LTA-based grammar for $a^n b^n e c^n d^n$

of a tree produced by adjoining A n -times to T .

The same language can be expressed by an LTA-based grammar shown in Figure 4. The word is e (a). The tree set associated with e consists of two trees T_1 and T_2 as shown in (b). The local automaton is a pushdown automaton that accepts tree sequence $(T_2)^n (T_1)^n$, and accepted trees are given the nonterminal symbol S as in (d). (e) shows a tree with $n = 2$. From this setting, it is apparent that this LTA-based grammar generates the same language as the LTAG in the figure 3.

By extending this construction, it will be obvious that for any LTAG, an equivalent LTA-based grammar can be constructed within the class of pushdown LTA.

2.4 Parsing LTA-based Grammars

Parsing algorithm for the LTA-based grammar is a straightforward extension of the CFG case. In the CFG case, an active edge is represented by a rewriting rule with a dot in the right hand side. The dot shows that the terminals and non-terminals up to that location have been already found, and the rule application becomes a success when the dot reaches the end of the rule. If we regard the right-hand side of a rule as an automaton accepting

a sequence of terminals and non-terminals, with the dot representing the current state, this picture can be easily generalized to the LTA-based grammar case.

Figure 7 shows an example parse for the sentence “He eats dinner”. Figure 5 shows the dictionary content of the verb “eats”, which is basically the same as figure 2. Figure 6 shows the dictionary of “he” and “dinner”. We suppose here that these words have no associated trees for simplicity. The basic strategy is left to right bottom-up chart parsing.

First, edges e_1 , e_2 and e_3 are loaded into the chart and set to the initial state. They correspond to “he”, “eats” and “dinner” respectively. The parsing proceeds from left to right, and the parser triggers the LTA of e_1 first. Since its only possible transition is a null transition, it arrives at the final state immediately and creates an edge e_4 labelled *subj*.

Then the focus moves one step to the right on the chart and the LTA of e_2 is activated. It tries to find the tree T_{a1} , and finds that an edge labelled *dob* is necessary to its right. Since there is no such edge, the LTA creates an active edge¹ with a hole *dob*, as in the case of CFG, and the LTA goes into a pause waiting for the hole to be filled.

Creation of e_5 from e_3 is similar to the creation of e_4 from e_1 . Then e_5 starts the completion step as in the CFG case. At this step, the active edge created above is found, and e_5 is found to match the hole. Then the LTA of e_2 is reactivated, arrives at the state S_1 , then creates an edge e_6 .

Next the LTA of e_6 is activated. It tries to find T_{a2} or T_{a3} . In searching for T_{a3} , an active edge with a hole *postmod* is created. While searching for T_{a2} , the LTA finds that an edge labelled *subj* to the left

Figure 5: LTA of “eats”

¹ Active edges are not shown in the figure.

Figure 6: LTAs of "he" and "dinner"

is what is necessary, and finds that e_4 satisfies this condition. By accepting T_{a2} , the LTA creates the edge e_7 , label it as *sentence* and advances to the final state. There is no more possible action on the chart, and the parsing is completed successfully.

Please note that the algorithm exemplified above does not depend on the concrete form of LTA. The same algorithm can be applied to pushdown automata and other class of automata having internal memories.

3 Translation Module

We built a bi-directional translation system between the Japanese and English languages using the proposed method. It translates conversational texts as will appear in a dialogue between two people, to help them communicate in a foreign travel situation. Figure 8 shows an overview of the system.

3.1 Translation Engine

Since each word in the dictionary has its own tree set and tree automaton, a simple implementation will lead to inefficiency. To cope with this problem, we provided two mechanisms to share the LTA. A "rule template" mechanism is provided to share the triplet, namely A_w in the definition of LTA, while a "shared tree" mechanism is provided to share the elementary trees among different A_w .

The rule template is applied just after dictionary loading, and assigns an LTA to a word that matches the condition in the template. It is mainly used for words such as common nouns. A shared tree is represented by a pointer to an elementary tree in the pool, and is loaded into the system when it is used for the first time.

The language conversion method is based on synchronous derivation of analysis and generation trees, basically the same as the syntax directed translation by Aho and Ullman (1969) and the synchronous LTAG by Shieber and Schabes (1990). In this method, elementary analysis tree of each word is paired with another tree (elementary generation tree). Starting from the root, at each

Figure 7: Example Parse

node of the analysis tree, the direct descendant nodes are reordered in the generation tree, according to the correspondence of elementary analysis and generation trees. This translation mechanism is basically independent of how the analysis tree is constructed, hence the grammar formalism. In our implementation, the generation tree is a call graph of target language generating functions, which enables detailed procedural control over the syntactic generation process.

3.2 Grammars and Dictionaries for English to Japanese Translation

The English to Japanese translation grammar and dictionary has been developed. In order to achieve wide coverage for general input and high quality for the target domain, we developed general grammar rules and domain-specific rules simultaneously. General rules are based on a standard X-bar theory. Nodes of a tree are associated with attribute-value structure in a standard way. As nonterminals, we employed a grammatical function-centered approach as Link Grammar (Sleator and Temperley, 1991). A phrase

Figure 8: Translation Module

level node is assigned attribute-values that express their syntactic function such as subject, direct object, etc, instead of a single part-of-speech symbol such as NP. This approach is suitable to capture idiosyncratic behaviour of words.

Domain-specific rules are mostly pattern-like rules with special attention to aspects that are important for carrying conversations, such as modality and the degree of politeness. The English to Japanese translation dictionary contains about seventy thousand words. The number of words that required individual LTA was a few thousand at the time of this report.

3.3 Current Status of Implementation

The system has been implemented using C++, and runs on Windows 98 and NT. The requirement is Pentium II 400MHz or above for the CPU, about 60 MB of memory, and 200 MB of disk space. Most of the disk space is used for statistical data for disambiguation.

We performed a preliminary evaluation of the translation quality of English to Japanese translation. A widely used commercial system was chosen as a reference system, of which the dictionaries were expanded for the target domain. Five hundred sentences were randomly chosen from a large (about 40K) pool of conversational texts of the target domain. Then the output of our system and the reference system were mixed, and then presented to a single evaluator at a random order. The evaluator classified them into four levels (natural, good, understandable and bad). The result showed that the number of sentences classified to "natural" increased about 45% compared to that of the reference system, i.e. the ratio of the number of sentences was around 1.45. The number of sentences classified as "bad" decreased about 40% in the same measure.

We applied this module to an experimental speech translation system (Watanabe et al., 2000).

4 Discussions

The proposed grammar formalism is a kind of lexicalized grammar formalism and shares its advantages. The largest difference from other strongly lexicalized grammar formalisms is that it employs lexicalized tree automata (LTA) to describe the tree set associated with a word, which allows a finite description of a non-finite set of local trees. These automata's role is equivalent to additional tree operations in other formalisms. In

addition, an LTA provide an extended domain of locality (EDOL) of the word.

If all the LTAs are finite automata in the string automaton representation, then the tree language recognized by this grammar is regular and its yield is a context-free language. The grammar can accept general Tree Adjoining Language (TAL) if the LTAs belong to the class of pushdown automata in the string automaton representation. This is a reflection of the fact that pushdown tree automata can accept the indexed languages (Gécseg and Steinby, 1997), of which the TAL is a subclass.

As shown in the section 2.4, the control strategy of bottom-up chart parsing does not rely on the concrete content of the LTA, which is an advantage of the proposed formalism. This implies that we can alter even the grammar class without affecting the parsing. Suppose the current LTAs are finite automata, hence the yield language is context-free. If we want to introduce a word e that induces a non-context-freeness, such as e in $a^*b^*ec^*d^*$, then what we have to do is to write a pushdown automaton in the figure 4 for the word e . We change neither the grammar formalism nor the parsing algorithm, and the change is localized to the LTA of e .

Writing automata by hand may seem much more complex than writing trees, but our experience shows that it is not much different from conventional grammar development. As long as appropriate notations are used, writing automata for a word amounts to determining possible form of trees headed by that word, a task always required in grammar development. In fact, there is less amount of work since the grammar writer does not need to pay attention to assigning proper nonterminals and/or proper attributes to internal nodes of trees in order to control their growth.

It is another advantage of the proposed formalism that it can utilize various automata operations, such as composition and intersection. For example, a word can append an automaton to that of the headword when it becomes a child, which enables to specify a constraint from a lower-positioned word to a higher-positioned word in the tree. Another example is coordination. Two edges are conjoined when the unapplied parts of LTAs have nonempty intersection as automata, and the conjoined edge is given with this intersection as the LTA. Verb phrase conjunction such as "John eats cookies and drinks beer" is handled in this manner, by conjoining "eats cookies" and "drinks beer". The intersected automaton will accept the subject tree and other sentence-level trees.

In the proposed method, elementary trees are always anchored by the syntactic headword. For example, a verb in a relative clause is in the EDOL of the antecedent. Then, if the embedded verb puts a constraint on the antecedent, that constraint is not expressed in a straightforward manner, which may seem a weakness of the method. We just point out that this type of problem occurs when the syntactic head and the semantic head are different, and is common to lexicalized grammars as long as a tree is anchored to one word, because constraints are often reciprocal. In our current implementation, the constraint written in the verb's dictionary is found and checked by the relative-clause-tree accepting automaton of the antecedent noun.

There have been many work on syntactic analysis based on automata attached to the headword. Evans and Weir (1998) used finite state automata as representation of trees that can be merged and minimized to improve parsing efficiency. In their method, the grammar is fixed to be LTAG or some lexicalized grammar and the automata are obtained by automatic conversion from the trees. Our method differs from theirs in the point that ours employs trees as the basic object of automata, which enables to handle general recursive adjunction in LTAG, while their automata work on the nonterminal and terminal symbols. In the center of our method is the notion of the local grammar of a word. The whole grammar is divided into the global part and the set of local grammars specific to the words, which is represented by the LTAs.

Alshawi (1996) introduced Head Automata, a weighted finite machine that accepts a pair of sequences of relation symbols. The difference is similar as above. Since the tree automata in our method are used to define the set of the local trees, their role will be equivalent to building the head automata themselves, but not to combining the trees that are already built, like the Head Automata.

5 Conclusion

We proposed a new lexicalized grammar formalism, called Lexicalized Tree Automata-based Grammar. In this formalism, the trees anchored to a word are described by a tree automaton associated with it. We showed a chart parsing algorithm that does not depend on the concrete content of the automata. We have implemented a bi-directional translation module between Japanese and English for conversational texts using this formalism. A preliminary evaluation of English to Japanese translation quality revealed a promising result.

Acknowledgement

We would like to thank Shinichiro Kamei for useful discussions and Yoshinori Ishihara for his help in the implementation work.

References

- Abeillé, A., Schabes, Y. and Joshi, A.K. (1990). *Using Lexicalized TAGs for Machine Translation*. In *Proceedings of COLING-90*, pp.1-6.
- Aho, A.V., and Ullman, J.D. (1969). *Properties of Syntax Directed Translations*. *Journal of Computer and System Sciences*, vol.3, pp.319-334.
- Alshawi, H. (1996). *Head Automata and Bilingual Tiling: Translation with Minimal Representations*. In *Proceedings of 34th Annual Meeting of Computational Linguistics*, pp.167-176.
- Brown, R.D. (1996). *Example-Based Machine Translation in the Pangloss System*. In *Proceedings of COLING-96*, pp.169-174.
- Evans, R. and Weir, D.J. (1998). *A Structure-Sharing Parser for Lexicalized Grammars*. In *Proceedings of COLING-ACL '98*, pp.372-378.
- Furuse, O. and Iida, H. (1994). *Constituent Boundary Parsing for Example-Based Machine Translation*. In *Proceedings of COLING-94*, pp.105-111.
- Gécseg, F. and Steinby, M. (1997). *Tree Languages*. In *Handbook of Formal Languages*, G. Rosenberg and A. Salomaa, editors, Springer Verlag, Vol.3, pp.1-68.
- Joshi, A.K. and Schabes, Y. (1992). *Tree-Adjoining Grammars and Lexicalized Grammars*. In *Tree Automata and Languages*, M. Nivat and A. Podelski, ed., Elsevier Science Publishers B.V., pp.409-431.
- Ruland, T., Rupp, C.J., Spilker, J., Weber, H. and Worm, K. (1998). *Making The Most of Multiplicity: A Multi-Parser Multi-Strategy Architecture for the Robust Processing of Spoken Language*. In *Proceedings of ICSLP'98*, pp.1163-1166.
- Schabes, Y., Abeillé, A. and Joshi, A. (1988). *Parsing Strategies with 'Lexicalized' Grammars*. In *Proceedings of COLING'88*, pp.578-583.
- Schabes, Y. and Waters, R.C. (1995). *Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced*. *Computational Linguistics*, Vol. 21, pp.479-513.
- Shieber, S.M. and Schabes, Y. (1990). *Synchronous Tree Adjoining Grammars*. In *Proceedings of COLING'90*, pp.253-258.
- Sleator, D.D.K. and Temperley, D. (1991). *Parsing English with a Link Grammar*. CMU Technical Report CMU-CS-91-196.
- Takeda, K. (1996). *Pattern-Based Machine Translation*. In *Proceedings of COLING '96*, pp.1155-1158.
- Watanabe, T., Okumura, A., Sakai, S., Yamabana, K. and Doi, S. (2000). *Automatic Interpretation*. To appear in *NEC Technical Journal*, Vol.53, No.6. (in Japanese).