Pattern-Based Machine Translation

Koichi Takeda
Tokyo Research Laboratory, IBM Research
1623-14 Shimotsuruma, Yamato, Kanagawa 242, Japan
Phone: 81-462-73-4569, 81-462-73-7413 (FAX)
takeda@trl.vnet.ibm.com

Abstract

In this paper, we describe a "pattern-based" machine translation (MT) approach that we followed in designing a personal tool for users who have access to large volumes of text in languages other than their own, such as WWW pages. Some of the critical issues involved in the design of such a tool include easy customization for diverse domains, the efficiency of the translation algorithm, and scalability (incremental improvement in translation quality through user interaction). We also describe how our patterns fit into the context-free parsing and generation algorithms, and how we implemented a prototype tool.

1 Introduction

It would be difficult for anyone to dispute the idea that the World-Wide Web (WWW) has been the most phenomenal invention of the last decade in the computing environment. It has suddenly opened up a window to vast amounts of data on the Internet. Unfortunately for those who are not native English speakers, textual data are more often than not written in a foreign language.

A dozen or so machine translation (MT) tools have recently been put on the market, to make such textual data more accessible, but novice PC users will be simply amazed at the meagerness of their reward for the effort of building a so-called "user dictionary."

The main reasons for this problem are:

1. Most MT systems do not employ a powerful "lexicalist" formalism.
2. Most MT systems can be customized only by adding a user dictionary.

Therefore, users can neither give preferences on individual prepositional-phrase attachments (e.g., to obtain information *from a server*) nor define translations of specific verb-object pairs (e.g, to *take advantage* of *something*).

Powerful grammar formalisms and lexical-semantics formalisms have been known for years (see LFG(Kaplan and Bresnan, 1982), HPSG(Pollard and Sag, 1987), and Generative Lexicon(Pustejovsky, 1991), for example), but practical implementation of an MT system has yet to tackle the computational complexity of parsing algorithms for these formalisms and the workload of building a large-scale lexicon.

Example-based MT(Sato and Nagao, 1990; Sumita and Iida, 1991) and statistical MT(Brown et al., 1993) are both promising approaches that generally demonstrate incremental improvement in translation accuracy as the quality of examples or training data grows. It is, however, an open question whether these approaches alone can be used to create a full-fledged MT system; that is, it is uncertain whether such a system can be used for various domains without showing severe degradation in translation accuracy, or if it has to be fed by a reasonably large set of examples or training data for each new domain.

TAG-based MT(Abeillé et al., 1990)[1] and pattern-based translation(Maruyama, 1993) share many important properties for successful implementation in practical MT systems, namely:

- The existence of a polynomial-time parsing algorithm
- A capability for describing a larger *domain of locality*
- *Synchronization* of the source and target language structures

In this paper, we show that there exists an attractive way of crossing these approaches, which we call *pattern-based* MT.[2] In the following two sections, we introduce a class of translation "patterns" based on Context-Free Grammar (CFG), and a parsing algorithm with $O(|G|^2n^4)$ worst-case time complexity. Furthermore, we show that our framework can be extended to incorporate example-based MT and a powerful learning mechanism.

2 Translation Patterns

A translation *pattern* is defined as a pair of CFG rules, and zero or more *syntactic head* and *link* constraints for nonterminal symbols. For example, the English-French translation pattern[3]

NP:1 miss:V:2 NP:3 → S:2 S:2 ← NP:3 manquer:V:2 *à* NP:1

essentially describes a *synchronized*[4] pair consisting of a left-hand-side English CFG rule (called a *source* rule)

NP V NP → S

and a right-hand-side French CFG rule (called a *target* rule)

S ← NP V *à* NP

accompanied by the following constraints:

1. **Head constraints:** The nonterminal symbol V in the source rule must have the verb "miss" as a syntactic head. The symbol V in the target rule must have the verb "manquer" as a syntactic head. The head of symbol S in the source (target) rule is identical to the head of symbol V in the source (target) rule, as they are co-indexed. Head constraints can be specified in either or both sides of the patterns.
2. **Link constraints:** Nonterminal symbols in source and target CFG rules are *linked* if they are given the same index ": *i*". Thus, the first NP (NP:1) in the source rule corresponds to the second NP (NP:1) in the target rule, the Vs in both rules correspond to each other, and the second NP (NP:3) in the source rule corresponds to the first NP (NP:3) in the target rule.

[1]See LTAG(Schabes et al., 1988)(Lexicalized TAG) and STAG(Shieber and Schabes, 1990)(Synchronized TAG) for each member of the TAG (Tree Adjoining Grammar) family.

[2]Recently, Tree Insertion Grammar(Schabes and Waters, 1995) has been introduced to show a similar possibility. Our approach, however, is more inclined toward the CFG formalism.

[3]And its inflectional variants - we will discuss agreement issues later, in the "Extended Formalism" section.

[4]The meaning of the word "synchronized" here is exactly the same as in STAG(Shieber and Schabes, 1990).

The source and target rules, that is, the CFG rules with no constraints, are called the *CFG skeleton* of the patterns. The notion of a syntactic head is similar to that used in unification grammars, although the heads in our patterns are simply encoded as character strings rather than as complex feature structures. A head is typically introduced[5] in preterminal rules such as

leave $\rightarrow$ V V $\leftarrow$ partir

where two verbs, "leave" and "partir," are associated with the heads of the nonterminal symbol V. This is equivalently expressed as

leave:1 $\rightarrow$ V:1 V:1 $\leftarrow$ partir:1

which is physically implemented as an entry of a lexicon.

A set T of translation patterns is said to *accept* an input s iff there is a derivation sequence Q for s using the source CFG skeletons of T, and every head constraint associated with the CFG skeletons in Q is satisfied. Similarly, T is said to *translate* s iff there is a synchronized derivation sequence Q for s such that T accepts s, and every head and link constraint associated with the source and target CFG skeletons in Q is satisfied. The derivation Q then produces a translation t as the resulting sequence of terminal symbols included in the target CFG skeletons in Q. Translation of an input string s essentially consists of the following three steps:

- Parsing s by using the source CFG skeletons
- Propagating link constraints from source to target CFG skeletons to build a target CFG derivation sequence
- Generating t from the target CFG derivation sequence

The third step is trivial as in the case of STAG translation.

Some immediate results follow from the above definitions.(Takeda, 1996)

1. Let a CFG grammar G be a set of source CFG skeletons in T. Then, T accepts a context-free language (CFL), denoted by $L(T)$, such that $L(T) \subseteq L(G)$.
2. Let a CFG grammar H be a subset of source CFG skeletons in T such that a source CFG skeleton k is in H iff k has no head constraints associated with it. Then, H accepts a subset $L(H)$ of language $L(T)$.
3. $L(T)$ is a proper subset of $L(G)$ if, for example, there exists a pattern p ($\in$ T) with a source CFG rule $X \rightarrow X_1 \cdots X_k$ such that[6]
   (a) p has a head constraint $h : X$ for some nonterminal symbol X_i ($i = 1, 2, \ldots, k$).
   (b) T has a derivation sequence $X \rightarrow \cdots \rightarrow w$ such that X is associated with a head g ($h \neq g$), and T has no sequence of nonterminal symbols $Y_1 \ldots Y_l$ that derives exactly the same set of strings as X does.

[5]A nonterminal symbol X in a source or target CFG rule $X \rightarrow X_1 \cdots X_k$ can only be constrained to have one of the heads in the RHS $X_1 \cdots X_k$. Thus, *monotonicity* of head constraints holds throughout the parsing process.

[6]This is not a necessary condition for $L(T) \subset L(G)$. It is provable that for any set T of patterns, there exists a (weakly) equivalent CFG grammar F, with possibly exponentially more grammar rules, such that $L(T) = L(F)$. A decision problem of two CFLs, $L(T) \subset L(G)$, is solvable iff $L(F) = L(G)$. This includes an undecidable problem, $L(F) = \Sigma^*$. Therefore, we can conclude that $L(T) \subset L(G)$ is undecidable. Similar discussions can be found in the literature on Generalized Phrase Structure Grammar(Gazdar et al., 1985).

Although our "patterns" have no more descriptive power than CFG, they can provide considerably better descriptions of the *domain of locality* than ordinary CFG rules. For example,

be:V:1 year:NP:2 old $\rightarrow$ VP:1 VP:1 $\leftarrow$ avoir:V:1 an:NP:2

can handle such NP pairs as "one year" and "un an," and "more than two years" and "plus que deux ans," which would have to be covered by a large number of plain CFG rules. TAGs, on the other hand, are known to be "mildly context-sensitive" grammars, and they can capture a wider range of syntactic dependencies, such as cross-serial dependencies. The computational complexity of parsing for TAGs, however, is $O(|G|n^6)$, which is far greater than that of CFG parsing. Moreover, defining a new STAG rule is not as easy for the users as just adding an entry into a dictionary, because each STAG rule has to be specified as a pair of syntactic tree structures. Our patterns, on the other hand, can be specified as easily as

to leave * = de quitter *
to be year:* old = d'avoir an:*

by the users. Here, the wildcard "*" stands for an NP by default. The prepositions "to" and "de" are merely used to specify that these patterns are for VPs, and they are removed when compiled into internal forms so that these patterns are applicable to finite as well as infinite forms. Similarly, "to be" is used to show that the phrase is a be-verb and its complement. The wildcards can be constrained with a head, as in "year:*" and "an:*". In addition, they can be associated with an explicit nonterminal symbol such as "V:*" or "ADJP:*" (e.g., "leave:V:*"). By defining a few such notations, these patterns can be successfully converted into the formal representations defined above. The notations are so simple that even a novice PC user should have no trouble in writing our patterns, as if he or she were making a vocabulary list for English or French exams.

3 Pattern-Based Translation Algorithm

A parsing algorithm for translation patterns can be any of the known CFG parsing algorithms, including CKY and Earley algorithms. It should be first noted, however, that CFG could produce exponentially ambiguous parses for some input, in which case we can only apply heuristic or stochastic measurement to select the most promising parse.

It is known that an Earley-based parsing algorithm can be made to run in $O(|G|Kn^3)$ rather than $O(|G|^2n^3)$,(Maruyama, 1993; Graham et al., 1980) where K is the number of distinct nonterminal symbols in the grammar G. We can expect a very efficient parser for our patterns.[7] The input string can also be scanned to reduce the number of relevant grammar rules before parsing.[8] The combined process is also known as offline-parsing in LTAG.

Handling ambiguous parses is a difficult task. The basic strategy for choosing a candidate parse during Earley-based parsing is as follows:

1. Prefer a pattern p with a source CFG skeleton $X \rightarrow X_1 \cdots X_k$ over any other pattern q such that the source CFG skeleton of q is $X \rightarrow X_1 \cdots X_k$, and such that X_i in p has a head constraint h if q has $h : X_i$ ($i = 1, \ldots, k$). The pattern p is said to be *more specific* than q. This relation is similar to a subsumption relationship(Pollard and Sag, 1987).

[7]Schabes and Waters(Schabes and Waters, 1995) also discuss several techniques for optimizing parsing algorithms.

[8]Such scanning is essential for some languages with no explicit word boundaries (such as Japanese and Chinese).

2. Prefer a pattern p with a source CFG skeleton over one with fewer terminal symbols than p.

3. Prefer a pattern p that does not violate any head constraint over one that violates a head constraint.

4. Prefer the shortest derivation sequence for each input substring. A pattern for a larger domain of locality tends to give a shorter derivation sequence.

Thus, our strategy favors *lexicalized* (or head-constrained) and *collocational* patterns, which is exactly what we are going to achieve with pattern-based MT. Selection of patterns in the derivation sequence accompanies the construction of a target derivation sequence. Link constraints are propagated from source to target derivation trees. This is basically a bottom-up procedure.

Since the number M of distinct pairs ⟨X,w⟩, for a nonterminal symbol (or a *chart*) X and a subsequence w of input string s, is bounded by Kn^2, there are at most Kn^3 possible triples ⟨X,w,h⟩, such that h is a head of X. Thus, we can compute the *m-best choice* of translation candidates in $O(|T|Kn^4)$ time. Here, K is the number of distinct nonterminal symbols in T, and n is the size of the input string.

The reader should note critical differences between lexicalized grammar rules (in the sense of LTAG) and translation patterns when they are used for MT.

Firstly, a pattern is not necessarily lexicalized. An economical way of organizing translation patterns is to include non-lexicalized patterns as "default" translation rules. For example, the pattern

V:1 NP:2 → VP:1 VP:1 ← V:1 NP:2

is used as a default translation of "verb + direct object" expressions, but

resemble:V:1 NP:2 → VP:1 VP:1 ← resembler:V:1 à NP:2

is always preferred over the default rule because of our preference strategy. Similarly, the pattern

please VP:1 → VP:1 VP:1 ← VP:1 , s'il vous plaît

should be preferred over a lexicalized pattern, if any,

ADVP:1 xxx:VP:2 → VP:2 VP:2 ← ADVP:1 yyy:VP:2

Secondly, lexicalization might considerably increase the size of STAG grammars (in particular, compositional grammar rules such as ADJP NP → NP) when a large number of lexical items are associated with them. Since it is not unusual for a noun in a source language to have several counterparts in a target language, the number of tree-pairs in STAG would grow much larger than that of source LTAG trees. Although in LTAG the grammar rules are differentiated from their physical objects ("parser rules"), and "structure sharing"(Vijay-Shanker and Schabes, 1992) is proposed, this ambiguity remains in the parser rules, too.

Thirdly, a translation pattern can omit the tree structure of a collocation, leaving it as just a sequence of terminal symbols. For example,

See you later, NP:1 → S S ← Au revoir, NP:1

is perfectly acceptable as a translation pattern.

4 Extended Formalism

Syntactic dependencies in natural language sentences are so subtle that many powerful grammar formalisms have been proposed to account for them. The adequacy of CFG for describing natural language syntax has long been questioned, and unification grammars, among others, have been used to build a precise theory of the computational aspects of syntactic dependencies, which are described by the notion of unification and by feature structures.

Translation patterns can also be extended by means of unification and feature structures. Such extensions must be carefully applied so that they do not sacrifice the efficiency of parsing and generation algorithms. Shieber and Schabes briefly discuss the issue(Shieber and Schabes, 1990). We can also extend translation patterns as follows:

> Each nonterminal node in a pattern can be associated with a fixed-length *vector* of *binary features*.

This will enable us to specify such syntactic dependencies as agreement and subcategorization in patterns. Unification of binary features, however, is much simpler: unification of a feature-value pair succeeds only when the pair is either ⟨0,0⟩ or ⟨1,1⟩. Since the feature vector has a fixed length, unification of two feature vectors is performed in a constant time. For example, the patterns

V:1:+TRANS NP:2 → VP:1 VP:1 ← V:1:+TRANS NP:2

V:1:+INTRANS → VP:1 VP:1 ← V:1:+INTRANS

are unifiable with transitive and intransitive verbs, respectively. We can also distinguish *local* and *head* features, as postulated in HPSG. Verb subcategorization is then encoded as

VP:1:+TRANS-OBJ NP:2 → VP:1:+OBJ

VP:1:+OBJ ← VP:1:+TRANS-OBJ NP:2

where "-OBJ" is a local feature for head VPs in LHSs, while "+OBJ" is a head feature for VPs in the RHSs. Unification of a head feature with +OBJ succeeds when it is not *bound*.

Another extension is to associate weights with patterns. It is then possible to rank the matching patterns according to a linear ordering of the weights rather than the pairwise partial ordering of patterns described in the previous section. Numeric weights for patterns are extremely useful as a means of assigning higher priorities to user-defined patterns.

The final extension of translation patterns is integration of examples, or *bilingual corpora*, into our framework. It consists of the following steps. Let T be a set of translation patterns, B a bilingual corpus, and ⟨s,t⟩ a pair of source and target sentences.

1. If T can translate s into t, do nothing.

2. If T can translate s into t' ($t \neq t'$), do the following:

   (a) If there is a paired derivation sequence Q of ⟨s,t⟩ in T, create a new pattern p' for a pattern p used in Q such that every nonterminal symbol X in p with no head constraint is associated with $h : X$ in q, where the head h is instantiated in X of p. Add p' to T if it is not already there.

   (b) If there is no such paired derivation sequence, add the pair to T ⟨s,t⟩ as a translation pattern.

3. If T cannot translate s, add the pair ⟨s,t⟩ to T as a translation pattern.

The simplest way of integrating the corpus B into T is just to consider the sentence pair ⟨s,t⟩ as a translation pattern. Some additional steps are necessary to achieve higher MT accuracy for a slightly wider range of sentences than those included in B. However, the degree of improvement in MT accuracy that can be achieved with this learning mechanism is open to question, since the addition of translation patterns does not necessarily guarantee a monotonic improvement in MT accuracy.

5 Implementation

Our experimental implementation of a pattern-based MT system consists of about 500 default-translation patterns, about 2400 idiomatic and collocational patterns, and about 60,000 lexical items for English-to-Japanese translation. A sample run of the prototype system is shown in Figure 1. It shows one of the derivation sequences for the input sentence

> John should hear from Mary about the news if he returns home.

Each line in the derivation sequence shows an English source CFG rule of a pattern used for the derivation. For example, the first line

```
[(0 13) S:1:/eFIN,ePRES,eSUBJ,eAUX/
   -> S1:1:+eFIN PUNCT:2
```

in the derivation sequence shows that two nonterminal symbols, S1 and PUNCT, form a sentence S, that S is co-indexed with S1, and that S1 must have a *finite* form feature +eFIN. The current instance of S has four features — *finite, present (ePRES), with-subject (eSUBJ)*, and *with-auxiliary-verb (eAUX)* — and it spans the word positions 0 to 13.[9] We can also find several *head-constrained* patterns there. For example,

```
[(10 12) VP:1:/eFIN,e3SG,ePRES,eOBJ,eSAT/ ->
   VP"return":1:-eOBJ NP"home":2:+eCAUS
```

is a pattern for translating "return:V home:NP". The default V+NP translation pattern will assign a wrong Japanese case marker for this phrase.

Our prototype took about 9 sec (elapsed time) to translate this input sentence and produce seven alternative translations. The derivation shown in the figure was the first (i.e., the best), and generates a correct translation. Therefore, collocational patterns and default patterns have been appropriately combined under our preference strategy.

6 Conclusions and Future Work

In this paper, we have proposed a pattern-based MT system that satisfies three essential requirements of the current market: efficiency, scalability, and ease-of-use. We are aware that CFG-based patterns are less adequate for describing syntactic dependencies than linguistically motivated grammar formalisms such as TAGs and HPSG. To achieve the best possible average runtime and accuracy, perhaps our pattern-based system should be combined with more powerful grammar formalisms. We believe that the theory and implementation of pattern-based MT will contribute to the realization of computational linguistic theories. A corpus integration method to verify efficiency of the grammar acquisition has yet to be implemented.

Some of the assumptions on patterns should be reexamined when we extend the definition of patterns. The notion of Head constraints may have to be extended into that of a set membership constraint if we need to handle coordinated structures. Some light-verb phrases cannot be correctly translated without "exchanging" several feature values between the verb and its object. A similar problem has been found in be-verb phrases.

[9] Other features include *nominative* and *causative* cases, 3rd-person-singular forms, and capitalized words. Two features, "*eARGS" and "*eARGV," are special ones for representing subject-verb agreement without splitting a pattern into an equivalent set of several patterns for a specific type of agreement. This source derivation sequence is actually accompanied by its Japanese counterpart, which was omitted due to the space limitation.

```
> John should hear from Mary about the news
  if he returns home.

 [(0 13) S:1:/eFIN,ePRES,eSUBJ,eAUX/ -> S1:1:+eFIN PUNCT:2
   [(0 12) S1:2:/eFIN,ePRES,eSUBJ,eAUX/ ->
      NP:1:*eAGRS+eNOMI VP:2:*eAGRV+eFIN-eSUBJ
     [(0 1) NP:1:/e3SG,eCAP,eNOMI,eCAUS/ -> NOUN:1:-ePRO
       [(0 1) NOUN:1:/e3SG,eCAP/ -> NOUN"John"]]
     [(1 12) VP:1:/eFIN,ePRES,eAUX/ -> VP:1 SADJ:2
       [(1 8) VP:1:/eFIN,ePRES,eAUX/ -> VP:1 PP:2
         [(1 5) VP:1:/eFIN,ePRES,eAUX/ -> VP:1: PP:2
           [(1 3) VP:1:/eFIN,ePRES,eAUX/ ->
               AUX"should":-eNEG VP:1:+eINF-eSUBJ-eAUX
             [(1 2) AUX:1:/eFIN/ -> AUX"should":1]
             [(2 3) VP:1:/eFIN,eINF/ -> VERB:1:-ePS
               [(2 3) VERB:1:/eFIN,eINF/ -> VERB"hear":1]]]
           [(3 5) PP:1:/e3SG,eCAP,eNOMI,eCAUS/ -> "from" NP:1
             [(4 5) NP:1:/e3SG,eCAP,eNOMI,eCAUS/
                     -> NOUN:1:-ePRO
               [(4 5) NOUN:1:/eSG,eCAP/ -> NOUN"Mary":1]]]]
         [(5 8) PP:1:/eDEF,eNOMI,eCAUS,e3SG/ -> "about" NP:1
           [(6 8) NP:1/eDEF,eNOMI,eCAUS,e3SG/
                   -> "the" NP:1:-eDEF-eINDEF
             [(7 8) NP:1:/eNOMI,eCAUS,e3SG/ -> NOUN:1:-ePRO
               [(7 8) NOUN:1:/e3SG/ -> NOUN"news"]]]]]
       [(8 12) SADJ:2:/eFIN,e3SG,ePRES,eSUBJ,eOBJ,eSAT/ ->
           "if" NP:1:*eAGRS+eNOMI VP:2:*eAGRV+eFIN-eSUBJ
         [(9 10) NP:1:/ePRO,eNOMI,e3SG,eHUM/ -> PRON:1:-ePOSS
           [(9 10) PRON:1:/ePRO,eNOMI,e3SG,eHUM/
                   -> PRON"he":1]]
         [(10 12) VP:1:/eFIN,e3SG,ePRES,eOBJ,eSAT/ ->
             VP"return":1:-eOBJ NP"home":2:+eCAUS
           [(10 11) VP:1:/eFIN,e3SG,ePRES/ -> VERB:1:-ePS
             [(10 11) VERB:1:/eFIN,e3SG,ePRES/
                       -> VERB"return":1]]
           [(11 12) NP:1:/e3SG,eNOMI,eCAUS/ -> NOUN:1:-ePRO
             [(11 12) NOUN:1:/e3SG/ -> NOUN"home":1]]]]]]
  [(12 13) PUNCT:1 -> PUNCT".":1]]
```

ジョンは, もし彼が 家に 帰るとすれば,
ニュースを 聞くはずだ.
John+SUBJ, if he+SUBJ home+GOAL return,
news+OBJ hear+should

Figure 1: Sample Parsing

References

A. Abeillé, Y. Schabes, and A. K. Joshi. 1990. "Using Lexicalized Tags for Machine Translation". In Proc. of the 13th International Conference on Computational Linguistics, volume 3, pages 1–6, Aug.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. "The Mathematics of Statistical Machine Translation: Parametric Estimation". Computational Linguistics, 19(2):263–311, June.

G. Gazdar, G. K. Pullum, and I. A. Sag. 1985. "Generalized Phrase Structure Grammar". Harvard University Press, Cambridge, Mass.

S. L. Graham, M. A. Harrison, and W. L. Ruzzo. 1980. "An Improved Context-free Recognizer". ACM Transactions on Programming Languages and Systems. 2(3):415–462, July.

R. Kaplan and J. Bresnan. 1982. "Lexical-Functional Grammar: A Formal System for Generalized Grammatical Representation". In J. Bresnan, editor, "Mental Representation of Grammatical Relations", pages 173–281. MIT Press. Cambridge, Mass.

H. Maruyama. 1993. "Pattern-Based Translation: Context-Free Transducer and Its Applications to Practical NLP". In Proc. of Natural Language Pacific Rim Symposium (NLPRS' 93), pages 232–237. Dec.

C. Pollard and I. A. Sag. 1987. "An Information-Based Syntax and Semantics, Vol.1 Fundamentals". CSLI Lecture Notes, Number 13.

J. Pustejovsky. 1991. "The Generative Lexicon". Computational Linguistics, 17(4):409–441, December.

S. Sato and M. Nagao. 1990. "Toward Memory-based Translation". In Proc. of the 13th International Conference on Computational Linguistics, pages 247–252, Helsinki, Aug.

Y. Schabes and R. C. Waters. 1995. "Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced". Computational Linguistics, 21(4):479–513, Dec.

Y. Schabes, A. Abeillé, and A. K. Joshi. 1988. "Parsing Algorithm with 'lexicalized' grammars: Application to tree adjoining grammars". In Proc. of the 12th International Conference on Computational Linguistics, pages 578–583, Aug.

S. M. Shieber and Y. Schabes. 1990. "Synchronous Tree-Adjoining Grammars". In Proc. of the 13th International Conference on Computational Linguistics, pages 253–258, August.

E. Sumita and H. Iida. 1991. "Experiments and Prospects of Example-Based Machine Translation". In Proc. of the 29th Annual Meeting of the Association for Computational Linguistics, pages 185–192. Berkeley, June.

K. Takeda. 1996. "Pattern-Based Context-Free Grammars for Machine Translation". In Proc. of the 34th Annual Meeting of ACL, Santa Cruz, Calif., June.

K. Vijay Shanker and Y. Schabes. 1992. "Structure Sharing in Lexicalized Tree-Adjoining Grammars". In Proc. of the 14th International Conference on Computational Linguistics, pages 205–211. Aug.