Incremental Translation Utilizing Constituent Boundary Patterns

Osamu FURUSE, Hitoshi IIDA
ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan
{furuse, iida}@itl.atr.co.jp

Abstract

We have proposed an incremental translation method in Transfer-Driven Machine Translation (TDMT). In this method, constituent boundary patterns are applied to an input in a bottom-up fashion. Also, by dealing with best-only substructures, the explosion of structural ambiguity is constrained and an efficient translation of a lengthy input can be achieved. Through preliminary experimentation our new TDMT has been shown to be more efficient while maintaining translation quality.

1 Introduction

A system dealing with spoken language requires a quick response in order to provide smooth communication between humans or between a human and a computer. Therefore, assuring efficiency in spoken-language translation is one of the most crucial tasks in devising such a system. In spoken language, the translation of lengthy utterances can yield a huge amount of structural ambiguity, which needs to be efficiently processed by the system. As a solution for achieving an efficient spoken-language system, several techniques, such as incremental generation (Finkler, 1992; Kempen, 1987) and marker-passing memory-based translation (Kitano, 1994), have been proposed. Many of these techniques adopt a left-to-right strategy to handle an input incrementally and a best-first strategy to avoid the explosion of structural ambiguity. These strategies can be achieved with bottom-up processing.

We have already proposed Transfer-Driven Machine Translation (TDMT) for efficient and robust spoken-language translation (Furuse, 1994a; Furuse, 1994b). However, the top-down and breadth-first translation strategy in the earlier versions of TDMT, which yields a quick response for inputs with restricted lengths, may show poor efficiency when processing a very lengthy input or inputs having many competing structures.

In a top-down and breadth-first application, all the possible structures are retained until the whole input string is parsed. This requires many computations and results in inefficient translation. For instance, the sentence below has many competing structures, mainly because of possible combinations within noun sequences. If this expression is combined with another expression, the structural ambiguity will be further compounded.

With bacon chicken eggs lettuce and tomato on it.

In contrast, if structural ambiguities of substrings are always settled and are never inherited to the upper structures, the explosion of structural ambiguity could be constrained. Thus, an incremental strategy that fixes partial results is necessary for efficient processing and is achieved by bottom-up processing in left-to-right order.

This paper proposes TDMT using an incremental strategy for achieving efficient translation of a lengthy input or one having a lot of structural ambiguity. In this method, several constituent boundary patterns are applied to an input string in a bottom-up fashion. This bottom-up application, based on the concept of chart parsing, can constrain the explosion of structural ambiguity by dealing with best-only substructures using semantic distance calculations.

In this paper, we will first outline our new translation strategy. We will then explain how constituent boundary patterns can be used to describe the structure of an input string in TDMT. Then we will describe the bottom-up pattern application, based on chart parsing. Next, we will show how the explosion of structural ambiguity is constrained by dealing with the best-only substructures, based on semantic distance calculations. By comparing the preliminary experimental results from the former top-down method and those from our new method, we will demonstrate the usefulness of our new method. A summary of our approach will conclude the paper.

2 Translation strategy

In TDMT, translation is performed by applying stored empirical transfer knowledge, which de-

scribes the correspondence between source language expressions and target language expressions at various linguistic levels. The source and target expressions of the transfer knowledge in TDMT are expressed by constituent boundary patterns, which represent meaningful units for linguistic structure and transfer. An efficient application of transfer knowledge source parts to an input string plays a key role in achieving quick translation.

The procedure for applying constituent boundary patterns is perfomed after the assignment of morphological information to each word of an input string, and is as follows:

(a) Insertion of constituent boundary marker;

(b) Derivation of possible structures;

(c) Structural disambiguation by semantic distance calculation.

In the top-down and breadth-first pattern application, the above procedure is executed in the described order. Because the selection of the best structure might have to be postponed until all possible structures are derived, the costs of translation could be high.

In contrast, the incremental method determines the best structure locally and can constrain the number of competing structures for the whole input by performing (b) in parallel with (c); consequently, translation costs are reduced.

The structure selected in (c) contains its transferred result and head word information, which is used for semantic distance calculation when combining with other structures. The output sentence is generated as a translation result from the structure for the whole input, which is composed of best-first substructures.

In the three subsequent sections, we will explain (a), (b), and (c), focusing on the bottom-up and best-first translation strategy.

3 Constituent boundary pattern

In this section we will briefly explain how constituent boundary patterns are used to describe the structure of an input string in TDMT and what procedures are applied before constituent boundary pattern applications (Furuse, 1994b).

We will show bottom-up pattern application by translating the following sample English sentence into Japanese:

The bus goes to Chinatown at ten a.m.

First, all the words in this sequence are assigned the following parts-of-speech.

article, noun, verb, preposition, proper-noun, preposition, numeral, postnominal

A constituent boundary pattern is defined as a sequence that consists of variables and symbols representing constituent boundaries. A variable corresponds to some linguistic constituent and is expressed as a capital letter (e.g. *X*). A constituent boundary is expressed by either a functional word or a part-of-speech bigram marker (e.g. *noun-verb*). Variables in the source language expression must be separated by constituent boundaries.

For instance, the expression "*goes to Chinatown*" is divided into two constituents, i.e. "*goes*" and "*Chinatown*". The preposition "*to*" can be identified as a constituent boundary. Therefore, in parsing "*goes to Chinatown*", we use the pattern "*X to Y*", which has two variables *X* and *Y* and a constituent boundary "*to*".

The expression "*the bus goes*" can be divided into two constituents "*the bus*" and "*goes*". However, there is no functional surface word that divides the expression into two constituents. In such cases, we employ part-of-speech bigrams as boundary markers. "*bus*" and "*goes*" are a noun and a verb, respectively. Thus the marker *noun-verb* can be inserted as a boundary marker into the input "*the bus goes*", giving "*The bus noun-verb goes*". This sequence will now match the general transfer knowledge pattern "*X noun-verb Y*".

Of the possible bigrams in the above part-of-speech sequence, only "*noun-verb*" is an eligible constituent boundary marker (Furuse, 1994b). This marker is inserted into the above sentence:

The bus noun-verb goes to Chinatown at ten a.m.

Indices to possible patterns are obtained from several words and bigrams in the above marker-inserted string (Table 1).

Table 1: Retrieved patterns

word	retrieved pattern (linguistic level)
the	*the X* (compound noun)
noun-verb	*X noun-verb Y* (simple sentence)
to	*X to Y* (verb phrase, noun phrase)
at	*X at Y* (verb phrase, noun phrase)
a.m.	*X a.m.* (compound noun)

The procedure explained so far is the part that the top-down and bottom-up pattern application methods have in common.

4 Incremental pattern application

In this section, we will show the application of constituent boundary patterns based on the concept of bottom-up chart parsing.

4.1 Linguistic level

In order to limit the combinations of patterns during pattern application, we distinguish pattern levels and for each linguistic level, we specify the linguistic sublevels which are permitted to be used in the assigned variables.

Table 2 shows examples of the relationships between linguistic levels. A variable on a given level

is instantiated by a string on the lingustic levels in the second column of Table 2. For instance, in the noun phrase "*X of Y*", the variables *X* and *Y* cannot be instantiated by a simple sentence, but can be instatiated by a noun phrase, a compound noun, and so on.

Table 2: Possible linguistic sublevels in variables

linguistic level	sublevels of variables
simple sentence	VP, NP, ...
verb phrase (VP)	VP, NP, verb, ...
noun phrase (NP)	NP, CN, proper-noun, ...
compound noun (CN)	CN, noun, ...

According to the regulation of the linguistic levels' relations shown in Table 2, a marker-inserted string is parsed using the constituent boundary patterns.

4.2 Active and passive arcs

A chart parsing method (Kay, 1980) can avoid repeatedly recomputing partial results and achieve incremental processing by using a bottom-up and left-to-right strategy. In chart parsing, an input string is parsed by combining active and passive arcs. These can be assigned to a substring of an input string when a pattern is applied to it. If all the variables of the applied pattern are instantiated or a substring can be matched to a pattern whose variables are all instantiated, a passive arc is created for the substring. When a substring can be matched to the left part of a pattern and the right variables of the pattern are not instatiated, an active arc is created for the substring.

In conventional chart parsing, many arcs can be created because every word can create active and passive arcs based on its part-of-speech. Also, many arcs can be chained via non-terminal symbols such as a part-of-speech and NP (noun phrase). For instance, the pronoun, "*I*" can create many active arcs relevant to the rules "Pronoun $\rightarrow$ *I*", "NP $\rightarrow$ Pronoun" and "S $\rightarrow$ NP VP", which can be chained. Therefore, a lot of computation is required in conventional chart parsing.

In contrast, chart parsing with constituent boundary patterns can constrain the number of arc creations because only an constituent boundary creates active arcs while a variable (e.g. *X*) never creates an arc. We obtain indices to patterns from each word of the sentence. With these indices, patterns are retrieved and checked to determine whether each of them can create an arc.

4.3 Pattern application algorithm

Our algorithm for bottom-up application of patterns is as follows. If the whole input string can be covered with a passive arc, the parsing will succeed and the derivation of the passive arc will be the parsed result.

1. If the processed string is a content word (e.g. noun, verb) create a passive arc.
2. If the processed string is a constituent boundary "α", create each kind of arc as follows, according to the pattern[1] retrieved from the constituent boundary.

2a. If the retrieved pattern is of the type "$X\ \alpha\ Y$" and a left-neighboring passive arc can satisfy the condition for *X*'s instantiation, create an active arc for "$X\ \alpha\ Y$", in which *Y* has not yet been instantiated.

2b. If the retrieved pattern is of the type "$X\ \alpha$" and a left-neighboring passive arc can satisfy the condition for *X*'s instantiation, create a passive arc for "$X\ \alpha$".

2c. If the retrieved pattern is of the type "$\alpha\ X$", create an active arc for "$\alpha\ X$".

3. If the created passive arc satisfies the leftmost part of an uninstantiated variable in the pattern of neighboring active arcs, the variable is instantiated with the passive arc, and a new passive or active arc is created. If a passive arc is generated in this operation, repeat the procedure until a new arc can no longer be created.

Figure 1 shows how an input string is parsed using our bottom-up chart method. A solid line denotes a passive arc that covers a substring of the input below, while a dotted line denotes an active arc.

The content words "*bus*", "*goes*", "*Chinatown*" and "*ten*" create passive arcs. The functional word "*the*", which is relevant to the pattern "$\alpha\ X$", creates an active arc. The assignment of the functional word "*a.m.*" to the pattern "$X\ \alpha$" creates a passive arc by combining another passive arc. The boundary markers "*noun-verb*", "*to*" and "*at*", which are relevant to the pattern "$X\ \alpha\ Y$", create active arcs by combining left-neighboring passive arcs.

First "*the*" creates the active arc (1) relevant to the pattern "*the X*". "*bus*" creates the passive arc (2). The passive arc (3) is created by combining (1) and (2). "*noun-verb*" creates the active arc (4), whereby the variable *X* of "*X noun-verb Y*" is matched against (3). "*bus*" creates the passive arc (5), and the passive arc (6) is created by combining (4) and (5). "*to*" creates the active arc (7), whereby the variable *X* of "*X to Y*" at verb phrase is matched against (5).

[1]There are other types of patterns, such as "$X\ \alpha\ Y\ \beta\ Z$", where α and β are constituent boundaries. They can be easily processed by slightly extending the algorithm.

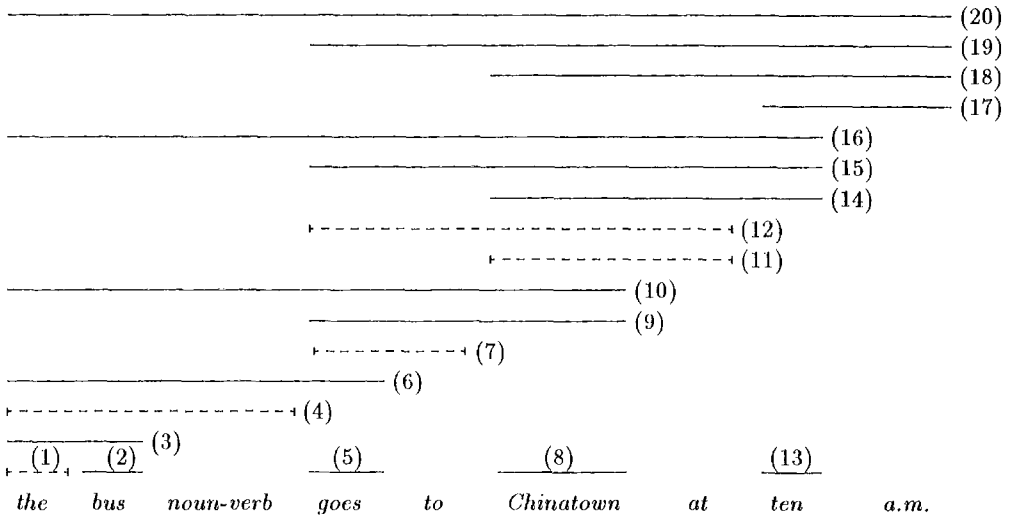

Figure 1: Chart diagram

We continue the procedure incrementally. When the rightmost word has been processed, the derivation of the passive arc of the whole input gives the parsed result, in our example the derived process of the passive arc (20), which is the combination of (4) and (19).

5 Preference of substructure

The passive arc (19), which is relevant to "*goes to Chinatown at ten a.m.*", has two competing results. One is the combination of (7) and (18), where "*X at Y*" is a noun phrase. The other is the combination of (12) and (17), where "*X at Y*" is a verb phrase. Thus, (19) has two possible structures by the application of "*X at Y*". "*X to Y*" at the verb phrase level and "*X a.m.*" at the compound noun level are also applied.

The technique for obtaining substructure preference is the determination of the best substructure when a relative passive arc is created. Only the best substructure can be retained and combined with other arcs.

5.1 Semantic distance

The most appropriate structure is selected by computing the total sum of all possible combinations of partial semantic distance values. The structure with the least total distance is judged most consistent with empirical knowledge and is chosen as the most plausible structure.

The semantic distance between words is calculated according to the relationship of the positions of words' semantic attributes in the thesaurus. The distance between expressions is the sum of the distance between the words comprising the expressions, multiplied by some weights (Sumita, 1992).

5.2 Head word information

The head words within variable bindings serve as input for distance calculations. An input for distance calculation consists of head words in variable parts. The head part is designated in each pattern. Table 3 shows the head parts of the possible substructures for "*goes to Chinatown at ten a.m.*", which corresponds to the passive arc (19).

Table 3: Head words for (19)'s substructures

passive arc	matched pattern	designated head	head word
(9),(19)	*X to Y*	X	*goes*
(17)	*X a.m.*	*a.m.*	*a.m.*
(18)	*X at Y*	X	*Chinatown*
(19)	*X at Y*	X	*goes*

In "*X at Y*" for the substring "*goes to Chinatown at ten a.m*" combined with (12) and (17), the variables *X* and *Y* are substituted for the compound expressions "*goes to Chinatown*" and "*ten a.m.*", respectively. Thus, in "*X at Y*" for the structure in (19), the input for distance calculation is "*goes*" for "*X*" and "*a.m.*" for "*Y*". Since the head of "*X at Y*" is designated as "*X*", "*goes*" becomes the head word for (19). This information is used when (19) is combined with another substring.

5.3 Structure selection

The difference in total distance value between the two possible structures is due only to the distance value of "*X at Y*". Table 4 shows the results of the distance calculation in "*X at Y*" for the combination of (7) and (18), and for that of (12) and (17). (*goes*, *a.m.*) expresses the bindings for variables *X*

and Y, where X ="*goes*", and Y ="*a.m.*". "X'" is the target expression corresponding to "X".

Table 4: Distance calculation in "X *at* Y"

	(7)+(18)	(12)+(17)
level	noun phrase	verb phrase
input	(*Chinatown, a.m.*)	(*goes, a.m.*)
closest example	(*morning, a.m.*)	(*depart, a.m.*)
target	Y' *no* X'	Y' *ni* X'
distance	0.50	0.21

According to the distance calculation in the combination of (7) and (18), "Y' *no* X'", with the distance value 0.50, is selected as a target expression. In the combination of (12) and (17), "Y' *ni* X'" with the distance value 0.21 is selected as a target expression. Thus, the combination of (12) and (17) is selected as the structure of the passive arc (19). Based on the results of distance calculations, other partial source patterns for (19), "X *to* Y" and "X *a.m*", are transferred to "Y' *ni* X'" with the distance value 0.12, and "*gozen* X' *ji*" with the distance value 0.00. Thus, the passive arc (19) has its source and target structure through the combination of (12) and (17), the total distance value 0.33, and the head word "*goes*".

Then, the structure of the whole input string, which corresponds to (20), is constructed by combining (19) with (4). In this combination, "X *noun-verb* Y" matches the input string and is transferred to "X' *wa* Y'" based on the result of distance calulation. From the combined structure for (20), the sentence below is generated after adjustment necessary for Japanese grammar. The words "*bus*", "*goes*", and "*Chinatown*" are transferred to "*basu*", "*iku*", and "*Chainataun*"[2], respectively.

Basu wa gozen 10 ji ni Chainataun ni iki masu

"*iki*" is the conjugated form of "*iku*" followed by *masu*, a polite sentential-final form.

6 Preliminary Experiment

In this section, we perform English-to-Japanese translation to compare the efficiency of the top-down pattern application with that of our new method, based on the bottom-up application and substructure preference in the TDMT prototype system.

6.1 TDMT prototype system

The TDMT prototype system, whose domain is travel conversations, is designed to achieve multi-lingual spoken-language translation (Furuse, 1995). While language-oriented modules, such as morphological analysis and generation, are provided to treat multi-lingual translation, the transfer module, which is a central component, is a common part of the translation system for every language pair. The system is written in LISP and runs on a UNIX machine. Presently, the prototype system can translate bilingually between Japanese and English and between Japanese and Korean. In English-to-Japanese translation, the present vocabulary size is about 3,000 words [3] and the number of training sentences is about 2,000.

6.2 Experimental results

We have compared translation times in the TDMT prototype system for two cases. One case utilizes top-down application; the other case utilizes the new application method presented in this paper, which adopts bottom-up pattern application and retains only one substructure using semantic distance calculation. The translation times are measured using a Sparc10 workstation.

We have experimented with the translation times of some English sentences into Japanese. The following sentences cause only minor structural ambiguity. Note that a comma is not used in the input sentence, because it is assumed to be a spoken-language input such as the output of speech recognition.

(1) *I have a reservation for tomorrow.*

(2) *Will my laundry be ready by tomorrow?*

(3) *You can walk there in about three minutes.*

(4) *Then may I have your credit card number please?*

Table 5 shows the translation time of the above sentences. For these translations, not much difference could be seen between the new bottom-up method and the top-down method. For such inputs TDMT can quickly produce the same translation results with either method.

Table 5: Translation time for short sentences

input sentence	# of structures	translation time (sec)	
		top-down	new method
(1)	2	0.18	0.17
(2)	4	0.17	0.20
(3)	4	0.38	0.35
(4)	11	0.85	0.70

The following sentences cause much structural ambiguity because of PP-attachment, relative clauses, conjunctions, etc.

[2]The prototype system assigns a default target expression to a surface source expression. Another target expression is selected when a specific example in the transfer knowledge is closest to the input.

[3]In the Japanese-to-English translation system, the present vocabulary size is about 5,000 words.

(5) *This sales clerk doesn't understand anything I say and I'm wondering if you would help me explain what I want.*

(6) *Could I please have your name the date of arrival and the number of persons in your party?*

(7) *Tell someone at the front desk what game you want to see and what type of seat you want and they'll get the tickets for you.*

(8) *I left some laundry to be cleaned but I can't remember where the cleaners is and I was wondering if you could help me.*

Table 6 shows the translation time of the above sentences. In the above translations the same translation results could again be obtained for both methods. However the new method can achieve a far more efficient translation than the top-down method.

Table 6: Translation time for long sentences

input sentence	# of structures	translation time (sec)	
		top-down	new method
(5)	312	6.12	1.22
(6)	442	4.03	0.92
(7)	544	13.22	2.37
(8)	696	12.10	2.17

Average translation times in the top-down method were 1.15 seconds for a 10-word input and 10.87 seconds for a 20-word input. Average translation times in the bottom-up method were 0.55 seconds for a 10-word input and 2.04 seconds for a 20-word input. The translation time in the top-down method is considered to be closely related to the number of possible structures, while the translation time in our new method is not directly reflected by this number. The increase in the number of substructures retained with the new method is much smaller than that of the number of possible structures in the top-down method. Therefore, our new method can efficiently translate a longer input string having many competing structures.

Also, we have performed a small translation-quality experiment on the two pattern application methods with the 95 untrained sentences within the system's vocabulary. Both the top-down method and the proposed bottom-up method gave the correct translation for the same 60 sentences with a success rate of 63.2%. For only two sentences, different structures were produced by the two methods; however, all of them were incorrect translations. This experimental result shows that our new translation strategy maintains translation quality.

Similar results, which show the usefulness of the new TDMT for spoken-language translation, were obtained in other types of translation such as Japanese-to-English (or, -Korean) translation.

7 Conclusion

We have proposed an incremental translation method in Transfer-Driven Machine Translation (TDMT). In this method, constituent boundary patterns are applied to an input in a bottom-up and left-to-right fashion. Additionally, by dealing with best-only substructures, the explosion of structural ambiguity is constrained and efficient translation of a lengthy input can be achieved. Through preliminary experimentation, our new TDMT has been shown to be efficient and particularly promising for spoken-language translation.

One important future research goal is the incorporation of incremental morphological analysis and generation into the proposed translation strategy, which would provide a simultaneous interpretation mechanism for application to a practical spoken-language translation system. Also important is the introduction of a repair mechanism to correct the best-first results.

References

W. Finkler and A. Schauder 1992. Effects of Incremental Output on Incremental Natural Language Generation. In *10th European Conference on Artificial Intelligence*, pages 505–507, Vienna, Austria.

O. Furuse, E. Sumita, and H. Iida. 1994a. Transfer-Driven Machine Translation Utilizing Empirical Knowledge (in Japanese). *Transactions of Information Processing Society of Japan*, Vol. 35, No. 3, pages 414–425.

O. Furuse, and H. Iida. 1994b. Constituent Boundary Parsing for Example-Based Machine Translation. In *Proc. of Coling '94*, pages 105–111.

O. Furuse, J. Kawai, H. Iida, S. Akamine, and D.B. Kim. 1995. Multi-lingual Spoken-Language Translation Utilizing Translation Examples. In *Proc. of NLPRS'95*, pages 544–549.

M. Kay. 1980. Algorithm Schemata and Data Structures in Syntactic Processing. *Technical Report CSL-80-12*, XEROX Palo Alto Research Center.

G. Kempen and E. Hoenkamp. 1987. An Incremental Procedural Grammar for Sentence Formulation. *Cognitive Science*, 2(11): pages 201–258.

H. Kitano. 1994. The ΦDMDIALOG System. In *Speech-To-Speech Translation*, H. Kitano, Kluwer Academic Publishers, pages 47–113.

E. Sumita and H. Iida. 1992. Example-Based Transfer of Japanese Adnominal Particles into English. *IEICE Transactions on Information and Systems*, E75-D, No.4, pages 585–594.