Practical Machine Translation System allowing Complex Patterns

Mihoko Kitamura and Toshiki Murata

Oki Electric Industry Co., Ltd.
Osaka, Japan
{kitamura655, murata656}@oki.com

Abstract

Pattern-based machine translation systems can be easily customized by adding new patterns. To gain full profits from this character, input of patterns should be both expressive and simple to understand. The pattern-based machine translation system we have developed simplifies the handling of features in patterns by allowing sharing constraints between non-terminal symbols, and implementing an automated scheme of feature inheritance between syntactic classes. To avoid conflicts inherent to the pattern-based approach the system has priority control between patterns and between dictionaries. This approach proved its scalability in the web-based collaborative translation environment '*Yakushite Net.*'

1 Introduction

There have been many attempts at using translation examples to improve the quality of translation systems. One practicable approach would be to combine a machine translation system with a translation memory(ATLAS,02). Sentences already present in the translation memory can be translated directly, while sentences not yet translated have to go through machine translation, and can be added to the translation memory after post editing. This combination improves the translation system as it enables correct translation results to be reused. However, as accumulated translation examples are used only literally, they don't affect the quality of machine translation itself.

In order to use examples to improve the translation process itself, we developed a pattern-based machine translation system that utilizes translation patterns created by decomposing translation examples.

Pattern-based translation systems execute the parsing, transferring and generating processes by using only translation patterns, all the knowledge necessary for the translation being written in patterns. This provides good readability for all this knowledge, and what is more, it is easy for users to add new translation patterns. However, in previous pattern-based systems, writing new patterns was difficult due to the lack of flexibility in the way to describe constraints on the features associated with a non-terminal, requiring for instance a new non-terminal for each semantic condition, so that a deep understanding of the internals of the system was necessary in order to add new patterns (Takeda,96).

HPSG parsers(Oepen,00) are stronger, their use of feature unification allowing for maximal flexibility, but they require deeper grammatical knowledge for the development of grammatical rule. The integration of syntactical and semantical information requires the use of many features making the task of writing patterns too difficult for a non-specialist. This is a major drawback as we hope that users add various patterns.

We have built a pattern-based machine translation system with good readability by writing all the conditions, including semantics, gender and number of non-terminals and words as a combination of features, and making it possible to match, share and inherit features, but without full feature unification. Moreover, this system solves the problem of large computation times, by implementing feature inheritance through copying rather than unification, and by drastically reducing

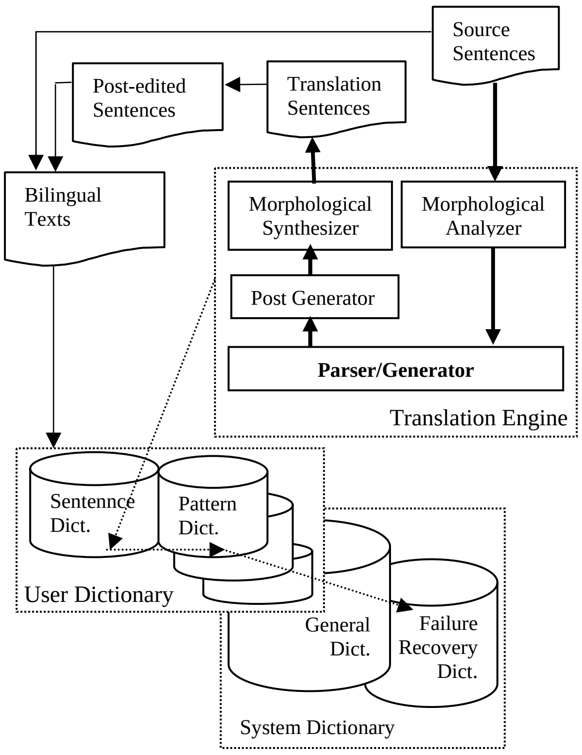

Figure 1: The Architecture of Pattern-Based MT system

the number of candidates through the pruning of the features kept on each non-terminal symbol.

Rich expressiveness enables the user to enter accurate patterns, reducing potential conflicts with other patterns. The user need not know the details of how the pattern will be processed during translation. It is also possible to enter translation patterns acquired by statistical methods directly.

The system provides also priority control between patterns and between dictionaries in order to avoid an explosion of the number of candidates and reduce side effects caused by newly introduced patterns.

Special cases where patterns cannot handle generation in the target language are processed by a post generator.

The next section shows the outline of the pattern-based translation system we have developed. Section 3 describes the implementation, the evaluation and the application. Section 4 presents the related works and section 5 concludes.

2 Pattern-Based Machine Translations

Figure 1 shows the architecture of our system. Thick arrows show the flow of the translation, thin arrows show the data flow for memorization of translation examples, and dotted lines show the sequence for referring dictionaries

First, the source sentence is analyzed morphologically, normalizing words and decorating them with morphological features. This decorated sequence of words is then passed to the parser. The sentence is parsed by using the source side of translation patterns in the appropriate user and system dictionaries, and combining them bottom-up. When the sentence has been parsed successfully, the parse tree is translated by top-down generation of the parse tree of the target language, using the target side of patterns.

Then, some features of the generated tree are handled by the Post Generator to produce refined sentences.

Lastly, the morphological synthesizer adjusts inflection and conjugation, and the translated sentence is output.

Automatic feedback of correct post-edited translations and accumulation of translation examples improve the quality of future translations.

2.1 Morphological Analyzer

Morphological analysis uses a morphological dictionary, and associates to each surface form a normalized form, together with features specifying the part of speech, agreement, surface conjugation, and case. In most cases only one normalized form will be associated with a surface form, eventually with some of its features being multi-valued (for instance, for a verb in basic form, its agreement might be all persons except 3rd singular). In the special case of homonyms, the same surface form comes from two different dictionary words, and the result of the morphological analysis contains several different candidates for an input word. This is later handled by trying all these candidates in the parsing phase.

To simplify our presentation we will omit this case, and suppose in the following that the result of the morphological analysis is a linear sequence of words decorated with (eventually multi-valued) morphological features.

```
(a) [en:VP:sSem=human    play:pos=v:*[1:NP:sem=instrument]]
     [ja:VP [1:NP] を:pos=particle 弾く:pos=v:* ];

(b) [en:VP:sSem!=human play:pos=v:* ]
     [ja:VP 鳴る:pos=v:* ];

(c) [en:VP:sSem=human play:pos=v:* [1:NP:sem=sport|game]]
     [ja:VP [1:NP] を:pos=particle する:pos=v:* ];

(d) +[en:VP:sSem=human play:pos=v:* ]
      [ja:VP 遊ぶ:pos=v:* ];

(e) [en:N piano:pos=n:sem=instrument:*]
     [ja:N ピアノ:pos=n:* ];

(f)  [en:N tennis:pos=n:sem=sport:*]
     [ja:N テニス:pos=n:* ];

(g) [en:N Ken:pos=n:sem=human:*]
     [ja:N 健:pos=n:* ];

(h) [en :Adv never :pos=adv :*]
     [ja: Fs 決して:pos=adv:*:postGen=neg];

 (i) [en:SentenceSub when:pos=conj [1:Sentence:*]]
     [ja:SentenceSub [1:Sentence:sentenceType=sub:*] 時:pos=conj];

(j) [en:NP [1:N:*]]
     [ja:NP [1:N:*]];

(k) [en:S [1:Sentence:*] ]
     [ja:S [1:Sentence:sentenceType=main:*] ];

(l) [en:Sentence    [1:NP:sem={SEM}:personNum={NUM}] [2:VP:sSem={SEM}:personNum={NUM}:*]]
    [ja:Sentence:sentenceType=main [1:NP] は:pos=particle [2:VP:*]];

(m) - [en:Sentence [1:NP:personNum={NUM}] [2:VP:personNum={NUM}:*]]
       [ja:Sentence:sentenceType=main [1:NP] は:pos=particle [2:VP:*]];

(n) [en:Sentence    [1:NP:sem={SEM}:personNum={NUM}] [2:VP:sSem={SEM}:personNum={NUM}:*]]
    [ja:Sentence:sentenceType=sub [1:NP] が:pos=particle [2:VP:*]];
```

Figure 2 : Examples of Translation Patterns

2.2 Parser and Generator

Figure 2 shows examples of translation patterns used in English to Japanese translation[1]. Examples (a)-(i) in Figure 2 are vocabulary patterns, (j)-(n) are grammatical patterns. In rule-based translation systems, vocabulary patterns would correspond to dictionaries, and grammatical patterns to grammar rules. As Figure 2 shows, patterns allow writing grammar rules and dictionaries in a united form without any specific distinction. All patterns are entered together in the system dictionary.

One can understand pattern (a) as the following CFG rules.

English:	VP -> play NP
Japanese:	VP -> NP を*(wo)* 弾く*(hiku)*

A pattern starts with the name of the language, and category and features on the left-hand side of the CFG rule (the parent node in the parse tree), followed by descriptions of non-bracketed words and bracketed non-terminals on the right-hand side of the CFG rule, in their textual order. ':' is a separator between features of a pattern element, and space a separator between pattern elements.

Patterns come in pairs: one pattern for each language. The mandatory numerical index in non-terminals allows relating non-terminals elements between source and target patterns.

Analysis uses source language patterns, marked by 'en' here. By applying patterns bottom-up, one can reduce word sequences to the corresponding left-hand side, and eventually reach the 'S' non-terminal (the root of the parse tree).

Once the source parse tree has been completed, it is sufficient to convert each node using the corresponding target language pattern, marked by 'ja'. Since there is a one-to-one relation between

[1] Real patterns contain more features, but we omitted here features that are not required by our examples.

non-terminals in the source and target patterns, generation of the target parse tree is carried out immediately.

Translation patterns can specify one or more features for both terminal and non-terminal symbols, such as *‘pos=verb’* (the part of speech is verb), *‘personNum=3sg’* (third person and singular), *‘sem=human’* (the semantics is human). They can allow one or more values for one feature and also can specify negative information as in *‘pos!=verb’* (the part of speech is not verb).

The features in the right-hand side of the source language patterns express conditions, either by requiring a specific value for a feature, or expressing a sharing constraint between two features, through unification variables (in curly brackets, like ‘{SEM}’ or ‘{NUM}’). Matching succeeds if all these conditions are satisfied. Corresponding words in the input sequence are then replaced by the non-terminal on the left-hand side, while the corresponding parse tree is built.

In order to ease the propagation of features inside the parse tree, one of the right-hand side pattern elements is designated as *head*, and marked by a “*”. Its features are inherited by the left-hand side non-terminal, except for those already defined in the left-hand side, which are ignored. Features on the left-hand side of source-language patterns, together with inherited features, appear in the newly replaced non-terminal, and they will be matched later by the right-hand side of other patterns.

Word selection in the target language is realized by checking features. In simple cases, the condition is directly applied to a symbol in the pattern. For instance, in patterns (a) and (c), “play” is associated with different semantic values according to whether its object is a music instrument, or either a sport or game; then it is translated into proper words in these different situations: “play the piano” gives “ピアノ(*piano*)を(*wo*)弾く(***hiku***)”, but “play tennis” gives “*テニス(tenisu)を(wo)*する***(suru)***”.

More complex cases, like the difference between “a piano plays” and “Ken plays”, use sharing constraints and feature inheritance. Here the semantic features ‘instrument’ and ‘human’ are inherited from both name patterns and verb phrases, and they are checked in the sentence construction pattern (l). Only agreeing subject and verb will be accepted, enabling the system to provide the proper translations “ピアノ*(piano)*は*(ha)*鳴る***(naru)***” and “健*(ken)*は*(ha)*遊ぶ***(asobu)***”.

```
Sentence = {   sentenceType   };
VP       = {  personNum
              conjugation
              subjSem         };
NP       = {  personNum
              sem             };
```

Figure 3 : The example of Feature Definition Table

In (l), (m) and (n), sharing constraints are also a concise way of uniting person and number information.

In target language patterns, propagation works the other way round: features on the left-hand side of the target pattern act as constraints for the generation process, and features on the right-hand side are propagated to child nodes. Inheritance goes from the parent node to the head node, with the same overriding mechanism for features present in both.

The matching of the target language features makes it possible to provide proper translations in different grammatical situations. For example, differences such as the one between the subordinate clause “私*(watashi)*が***(ga)***ピアノ*(piano)*を*(wo)*弾く*(hiku)*時*(toki)*” (“... when I play the piano”) in (n) and the complete sentence “私 *(watashi)*は***(ha)***ピアノ*(piano)*を*(wo)*弾く*(hiku)*” (“I play the piano”) in (l) can be translated accurately.

Lastly, two decisions were taken to avoid multiplication of candidates. One is that the set of features each non-terminal symbol can have is limited according to a feature definition table as seen in Figure 3. For instance the CFG rule for ‘S’ does not need any longer conjugation, which is one of the features of head ‘VP’. With this limitation, every non-terminal symbol has only necessary features, which simplifies parsing trees. This is effective for reducing the number of candidates, in that non-terminals symbols that have the same combination of feature values can be merged, and a disjunctive tree can be formed from the tree structure during parsing.

The other decision is that generation in the target language is not allowed to fail and backtrack: one can only choose between two patterns on the basis of target side constraints if the source side pattern is identical (i.e., the decision is local). Otherwise, failures in feature constraints are ignored, and generation goes on assuming they succeeded.

2.3 Our Approach to Search Space Control

The main problem pattern-based translation faces is that of effectively controlling the search space. If strict conditions are set for patterns, the translation is likely to end up in failure, however if patterns with very few conditions are used, too many patterns are applied, and the number of candidates increases explosively. To avoid this problem, we have introduced two priority control systems for patterns.

2.3.1 Control of Priority in a dictionary

Figure 2 (l) shows a translation pattern in which the semantics of the subject is limited so that it can respond to different situations. However, if a user is not careful enough and does not give accurate semantics information in his/her pattern, it will not be matched and the translation will fail. To protect the system from such mistakes, translation patterns without limitation of meaning are also needed. However, when the strict pattern succeeds, the unlimited one will also succeed, and the number of candidates increases combinatorially. Even worse, unless one pattern is given preference, after the parsing process the system cannot judge which result is better and cannot choose a unique plausible translation.

To avoid these situations, the system provides a way to mark a pattern as being applicable only when patterns with more detailed conditions are not matched, by putting a “-“ (minus) mark before it as in (m). This avoids the situation where both patterns are applied. Experience showed us that we needed three priority levels. So there is also “+” in (d) for higher priority patterns.

An additional criterion we use to select patterns is to choose a parse tree using a minimal number of patterns, as it will include patterns closer to the input sentence. This information is combined with the above priority of individual patterns to provide a comprehensive evaluation of parse trees.

2.3.2 Control of Priority between Dictionaries

Two problems may arise when users input a large number of patterns. One is a potential slowdown in translation speed, which is affected by the overall number of patterns. The other is that newly introduced patterns may conflict original ones and cause unstable translation behavior. We solve the two problems by developing a pruning mechanism, which would consider user patterns first, and then some dictionaries correlated with the user dictionary, and finally system dictionaries during translation. This pruning avoids an explosion of the number of candidates, and side effects caused by newly introduced user patterns are limited to this user dictionary.

2.3.3 Failure Recovery Dictionary

We have introduced the Failure Recovery Dictionary using the above pruning mechanism. Failure recovery dictionary is referred last among sub-dictionaries in the system dictionary. In other words, the failure recovery dictionary acts only when the normal parsing process using other dictionaries has failed.

The Failure Recovery Dictionary contains patterns with grammatical mistakes and patterns that help avoiding unsuccessful translation. For instance the following pattern allows the use of a subject and a verb for which agreement rules are not satisfied.

```
[en:Sentence [1:NP ] [2:VP:*]]
[ja:Sentence:setenceType=main [1:NP] は
  [2:VP:*]];
```

By default the system will work on a rigid translation that is grammatically correct, but does not consider rare phrase structures. This avoids slowing down translation of simple sentences. Whenever normal translation fails, the system tries again to translate with more patterns, which is slower but much more robust.

2.4 Post Generator

Generation using a synchronized grammar depends strongly on the structure of source language patterns, so pattern-based methods are weak at generating expressions peculiar to the target language.

Some features of the generated tree are handled by the Post Generator to produce refined sentences. To take a simple example, although the Japanese translation for the English word "never" is "決して *(kesshite) ...* ない*(nai)*", the pattern of figure 2-(h) cannot lexicalize "ない*(nai)*". Because the verb which "never" qualify cannot be identified when figure 2-(h) is applied.

The feature '*postGen=neg*' within "決して *(kesshite)*" is matched by a post generator rules which generates "ない*(nai)*" at the end of the verb phrase which includes "決して*(kesshite)*".

Figure 4 indicates an example of the rule for Post Generator. The rule means, if a word holds "postGen=neg", put "ない(nai)" backmost of VP(verb phrase) which includes the word. The rule is written in XML notation.

3 Implementation and Evaluation

3.1 Process for Development of Patterns

The number of grammatical patterns is about 2,000 and the vocabulary patterns are about 180,000. Vocabulary patterns were built based on dictionaries for a rule-based machine translation system which we had developed before.

Grammatical patterns newly was designed and developed to cover Collins' grammar (Collins,90). For each item in the grammar we made an example and then created the corresponding pattern by hand. We also created various test examples for each item in the grammar and used them to check for conflicts in subsequent patterns.

The conflict rate is about 3% when we added new grammatical patterns. But our debugger, which can indicate visually the pattern selection process and the result of applied patterns, facilitated the detection of the cause of conflicts. Furthermore when we detected the cause, we could adjust patterns easily by refinement and addition of conditions.

3.2 Implementation and Evaluation of the Translation Engine

The above English-Japanese machine translation system has been implemented in Java.

The parser uses the *Earley* algorithm. At the time of this writing, the number of non-terminal symbols

```
<Rule NAME= "postGen=neg">
<StartLeaf>
  <Feature NAME="postgen" VALUE="neg"/>
</StartLeaf>
<Scope TYPE="NEAREST">
  < Feature NAME="category" VALUE="VP"/>
</Scope>
<OriginalLeaves>
  <OriginalLeaf ID="1" DIR="LtoR">
    <Feature NAME="postgen" VALUE="neg"/>
  </OriginalLeaf>
  <OriginalLeaf ID="2" DIR="RtoL">
    < Feature NAME="pos" VALUE="v"/>
  </OriginalLeaf>
</OriginalLeaves>
<EditedLeaves>
  <EditedLeaf ID="1" COPYFROM="1"/>
  <EditedLeaf ID="2" COPYFROM="2"/>
  <EditedLeaf ID="2" DELTA="1">
    < Feature NAME="pos" VALUE="aux"/>
    < Feature NAME="baseForm" VALUE="ない"/>
  </EditedLeaf>
</EditedLeaves>
</Rule>
```

Figure 4 : Example of the Rule for Post Generator

is about 80, and about 60 types of features are defined.

Most of the vocabulary patterns are managed in databases. The databases are converted into pattern format for entry into the dictionary.

The number of rules for the post generator is about 280.

Using only system dictionaries, we evaluated the translation quality using the JEIDA English-Japanese translation evaluation set (JEIDA,95), which is composed of 770 bilingual sentences. The failure recovery dictionary was referred by 9 sentences and the number of sentences that failed to parse is 10. The percentage of translations that were judged correct by professional translators was about 94 percent.

Moreover, the speed is acceptable and the translation time is roughly proportional to the length of sentences. Figure 5 shows processing time per sentence on a Pentium III machine at 933MHz. Translation times are noticeably slower

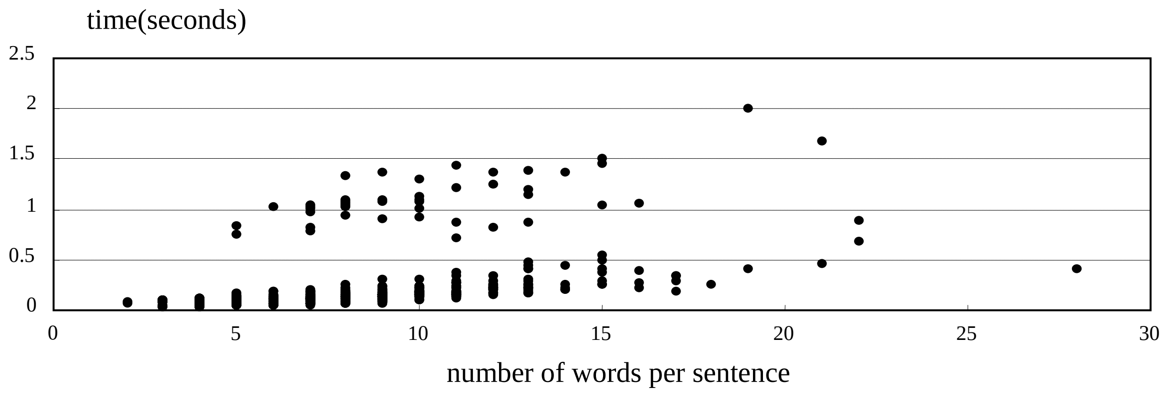

Figure 5 : Translation Speed

when a sentence contains several structurally ambiguous constructions, such as coordination.

3.3 Application to Collaborative Translation Environment "Yakushite Net" on Internet

Pattern-based translation systems get better as many users from various backgrounds use them, and enter lacking patterns, particularly technical words and idioms, which have an immediate impact on translation quality. For this purpose, we applied our system to the Collaborative Translation Environment 'Yakushite Net' on Internet (Shimohata,01) (Murata,03).
. The environment has '*community*' dictionary, which the user selects according to his/her needs and can be improved by contributions from members of the community, distributed over the Internet.

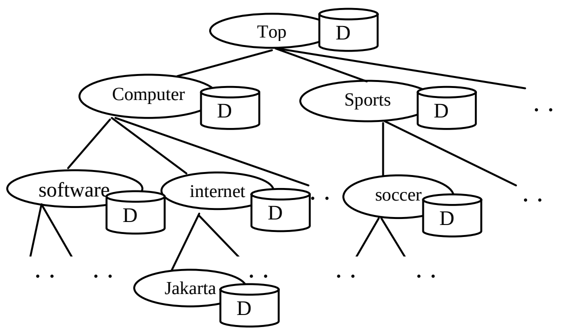

Figure 6 : Tree Structure of Community Dictionaries

The collaborative translation environment has a lot of communities, with their community dictionaries structured in a hierarchical directory, shown in figure 6.

When translating in a certain community environment, the translation engine refers first the community's own dictionary, and subsequently to dictionaries with ordering priority from the nearest parent community to the top. These community dictionaries except the top dictionary correspond to user dictionaries in figure 1, which are referred stepwise. The top dictionary is domain-independent, and corresponds to the system dictionary in figure 1.

We see the construction of well-targeted domain specific dictionaries and their use according to the context as the best solution to avoid unwieldy addition of user patterns.

4 Related Works

Now, we compare the translation patterns used in rule-based machine translation system and with those of our pattern-based machine translation system.

ALT-J/E(Hayashi,01), is a transfer-based machine translation system employing 'transfer patterns' as verbal word selection rules. Transfer patterns are similar to our patterns, as below

ex1: N1(subject) が N2(permission) を 取る
=>N1 take N2
ex2: N1(subject) が N2(hotel) を 取る
=>N1 reserve N2

Transfer-based machine translation applies the patterns after the parsing completes and transfers the structure from source language to target language. Consequently it allows only particular patterns that have explicit parsing result, and cannot describe patterns as freely as our method.

TDMT(Kashioka,99) could be described as pattern-based, it is however limited in a number of ways. First, each pattern, called 'transfer knowledge,' must contain constituent boundary either a functional word or a special part-of-speech bigram marker, inserted by the morphological analyzer. Then, pattern features are very limited, allowing for semi-automatic acquisition, but precluding efficient generalization.

These limitations mean that some complex phrase structures cannot be analyzed, and that even simple patterns must be given in lots of instances to overcome the absence of generalization.

5 Conclusion

The machine translation system we have developed has two major advantages.

(1) The system is pattern-based, but it is possible to share constraints and inherit features between non-terminal symbols, simplifying input of patterns.

(2) The system has two priority control systems. One is the priority control among patterns in a dictionary. The other is the priority order between dictionaries using a pruning algorithm. The dictionary with the least priority is the failure recovery dictionary.

This machine translation system is already available to users on Internet as the collaborative translation environment 'Yakusite Net.'

We have two future plans: adding the capability to extract translation patterns from bilingual corpora and supporting multilingual translation.

A translation pattern extraction tool, able to automatically extract translation patterns containing non-terminal symbols with appropriate constraints from translation examples(Kitamura,96), would help users to produce translation patterns.

Our interest in multilingual translation stems from the language independence of our parser and generator. It shall be possible to build a translation system for a new language using just decomposed translation examples as pattern dictionary, without deep knowledge of the language itself. In such a context, the above translation pattern extraction tool would allow to extract patterns from translation examples of the new language, and ideally to build a system from examples alone.

Acknowledgment

This research is supported by a grant from Telecommunications Advancement Organization of Japan (TAO).

6 Bibliographical References

ATLAS, http:// software.fujitsu.com/en/atlas/

Takeda, K. 1996. "Pattern-Based Context- Free Grammars for Machine Translation". In proceedings of *the 34th Annual Meeting of the Association for Computational Linguistics*, pp.144-151.

Oepen, S., Flickinger, D., Uszkoreit, H., and Tsujii, J. 2000. "Introduction to this Special Issue", Natural Language Engineering, 6(1):1-14, pp1-12

Collins, Collins Cobuild English Grammar. 1990. COBUILD. London

JEIDA, JEIDA (the Japan Electronic Industry Development Association). 1995. "Evaluation Standards for Machine Translation Systems (in Japanese)". 95-COMP-17. Tokyo.

Shimohata, S., Kitamura, M., Sukehiro T., and Murata, T. 2001. "Collaborative Translation Environment on the Web". In proceedings of *the MT Summit VIII*, pp331-334.

Murata, T., Kitamura, M., and Tatsuya, S. 2003. "Implementation of Collaborative Translation Environment: Yakushite Net", In proceedings of *the MT Summit IX*.

Hayashi, M., Yamada, S., Kataoka, A., and Yokoo, A. 2001, "ALT-J/C A Prototype Japanese-to-Chinese Automatic Language Translation System", In] proceedings of *the MT Summit VIII*, pp157-161.

Kashioka, H., and Ohta, H. 1999. "Applying TDMT to Abstracts on Science and Techology", In proceedings of *the MT Summit VII*, pp213-219.

Kitamura, M., and Matsumoto, Y. 1996. "Automatic Extraction of Word Sequence Correspondences in Parallel Corpora". In proceedings of *the 4th Annual Workshop on Very Large Corpora*, pp79-87.