Learning Mechanism in Machine Translation System "PIVOT"

Mitsugu Miura Mikito Hirata Nami Hoshino
C&C Systems Engineering Division
NEC Corporation
4-12-35, Shibaura, Minato-ku,
Tokyo, JAPAN

Abstract

NEC's machine translation system "PIVOT" provides analysis editing functions. The user can interactively correct errors in analysis results, such as dependency and case. However, without a learning mechanism, the user must correct similar dependency errors several times. We discuss the learning mechanism to utilize dependency and case information specified by the user. We compare four types of matching methods by simulation and show non-restricted best matching is the most effective.

1. Introduction

In the current machine translation system, users cannot always get correct translated sentences at the first translation. This is due to the low ability of the grammar rules and low quality of the dictionary. Moreover, the grammar rules and the dictionary need customization for each document of varying fields and contents. It is very difficult to prepare beforehand the information corresponding to various fields.

NEC has developed a machine translation system "PIVOT"(Japanese to English/English to Japanese) as the translation support system for business use. The translation part of PIVOT is the rule-based system and adopts the interlingua method. PIVOT provides a special editor so that the user can correct the analysis results. The user can interactively select suitable translation equivalents, can correct dependency, case (semantic relation), and so on. In technical manual documents which are the main objects of machine translation, there are many expressions that appear more than once. The analysis results of such expressions are often the same. At present, PIVOT has learning function for selection of translation equivalents, but it does not have such mechanism for dependency and case. The user has to correct many similar errors in dependency and case, so a heavy burden is laid on the user. Information given by the user can be regarded as customizing information for the document to be translated. Therefore, for a practical use system, it is an important issue to provide a framework to improve translation by using correction information from the user.

There are various approaches for analyzing sentences by using accumulated dependencies. One system automatically extracts all dependencies which have no ambiguity[5]. Another system accumulates only the dependencies which are directly corrected by the user [2]. In Miura et al.[4], the system accumulates all dependencies in the sentence that are corrected or confirmed by the user.

There are two ways for remembering the keys in the dependency structures to be accumulated: one by the spelling and the other by the semantic code. However, the rough semantic code used in the current system does not have high distinguishing ability, and often causes bad influence. For example, consider the following sentences.

a. 彼はオペラグラスで歌っている男を見た。
   He looked at the singing man with opera glasses.
b. 彼はマイクで歌っている男を見た。
   He looked at the man who is singing with the microphone.

The semantic code "Instrument" is usually assigned to "オペラグラス(opera glasses)" and "マイク(microphone)". Therefore, it isn't possible to fix dependency relation such as "歌う(singing)" with "マイク(microphone)", and "見る(look)" with "オペラグラス(opera glasses)".

In the process of using learning results there is an approach that adopts best matching by computing similarity with accumulated information[4]. The example-based approach that translates by retrieving examples and calculating similarity has been investigated. These systems also adopt best matching[1][6][7].

This paper proposes an approach that can improve the translation quality by interactively accumulating dependency and case structures corrected by the user. In the learning process, the syntactic head, the syntactic dependent, and the case between them are stored in the association database. To avoid side effects, head and dependent words are stored in the form of spellings. This makes it easier for the user to understand the behavior of the system. Four types of matching methods are examined that are used in matching between the possible analysis structures and the association database.

Section 2 describes analysis editing function in PIVOT/JE(Japanese to English). Section 3 explains the learning mechanism, and the results of simulation on actual manuals are presented in Section 4.

2. Analysis Editing Function

The user can interactively specify the following information related to dependency relation by using analysis editing function of PIVOT/JE.

(1) Dependency (syntactic dependent and syntactic head)
(2) Case
(3) Parallel
(4) Scope

(5) Sharing

The dependency relation which the system analyzes is displayed on the screen as shown in Figure 1. An underline is drawn under each Japanese phrase (a word with a particle). The dependency is shown by the line which connects two phrases. The thick line indicates the dependency corrected by the user. Case is displayed on the line of the dependency in the form of the particles which have one-to-one correspondence with one of the cases. The box indicates the correct case specified by the user. The user directly corrects above-mentioned information by using a mouse and carries out translation operation once again. The translation rule controls the analysis to reflect the correction by the user.

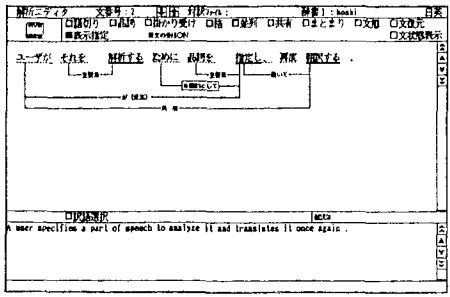


Figure 1: Display of Analysis Result

2.1 Dependency

The user can correct dependency. In Figure 2, syntactic head of "ユーザが(user)" is changed from "解析する(analyze)" to "指定する(specify)".

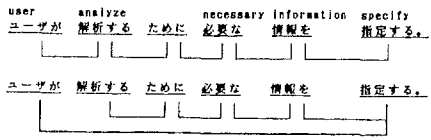


Figure 2: Example of Dependency Correction

2.2 Case

Case shows the semantic relation between two phrases which are in dependency relation. PIVOT has more than forty kinds of cases such as Agent and Reason. On the screen, particles are used to express cases.

In Figure 3, the case between "EWS4800" and "動作する(run)" is changed from "Contents" to "Place".

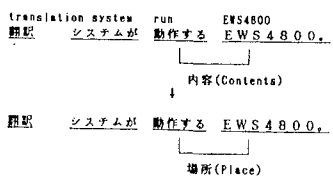


Figure 3: Example of Case Correction

2.3 Parallel

The user can specify the information that two phrases are in parallel relation. Because parallel relation is one of the PIVOT cases, this function enables the user to correct dependency and case at the same time.

2.4 Scope

The user can specify scope. Scope means the phrase sequence in which only the syntactic head has dependency relation with other phrases outside of it.

2.5 Sharing

In Figure 1, "ユーザ(user)" is the subject of "指定(specify)" and at the same time it is the subject of "翻訳する(translate)". In such a case, we say "user" is shared by "指定 (specify)" and "翻訳する(translate)". Specification of sharing is done by specifying more than one syntactic heads for the dependent. So the sharing is decomposed into dependency relations.

Useful information on dependency relation is gotten from the user's specification of scope and so on, but this paper discusses learning from correction operation for dependency and case only.

3. Learning Mechanism

Proposed learning mechanism is as follows.

3.1 Learning Process

(1) PIVOT analyzes a source sentence.
(2) PIVOT displays the analysis result.
(3) A user corrects mistakes in the analysis result.
(4) After the user finishes making corrections, PIVOT translates the sentence again.
(5) PIVOT asks the user whether translation has been a success or not.
(6) If the translation is a success, PIVOT stores the analysis result together with the instruction item into an association database. If the translation is a failure, PIVOT does nothing further.

3.2 Applying Process

(1) PIVOT analyzes a source sentence.
(2) If there is ambiguity at a certain stage of analysis, PIVOT retrieves data in the association database.
(3) PIVOT compares the possible analysis structures of the given sentence with the analysis results accumulated in the association database.
(4) PIVOT selects the analysis structure that matches with the analysis results accumulated in the association database. If no matching occurs, PIVOT selects one structure by further application of the analysis rules.

PIVOT learns correct analysis structures related to user's instruction. The smallest unit of PIVOT's analysis structure, that is, the triplet of syntactic dependent (with particles and voice information), syntactic head (with voice information), and the case

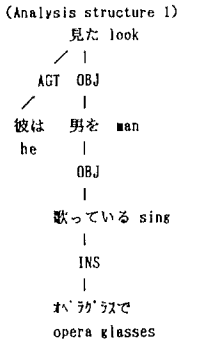
between them, combined with the instruction item forms the learning unit. The instruction item shows what the correction has been made on, namely, case or dependency correction. Each learning unit is accumulated in the association database. The database can be retrieved with the spelling of the syntactic dependent or head as the key. The learning unit corresponds to the following structure.

word2 (Syntactic head)
|
CASE1 (Case)
|
word1 (Syntactic dependent)

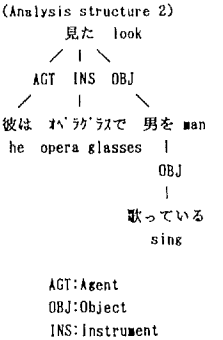
Example of the learning process and the applying process is shown below. This is the example of correcting dependency.

[Translation process at the first stage]
Source sentence:
彼はオペラグラスで歌っている男を見た。
↓(Translation)

Possible analysis structures:
(Analysis structure 1)



(Analysis structure 2)



AGT:Agent
OBJ:Object
INS:Instrument

If there is no information in the association database, analysis structure 1 is selected by further application of the rules.

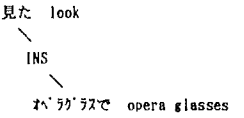
Translated sentence:
He looked at the man who is singing with opera glasses.

[Instruction by User and the Learning Process]
The user corrects the analysis results.

Correction of dependency:
The user changes the syntactic head of "オペラグラスで (opera glasses)" from "歌っている (sing)" to "見た (look)."

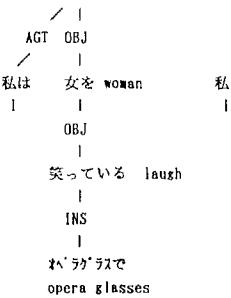
Translated sentence:
He looked at a singing man with opera glasses.

Learning:
PIVOT stores the correct analysis structure with dependency as the instruction item in the association database.

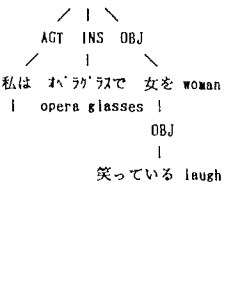


[Applying process]
PIVOT translates another similar sentence.

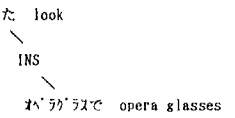
Source sentence:
私はオペラグラスで笑っている女を見た。
↓(Translation)
Possible analysis structures:
(Analysis structure 1)



(Analysis structure 2)



Database retrieval:
PIVOT retrieves information in the association database, because there exist two possible analysis structures.



Matching:
PIVOT succeeds in matching, and selects analysis structure 2.

Translated sentence:
I looked at a laughing woman with opera glasses.

3.3 Matching Methods
The learning mechanism decreases the number of user's instructions. The problem is to find the effective matching method in the learning mechanism.

We made experiments on four types of matching methods and compared the efficiency of each method.
The matching methods are:
(1) Restricted exact matching
(2) Non-restricted exact matching
(3) Restricted best matching

(4) Non-restricted best matching

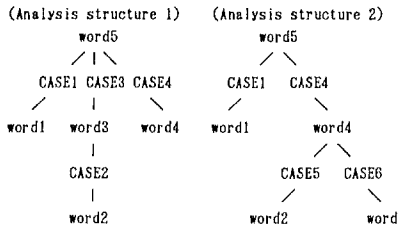
Restricted exact matching is a well-known method. This method is used in many fields now. There is no study about non-restricted exact matching. Restricted best matching is a comparatively new method. Experiment by Miura[4] is the first. There is no study about non-restricted best matching.

3.3.1 Restricted Matching and Non-restricted Matching

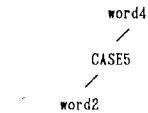
In restricted matching, the item in applying process has to be the same with the instruction item in learning. When the items are different, PIVOT will not use learned data. For example, if the instruction item in learning is case, PIVOT will use the learned correct analysis structure only for case selection. It will not use the data for selection of dependency or translation equivalent of each word.

In non-restricted matching, the item in applying process need not be the same with the instruction item in learning. For example, if the instruction item in learning is case, PIVOT will use this learned data for selection of dependency and translation equivalent of each word as well.

The difference between the actions of restricted matching and non-restricted matching is described below. Consider a sentence with two possible analysis structures.



Assume the following analysis structure is already learned by correcting case.

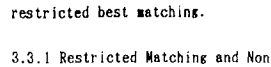


Using restricted matching, the system selects structure 1 with its usual analysis procedure. In this case, data learned by case correction cannot be used in selection of dependency. Using non-restricted matching, the system selects structure 2, because the learned pattern matches with the part of structure 2.

3.3.2 Exact Matching and Best Matching

Exact matching makes matching only once, while best matching makes matching several times. Best matching is also called associative reasoning.

The difference of actions between the two methods is illustrated below.



Let (C1,W2,W1) stand for the learned structure as shown on the left.

Suppose that the following data is accumulated in the association database through dependency instructions.

(C4,W3,W7)
(C3,W3,W2)
(C3,W5,W7)
(C1,W2,W6)
(C1,W3,W1)
(C1,W5,W1)
(C2,W3,W6)

Exact matching:

[Assumption]

There are two possible syntactic heads, W7 and W3, for W2.

[Action]

The association database is searched for patterns (*,W7,W2) and (*,W3,W2). (*:don't care)

Database	Search pattern	Matching
(C4,W3,W7)		
(C3,W3,W2)	(*,W3,W2)	(C3==*,W3==W3,W2==W2) Success
(C3,W5,W7)		
(C1,W2,W6)		
(C1,W3,W1)		
(C1,W5,W1)		
(C2,W3,W6)		

(C3,W3,W2) is selected as the correct answer.

Best matching:

[Assumption]

There are two possible syntactic heads, W7 and W5, for W2.

[Action]

First, the association database is searched for patterns (*,W7,W2) and (*,W5,W2). (*:don't care)

Database	Search pattern	Matching	
(C4,W3,W7)			
(C3,W3,W2)	(*,W7,W2)	(C3==*,W3!=W7,W2==W2)	Fail
	(*,W5,W2)	(C3==*,W3!=W5,W2==W2)	Fail
(C3,W5,W7)			
(C1,W2,W6)			
(C1,W3,W1)			
(C1,W5,W1)			
(C2,W3,W6)			

In this case, there is no data that exactly matches

with search patterns. However, there is data (C3,W3,W2) that matches with syntactic dependent. The system retrieves more information in the database so as to decide which of W5 and W7 is more similar to W3.

Searching database for patterns (*,*,W3) and (*,W3,*), the following data is obtained.

(C4,W3,W7)
(C3,W3,W2) Let this set of data be called
(C1,W3,W1) "database(W3)."
(C2,W3,W6)

Searching database for patterns (*,*,W7) and (*,W7,*), the following data is obtained.

(C4,W3,W7) Let this set of data be called
(C3,W5,W7) "database(W7)."

Searching database for patterns (*,*,W5) and (*,W5,*), the following data is obtained.

(C3,W5,W7) Let this set of data be called
(C1,W5,W1) "database(W5)."

On the assumption that W3 is the same as W7, the system performs exact matching between database(W3) and database(W7). In the following, [W3] is regarded as W7.

Database(W3)	Database(W7)	Matching
(C4,[W3],W7)	(C4,W3,W7)	Fail because [W3]==W7!=W3.
	(C3,W5,W7)	Fail
(C3,[W3],W2)		
(C1,[W3],W1)		
(C2,[W3],W6)		

On the assumption that W3 is the same as W5, the system performs exact matching between database(W3) and database(W5). In the following, [W3] is regarded as W5.

Database(W3)	Database(W5)	Matching
(C4,[W3],W7)	(C3,W5,W7)	(C4!=C3,[W3]==W5,W7==W7) Fail
(C3,[W3],W2)		
(C1,[W3],W1)	(C1,W5,W1)	(C1==C1,[W3]==W5,W1==W1) Success
(C2,[W3],W6)		

Because the number of matches between database(W3) and database(W5) is larger than that between database(W3) and database(W7), W5 is considered to be more similar to W3 than W7. W5 is selected as the head.

3.3.3 Matching Algorithm

Let PDBi(PCi,PHi,PDi,PTi) (1<=i<=n) be a possible analysis structure, where

PCi: Case, PHi: Head, PDi:Dependent, PTi:Item.
PDB is called "possible analysis structures database".

Let ADBk(ACk,AHk,ADk,ATk) (1<=k<=m) be an association database entry, where

ACk: Case, AHk: Head, ADk:Dependent, ATk:Item.
ADB is called "association database".

Matching algorithm for dependency selection is shown below. All PDi's in PDB are supposed to be the same and most of PCi's in PDB are supposed to be "don't care" for ease of understanding.

First Step:

Extract all ADBk's such that PDi==ADk(1<=i<=n, 1<=k<=m) from ADB and create SADBj(SCj,SHj,SDj,STj) (1<=j<=p), where

SCj: Case, SHj: Head, SDj:Dependent, STj:Item.
SADB is a subset of ADB.
If nothing is in SADB, stop search and return fail.

Second Step:

(1)Restricted exact matching

```
Let WORK be an empty database.
for i=1 to n
   for j=1 to p
      if (SCj==PCi & SHj==PHi & STj==PTi)
         then add PDBi to WORK;
      endif
   end
end
return WORK;
```

(2)Non-restricted exact matching

```
Let WORK be an empty database.
for i=1 to n
   for j=1 to p
      if (SCj==PCi & SHj==PHi)
         then add PDBi to WORK;
      endif
   end
end
return WORK;
```

(3)Restricted best matching

```
Let WORK1, WORK2 be empty databases.
cnt=0;
for i=1 to n
   for j=1 to p
      if (SCj==PCi & SHj==PHi & STj==PTi)
         then add PDBi to WORK1;
      endif
      else if (SCj==PCi & SHj!=PHi & STj==PTi &
               WORK1==NULL)
         then
           /* Calculate the similarity between
              SHj and PHi. */
              extract all ADBk's such that
                AHk==SHj or ADk==SHj (1<=k<=m)
              and create database X;
              extract all ADBk's such that
                AHk==PHi or ADk==PHi (1<=k<=m)
              and create database Y;
              assume SHj==PHi and perform restricted
              exact matching between X and Y;
              Let cnt1 be the number of matched
              entries between X and Y;
              if (cnt1>0 & cnt1==cnt)
                 then add PDBi to WORK2;
              endif
           /* Cnt is the largest number of matches
```

```
                made between X and Y, showing the
                degree of similarity between them. */
                else if (cnt1>cnt)
                   then
                      cnt=cnt1;
                      clear WORK2;
                      add PDBi to WORK2;
                endif
          endif
       end
    end
    if (WORK1 != NULL)
       then return WORK1;
    endif
    else return WORK2;
```

(4)Non-restricted best matching

The algorithm is the same as (3) except that non-restricted exact matching is performed between X and Y instead of restricted exact matching.

In the above, if more than one entries are in WORK or WORK1, the system will select one that is most recently stored by the user's instruction. If WORK2 has more than one entries, one entry will be selected by further application of the rules.

Matching algorithm for case selection is similar to that for dependency selection.

4. Experiments

Experiments have been made to evaluate the effect of learning mechanism described in Section 3 by simulation. In the experiments, the instruction items were limited to case and dependency.

A total of 1565 sentences were collected from six kinds of technical manuals. These sentences were translated with PIVOT/JE. Using the analysis editing function stated previously, correction of mistakes in dependencies and cases were made.

After all errors in the analysis results of the whole text were corrected, correction information for case and dependency was extracted and put into a file. A tool which simulates learning mechanism was prepared. After reading the file which stores the correction information, it counts the number of corrections to be made in each of the following cases: no application of the learned data, application with restricted exact matching, application with restricted best matching, application with non-restricted exact matching and with non-restricted best matching.

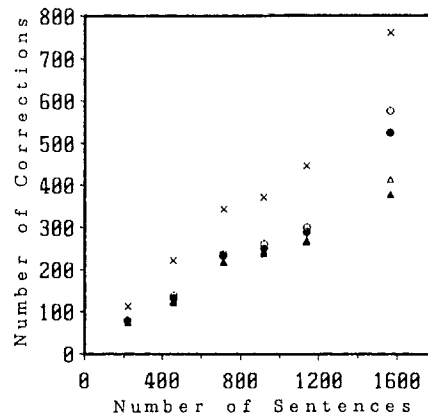
The results are shown in the table and the graph below. The value is the sum of the estimated number of the corrections and the estimated number of the recorrections needed to cancel the secondary effect.

Table 1

n Number of Sentences
× Without Learning
○ Restricted Exact Matching
△ Restricted Best Matching
● Non-restricted Exact Matching
▲ Non-restricted Best Matching

	Text 1	Text 2	Text 3	Text 4	Text 5	Text 6
n	220	456	713	920	1138	1565
×	112	220	345	372	447	760
○	81	137	236	262	301	576
△	76	127	217	243	271	414
●	78	131	232	251	289	524
▲	77	123	218	238	266	380

Graph 1



The results are shown in order of effectiveness.

1 non-restricted best matching
2 restricted best matching
3 non-restricted exact matching
4 restricted exact matching
5 without learning

Non-restricted best matching is the most effective among the five methods.

5. Conclusion

This paper discussed the learning mechanism for dependency and case corrected by the user. The learned data is accumulated in the association database. Four types of matching methods that are used in the applying process were examined. The simulation shows that non-restricted best matching is the most effective among the four types.

The learning mechanism discussed above is also effective for selection of a translation equivalent. This mechanism will be incorporated in PIVOT, taking over the current learning mechanism for selection of translation equivalents.

References

1. Nagao,M.: "A Framework of a Mechanical Translation between Japanese and English by Analogy Principle", in Artificial and Human Intelligence (Elithorn & Banerji, Eds.), Elsevier Science Publishers, pp173-180, 1984.

2. Shirai,K., Hayashi,Y., Hirata,Y., and Kubota,J.: "Database Formulation and Learning Procedure for Kakari-Uke Dependency Analysis", Transactions of IPSJ, Vol.26 No.4, 1985(in Japanese).

3. Stanfill,C. and Waltz,D.: "Toward Memory-Based Reasoning", CACM, 29-12, pp1213-1228, 1986.

4. Miura,M., Itahashi,S., and Nishino,H.: "Japanese Text Analysis System with Valency Frame", WGNL 63-4, IPSJ, 1987(in Japanese).

5. Inagaki,H., Kabeya,K., and Obashi,F.: "Modification Analysis using Semantic Pattern", WGNL 67-5, IPSJ, 1988(in Japanese).

6. Sato, S.: "Memory-based Translation II", WGAI 70-3, IPSJ, 1990(in Japanese).

7. Sumita,E., and Iida,H.: "Experiments and Prospects of Example-Based Machine Translation", WGNL 82-5, IPSJ, 1991.