Portable Knowledge Sources for 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 the acquisition and organization of knowledge sources for machine translation (MT) systems. It has been pointed out by many users that one of the most annoying things about MT systems is the repeated occurrence of identical errors in word sense and attachment disambiguation. We show the limitations of a conventional user-dictionary method and explain how our approach solves the problem.

1. Introduction

In the last decade, more and more commercial machine translation (MT) systems have become available for a wide variety of language pairs. An MT system is a very handy tool, but one quickly finds out that it makes the same errors over and over again even if a user dictionary is carefully maintained. There are several reasons for such repeated errors.

1. Commercial MT systems are not built in accordance with a powerful lexical semantic formalism. The user dictionary alone cannot disambiguate word senses and phrasal attachments satisfactorily.
2. MT systems cannot handle the domain and context dependency of word sense, phrasal attachment, and word selection.
3. In a shared environment, each user has a different user dictionary, and must therefore redundantly correct the same errors as all the other users.

A powerful lexical semantic approach [8] could give more accurate translation, but it might be too much to ask users to develop their dictionaries within that formalism. The simple structure of a user dictionary also restricts the learning ability of MT systems during the post-editing process. The second of the above reasons has motivated recent example-based and case-based machine translation research [9, 6, 10]. However, a method for finding the best-matching cases in a case base, where cases (or examples) are collected from different domains or contexts, has not been studied well. Nor is it known whether considering the frequency of cases gives a better result. The third reason is rarely discussed, but it is not desirable simply to share a single user dictionary, since the dictionary may become inconsistent by reflecting multiple users' updates. McRoy [5] discussed word sense disambiguation using multiple knowledge sources, but her method is still dictionary-based.

Some of the commercial systems for human-aided translation, such as the Translation Manager/2 [2], can provide the user with more flexible access to multiple dictionaries and the *translation memory* (a repository of pairs of source and target sentences). This organization of knowledge could be quite useful for selecting correct translations of words, but the types of knowledge available from the dictionaries and translation memories are rather limited, and are certainly not enough for resolving structural ambiguities in sentences.

In this paper, we propose *Portable Knowledge Sources* (PKSs) for machine translation. A PKS consists of preference information on word sense, phrasal attachment, and word selection for translation. It is acquired through user interaction in the post-editing process, and is stored with the document being translated. When translating a document by using an MT system, a user can specify a list of already-translated documents, and the system will make use of the PKSs included in the specified documents. We show how such a collection of PKSs is organized, used for translation, and integrated into a user dictionary, and how the problem stated above can be solved by using PKSs.

2. Portable Knowledge Sources

A Portable Knowledge Source (PKS) consists of preference information on three kinds of ambiguity:

1. Word sense
2. Phrasal attachement
3. Word selection

The preference information is acquired from the user through post-editing or interactive translation [4, 1], and is *paired* with the document that the user is working on. That is, a PKS is stored and managed together with the document for which it is created.

Let PK1, PK2, and PK3 be PKSs for the respective types of ambiguity mentioned above. The following is an example of *word sense* ambiguity:

Delete the *line*.

The word *"line"* could be (1) a single row of letters, (2) a geometric mark, (3) a hardware wire, and so on, for each of which a different translation is usually required in a target language. When the user specifies that a particular occurrence of the word "line" in a document D means a single row, the PKS

```
(PK1 ("line" (cat n)) (sense 1))
```

is created, and is stored with D.

An example of *phrasal attachment* ambiguity is as follows:

> Order the publication through the IBM branch *serving your locality.*

The present participle phrase can be either attached to the main verb "order," or to the noun "branch." If the user specifies that it modifies the noun as a post-nominal adjective phrase (ADJP), the PKS

```
(PK2 ("serve" (cat v) (form prsprt))
     ADJP ("branch" (cat n)))
```

is created.[1] The preference of prepositional phrase attachment is also represented by PK2.

Finally, an example of *word selection* ambiguity is (1) "メモリー・チップ" and (2) "記憶素子" for the compound noun "memory chip", where the first translation can often be found in PC documents, while the second one, which has the same meaning, is typically used in textbooks. When the user specifies that the second one should be used, the PKS

```
(PK3 ("memory chip" (cat n))
     ("記憶素子" (cat n)))
```

is created. If word sense is to be included in the definition of word selection, such that the word W1 is used in sense S and should be translated by the word W2, it is separately represented by (PK1 W1 S) and (PK3 W1 W2).

Each PKS collected through user interaction has an *age*, based on the time and date of its creation. The younger the PKS, the stronger its preferability in one document, since it could have been used to overrule the preceding PKSs. Note that the age of a PKS is valid only among other PKSs in the same set. Two sets of PKSs are not comparable if they are paired with different documents.

3. Organizing Portable Knowledge Sources

Once the user has translated several documents, the sets of PKSs paired with them begin helping the MT system to resolve the three kinds of ambiguity described in the previous section. When a new document is to be translated, the user either specifies a list of previously translated documents as a source of available PKSs, or lets the MT system automatically choose them. Such a list of documents is called a *document list.*

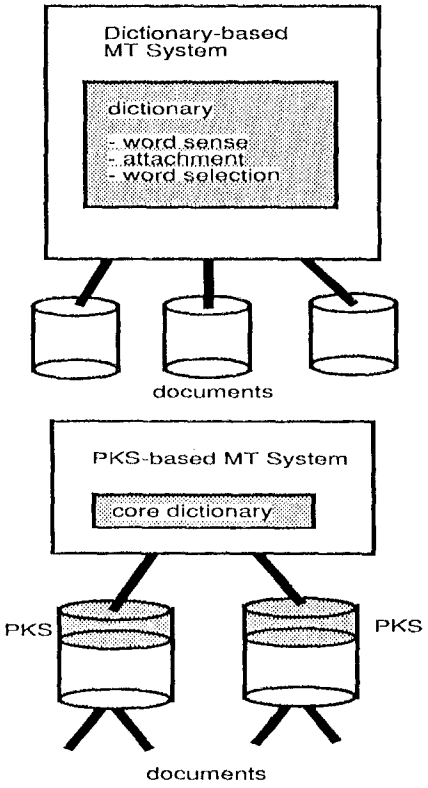

Figure 1: Dictionary-based and PKS-based MT Systems

Figure 1 compares a PKS-based MT system with that of a conventional dictionary-based MT system.

Even though a logical document may not be identified with a physical file, it is the easiest and most practical way to organize the hierarchy of documents. In practice, when translating technical documents, it is usual to translate the glossary first, agree on the translations of technical terms, and then work on individual chapters. This gives us a natural ordering of documents,

$$\text{glossary} \rightarrow \text{chapter 1} \rightarrow \text{chapter 2} \ldots$$

which is also used as an ordering of PKSs to be incorporated for machine translation. One way of automatically choosing the document X for translating a new document Y is to calculate the overlap of words contained in both X and Y, and to find the X with the largest overlap. This idea is similar to the context identification method [13], which is used effectively for word sense disambiguation.

[1] The representation of the PKS can vary depending on the MT system that uses the PKS. For example, the modifier and modifiee phrases can be represented by syntactic structures; word senses and semantic case relations can be associated; and so on.

One important characteristic of this PKS organization is that it can be dynamically rearranged. We can invalidate some PKSs by simply removing a document from the document list, or validate a new set of PKSs by adding its paired document to the list. This is extremely useful for domain-sensitive and context-sensitive translation, since a close look at documents in a seemingly similar domain will show that there are too many conflicting word senses and word selections to build a single consistent domain dictionary.[2] In the worst case, the user has to keep on asking the system to prefer one of several word senses as many times as a new document arrives to be translated.

Another important observation is that the system can calculate the quality of preference information as follows:

1. Given a document list, find all the PKSs in the document list, and create a *PKS graph*, which is a directed graph, for each type of PKS (see Figure 2):
   - If the PKS is of type (PK1 $word_i$ $sense_j$), create a node Nw_i for $word_i$, a node Ns_j for $sense_j$, and a directed arc a_{ij} (labeled *"sense"*) from Nw_i to Ns_j.
   - If the PKS is of type (PK2 $word_i$ role $word_j$) create a node Nr_i for $word_i$, a node Nl_j for $word_j$, and a directed arc a_{ij} (labeled with a *syntactic role*) from Nr_i to Nl_j.
   - If the PKS is of type (PK3 $word_i$ $trans_j$) create a node Ns_i for $word_i$, a node Nt_j for $trans_j$, and a directed arc a_{ij} (labeled *"trans"*) from Ns_i to Nt_j.
2. Count the number C1 of *conflicting arcs* for the PK1 and PK3 graphs. That is, find the number of arcs leaving the same node but going to different nodes.
3. Count the number C2 of *conflicting paths*[3] for each pair of nodes n1 and n2, connected by an arc a1 in the PK2 graph, such that for some node n3, there are two arcs a2 from n1 to n3, and a3 from n3 to n2, where a2 and a3 have the same label (see Figure 3).

Intuitively, C1 shows a number of ambiguous word senses and word selections, and C2 shows possible attachment ambiguities in the given document list.[4]

[2] Recall the word senses of the word "line" in Section 2. All of them appear in the computer domain, notably in the areas of text editors, graphics, and hardware manuals, respectively. Our approach allows the user to adjust the sense dynamically for each type of manual.

[3] A path is a sequence of directed arcs. Conflicting paths are two or more distinct sequences of paths between a given pair of nodes.

[4] Suppose that n1 is a prepositional phrase, n2 is a verb phrase, and n3 is another prepositional phrase. Then, we have an ambiguity in the n1 attachment. Multiple outgoing arcs from a node in the PK2 graph do not necessarily imply ambiguities.

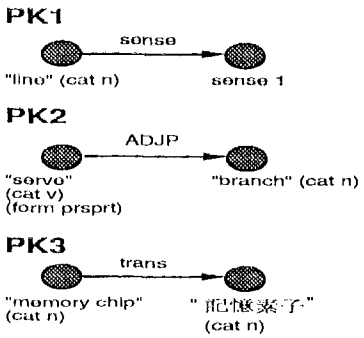

Figure 2: PKS graphs

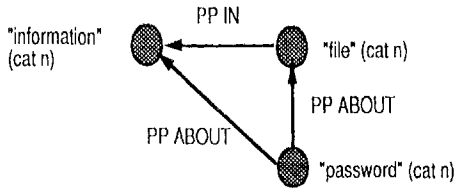

Figure 3: Conflicting paths in a PKS graph

Therefore, a document list with large C1 and/or C2 generally should be divided into smaller lists for consistent translation. In an ideal situation, C1 and C2 both should be 0. Gale et al. [3] and Nasukawa [7] suggest that there is a strong tendency for C1 to be very small in a reasonable span of text.

It is easily shown that for any two document lists L1 and L2 ($L1 \cap L2 = \phi$), the numbers of conflicting arcs and paths, C_{11}, C_{12}, C_{21}, and C_{22}, respectively, are *monotonic*. That is, the numbers C1 and C2 of conflicting arcs and paths of the combined document list L ($= L1 \cup L2$) satisfy $C1 \geq C_{11} + C_{21}$ and $C2 \geq C_{21} + C_{22}$.

4. Disambiguation Method

The basis of disambiguation of the three types of ambiguity discussed in Section 2 is to prefer the last PKS in the document list that matches the ambiguity, and to interpret the PKS as a rule for selecting a word sense, phrasal attachment, or word translation.

If there is no matching PKS, either the ambiguity is properly handled by the system, which results in no user correction in the documents, or it is new to the system. In the former case, the user will probably be satisfied with the translation by the system. In the latter case, the translation may have to be corrected by the user, but the interaction will be recorded as a new PKS and used for future disambiguation.

The matching algorithm for PK1 and PK3 rules is a simple exact matching of words and lexical features. If

two or more PKS rules match the ambiguous word, the ages of the rules and the ordering of documents in the document list uniquely determine the most preferable PKS. The PK2 rules, however, can be used with a more flexible matching algorithm [6, 12], since the coverage of PK2 rules would be very limited if two phrases (the modifier and modifiee phrases) had to match the rule exactly.

Once the document list has been given, the PK1 and PK3 rules can be polynomially converted into a simple lookup table, where the key is an ambiguous word, and only the most preferable rules are stored or retrieved.[5] PK2 rules can be organized similarly as a ternary lookup table.

It should be noted that sentences in a document list can be utilized as an example base [9], since the documents in the document list has already been translated, and the translation of the source sentence is readily available. Indeed, the conventional matching algorithm for a flat example base has to be extended into a hierarchical one, where the latest translation has the highest priority, and PKSs must be equally taken into consideration.

5. Knowledge Source Compilation

When a set of documents in one domain grows considerably, or when the MT system is to be transported to a different environment, it is convenient to be able to compile PKSs into a single, portable user dictionary. The compilation is similar to the creation of lookup tables, described in the previous section. The numbers of conflicting arcs and paths should be carefully examined to see whether a given document list yields a consistent user dictionary. The user can rearrange the ordering of documents, and choose the most preferable among conflicting PKS rules to make the optimal user dictionary for the domain.

The rearrangement of documents in the document list does not change the resulting PKS graphs. It just changes the preferences among the conflicting arcs or paths. Therefore, the optimal construction of a user dictionary does not have to consider an exponential number of possible document orderings, but only a polynomial number of the following pairwise constraints:

- If there are conflicting arcs a_1, a_2, ..., a_k in the PK1 (or PK3) graph, and the most preferable arc is a_i, the document d_i having the PK1 (or PK3) rule for a_i must be preceded by each document d_j having the arc a_j ($j = 1, \ldots, k.\ j \neq i$).
- If there are conflicting paths p_1, p_2, ..., p_k in the PK2 graph, and the most preferable path is p_i, the document d_i having the PK2 rule for the first arc in p_i must be preceded by each document d_j having the PK2 rule for the first arc in p_j ($j = 1, \ldots, k.\ j \neq i$).

It is polynomially decidable whether there is an ordering of documents that satisfies all of the above constraints. An ordering of documents exists iff the constraints are not cyclic (that is, iff there is a document D that must precede itself). Even if there is no linear ordering of such documents, the user dictionary can still be created from the user-selected arcs and paths. In this case, however, there is no natural correspondence between the user dictionary and a document list. Such a correspondence is indispensable if the user wishes to update the user dictionary when a new document is added at an arbitrary position in a document list. If the user dictionary is equivalently reducible to a document list, recompilation of the document list into the user dictionary is straightforward. When no such equivalent list exists, a document may only be added to the tail of the list, thus overruling all the conflicting PKSs.

6. Alternative Views of Knowledge Organization

In Section 3, we took a simplified view of the PKS organization, which we may call an "optimistic organization." It was implicitly assumed that only the elements in PKSs can conflict with each other. However, the system's default choice of word senses, may have satisfied a user, but may conflict with a PKS newly added to the document list. Thus, PKSs need to be more carefully organized if the user thinks that the translation by the system is adequate without the PKSs. This view may be called a "pessimistic organization" of PKSs. The optimistic organization is easier to implement, while the pessimistic organization can provide users with more consistent translation.

In the pessimistic organization, the conflicting knowledge has to be defined in terms of PKSs and the *choices* of word sense, phrasal attachment, and word translation by the system. Technically, it means that for every PKS in the document list, we have to examine if each rule in the PKS to determine whether it conflicts with a preceding PKS rule or a choice by the system. This is often very time-consuming. One way to deal with this pessimistic view is to keep track of the document list with which a new document is translated. It can easily be shown that time-consuming checking of PKS conflicts can be avoided by employing a monotonously growing sequence of document lists, $\{d_1\}, \{d_1, d_2\}, \ldots, \{d_1, d_2, \ldots, d_k\}$ such that each document d_i in the list has been translated by using the PKSs in the document list $\{d_1, d_2, \ldots, d_{i-1}\}$.

[5] Alternatively, all the conflicting PKS rules can be stored to give the user as many candidates as possible.

7. Conclusions

In this paper, we have proposed a new machine translation mechanism based on portable knowledge sources. It provides MT systems with an efficient way of acquiring and utilizing the vital information from the user in order to gradually achieve correct translation in a multi-domain, multi-user environment. Since the document list (or a list of previously translated documents) is usually read-only, it is not only a convenient unit for storing domain-oriented disambiguation knowledge, but also an ideal resource for machine translation that can be shared by many users.

We have started organizing our knowledge sources into a document list. The current documents consist of four IBM AS/400 computer manuals with about 22,000 sentences and a CAD manual with about 10,000 sentences. Currently only about 1,200 sentences have been translated by using our prototype MT system Shalt2[11], and corrected by the user for knowledge acquisition. The PKS rules are therefore few and not ready for quantitative analysis. Some interesting occurrences of words, however, have already been found. In one of the above manuals, for example, the noun "part" was used in two different word senses: as a hardware component and as an abstract portion of a whole. The first usage of this noun, however, seemed more restricted: it was always modified by a proper noun. In one manual, the word "line" was clearly and consistently used to mean a specific row on the screen, while in another it was always used to mean a geometric component.

The interaction of PKS rules has not been discussed in this paper, although it is a very interesting topic. Suppose that PK2 rules can also include the word senses for each word. It is not clear whether the word senses in the rules should be considered as *conditional* (valid only if the pair of modifier and modifiee is present) or *absolute* (no matter what the modifier and modifiee are). Only the latter case is handled by the PK1, and the former cases could be the exceptions to the PK1 rules.

If the "one sense per document" assumption seems too strong, we can modify the PK1 and PK3 rules by adding contextual dependency to the rules so that a word sense or a word translation is valid only if the word appears with a certain modifier or modifiee, has a certain syntactic role, and so on.

Acknowledgement

I would like to thank Tetsuya Nasukawa and Naohiko Uramoto for many discussions which have gradually materialized into the idea of portable knowledge sources. The comments from Hideo Watanabe and Hiroshi Nomiyama also contributed to the refinement of the technical contents. Michael McDonald, as usual, proofread the paper, and helped me write the final version.

References

1) C. Boitet and H. Blanchon. "Dialogue-Based MT for monolingual authors and the LIDIA project". In *Proc. of the Natural Language Processing Pacific Rim Symposium '93*, pages 208–222, Fukuoka, Japan, Dec. 1993.

2) IBM Corp. "IBM SAA AD/Cycle Translation Manager/2 Release 1.0". Technical report, SH12-5906, 1992.

3) W. Gale, K. Church, and D. Yarowsky. "One Sense Per Discourse". In *Proc. of the 4th DARPA Speech and Natural Language Workshop*, 1992.

4) H. Maruyama, H. Watanabe, and S. Ogino. "An Interactive Japanese Parser for Machine Translation". In *Proc. of the 13th International Conference on Computational Linguistics*, pages 257–262, Helsinki, Aug. 1990.

5) S. W. McRoy. "Using Multiple Knowledge Sources for Word Sense Discrimination". *Computational Liguistics*, 18(1):1–30, Mar. 1992.

6) K. Nagao. "Dependency Analyzer: A Knowledge-Based Approach to Structural Disambiguation". In *Proc. of the 13th International Conference on Computational Linguistics*, pages 282–287, Helsinki, Aug. 1990.

7) T. Nasukawa. "Discource Constraint in Computer Manuals". In *Proc. of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 183–194, Kyoto, Japan, July 1993.

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

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

10) E. Sumita and H. Iida. "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 1991.

11) K. Takeda, N. Uramoto, T. Nasukawa, and T. Tsutsumi. "Shalt2 - A Symmetric Machine Translation System with Conceptual Transfer". In *Proc. of the 14th International Conference on Computational Linguistics*, pages 1034–1038, July 1992.

12) N. Uramoto. "Lexical and Structural Disambiguation Using an Example-Base". In *Proc. of the 2nd Japan-Australia Joint Symposium on NLP*, pages 150–160, Oct. 1991.

13) D. Yarowsky. "Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora". In *Proc. of the 14th International Conference on Computational Linguistics*, pages 454–460, July 1992.