Enhancing Translation Memories with Semantic Knowledge

Natalia Elita
Technical University of Moldova
Informatics and Applied Modern Languages Department
Republic of Moldova
E-mail: vnatalia@mail.md

Monica Gavrila
University of Hamburg
Natural Language Systems Division
Germany

E-mail: gavrila@nats.informatik.uni-hamburg.de

The translation process can often be time-consuming, boring, and difficult. Translators need assistance for a faster, more qualitative, and accurate translation. Translation Memories (TMs) are tools that help them in doing this job that is not an easy one. In our paper, we present a new approach for building Translation Memories enhanced with semantic information.

The paper is structured in five sections, as follows: The first section includes theoretical considerations, mainly why are TMs used, and the TMs types. Section 2 describes our motivation. The Semantic Template Driven Method is described in more details in Section 3. Some preliminary evaluation results are given in the next section. In Section 5, further work and conclusions are presented.

1. Theoretical Considerations

The **Translation Memory** (TM) is a type of database that is used in programs designed to aid human translators. It is used together with word processing programs, terminology management systems, and multilingual dictionaries. A TM consists of text segments (e.g. blocks, paragraphs, sentences, phrases) in the source language (SL), and their translations into one or more target languages (TL).

TMs are used to allow translators increase their productivity by making easy for them to recycle parts of their translation. They guarantee a certain level of quality, as they recycle translations done by humans, and they allow non-negligible productivity gains, particularly on highly repetitive texts.

Three types of TMs are known: first, second, and third generation TMs.

First generation TMs are distinguished by the fact that they store pairs of complete sentences. Repetitions are searched on the level of a full sentence. The biggest problem of these TMs is that repetitions on the whole sentences rarely occur.

Second generation TMs appeared to overcome this big disadvantage of the first generation TMs. In this new TMs the two source sentences (the input and the example in the database) are considered to be identical if they differ slightly (with regard to name entities or edit distance operations). The concept of fuzzy matching was introduced together with this type of TMs. This also did not prove to be very effective. This is why third generation TMs appeared. They already give the possibility to search for repetitions on a sub-sentential level.

More details on TMs can be found in [1],[2], and [7].

2. Motivation

Our motivation comes with the following possible scenario: suppose we have a translator who wants to build a translation memory from the different translations he has already done. The resulting TM would consist of translation templates with variables that have

to be instantiated. We propose a method of building such a translation memory. Our approach has the following advantage: the presence of semantic information in the templates makes possible word sense disambiguation, which can prevent learning false templates. We intend to integrate the given approach also in an Example-based Machine Translation (EBMT) system[1].

3. Semantic Template Driven Method

In this section we give a detailed description of the needed resources, and of the template extraction algorithm.

In order to be able to extract semantic templates we need several resources and modules, such as a corpus of SL texts with their translations, a template extraction engine, a bilingual lexicon (semi) automatically derived from corpus, and some domain ontology. A short overview of these resources is made in the following sub-section.

3.1. Resources

In our work, we have several requirements towards the corpus. The languages involved are Romanian, Russian, English and German. Sentences are 1:1 aligned, and are exact translations of each other.

Another resource to be used is the bilingual lexicon. It is obtained via word-aligned SL and TL texts.

The domain ontology plays an important role, as the lexical entries from the bilingual lexicon are mapped on ontology concepts, and the corpus is annotated (semi-automatically) with concepts in the lexicon.

3.2 The Template Extraction Engine

The Template Extraction Engine implements a several phase algorithm.

In the first phase we use the template extraction algorithm proposed by McTait [6]. This algorithm is claimed to be language – independent. Our first step includes also testing this claim.

Before the algorithm description, we introduce the notion of translation template. **Translation templates** (TT) are generalizations of sentences that are translations of each other, where sequences of one or more words are replaced by variables, with alignments between the resulting word sequences, and/or variables made explicit. They are extracted from a bilingual corpus aligned at the level of sentence by a language neutral, recursive machine-learning algorithm. The algorithm is based on the principle of similar distributions of strings: source language (SL) and target language (TL) strings that co-occur in two (or more) sentence pairs of a bilingual corpus are likely to be translations of each other. TTs are formed from lexical items that occur minimum twice in the corpus, which means that the algorithm is useful in instances of sparse data.

Defined formally, a TT is a 4–tuple ***{S, T, Af, Av}***, where ***S*** is the sequence of SL sub-sentential text fragments, separated by SL variables, ***T*** the sequence of TL sub-sentential text fragments, separated by TL variables, ***Af*** the global alignment of text fragments between S and T, and ***Av*** the global alignment of variables. A possible configuration of S and T is:

$$F_1^S, V_1^S, F_2^S, V_2^S \ldots F_p^S, V_p^S <-> F_l^T, V_1^T, F_2^T, V_2^T \ldots F_q^T, V_q^T$$

An informal example of a TT is:

[1] More details on EBMT systems, and template extraction can be found in [3],[4], and [5].

<en>X **gave** Y **up**</en> <-> <ro>X **a renuntat la** Y</ro>

As the algorithm operates on the surface forms of lexical entries, there are great chances that false templates can be learned. For example, from the following two sentences in English, and their translations in Romanian:

1. The artist *played* wonderfully yesterday, at Music Hall (En) <-> Artistul *a cintat minunat ieri la Sala de Muzica* (Ro),

2. Stevenson *played for a new team yesterday, in the Central stadium* (En) <-> Stevenson *a jucat pentru o echipa noua ieri la stadionul central* (Ro),

the template **X played Y yesterday Z** can be learned[2]. It is false, because the English word "***played***" has different translation equivalents in Romanian. We propose as a solution to the false template problem the usage of semantic constraints (i.e concepts from ontology). This way, for the above-mentioned example, the following 2 semantic templates can be extracted:

1. X (C002_music) played Y yesterday Z
2. X (C070_sport) played Y yesterday Z

The input to the template extraction engine is a bilingual corpus aligned at the sentence level, the output is a set of translation templates. The algorithm tends to be language-neutral, and it operates on simple principles of string co-occurrence, and frequency thresholds. Template extraction is divided into three stages: monolingual, bilingual, and alignment.

The fist stage is a Monolingual Phase and it consists of SL (TL) Tokenisation, SL (TL) Word List Creation, and SL (TL) Collocation Tree Formation. All these three steps are detailed below.

Tokenisation is the process of dividing a string into tokens.

Word List Creation is the next step, where from the list of tokens obtained in the previous step the list of words is created as in the following example. From the sample corpus entries:

(1) The commission **gave** the plan **up** (En) <-> Comisia **a renuntat la** plan (Ro);
(2) Our government **gave** all laws **up** (En) <-> Guvernul nostru **a renuntat la** toate legile (Ro);

the lexical items occurring in two or more SL and TL sentences are collected. As a result the following output at this stage is produced:

(gave)[1,2], (up) [1,2], (a)[1,2], (renuntat)[1,2], (la) [1,2].

Here the integers denote the sentences from which they were retrieved.

The Collocation Tree Formation is the next phase. A collocation in our case is a data structure representing (possibly discontinuous) strings that co-occur in two or more sentences.

Lexical items combine recursively to form a tree-like data structure of collocations. Each lexical item is tested to see whether it can combine with any daughters of the root node, and if so, recursively with each subsequent daughter (and the daughters of that node and so on), as long as there is an intersection of at least two sentence IDs.

The combination process is constrained only by the integer IDs of the sentences from which the lexical items were retrieved, and the frequency threshold (a minimum of 2). The intersection constraint enforces string co-occurrence in two or more sentences.

If the node to be added cannot combine with any daughter of the root (or parent) node, it is added as a new daughter of the root (or parent) node. Consider the following example: given the word list: (gave)[1,2], (up) [1,2] a collocation tree as in Figure 1 can be built.

The collocation trees are not always so simple. For a given List of Words with their sentence ids: {(government)[1,2,3,4], (implemented)[2,3,4], (plan)[3,4], (policy)[2,3],

[2] The templates are learned if the minimum occurrences threshold is reached.

(decided)[1,2], (committee)[5,6,7]} a tree of collocations of increasing length, but decreasing frequency is formed, as the tree is descended from the root node to the leaves[3].

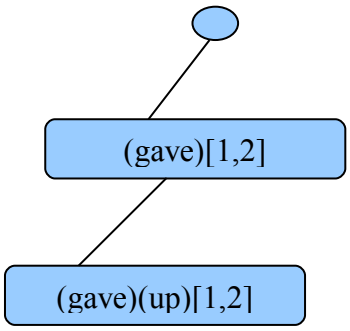

Figure 1. Collocation tree for the word list (gave)[1,2], (up) [1,2]

The leaves become the most informative parts of the tree and are collected at the end of this phase. The leaf-collocations are filtered so, that only the longest are selected. In our experiments we encountered several problems, for which we have to find solutions. For example useful information can be lost, the sequences extracted in SL and TL with the same sentence ID are often not translations of each other, etc. Collocations that are subsumed by other collocations with the same sentence IDs are removed.

The next phase is the Bilingual Phase. The filtered SL and TL leaf-node collocations extracted in the monolingual phase are equated on the basis of simple co-occurrence criteria. According to McTait [6] the used criterion is: SL and TL collocations that share exactly the same sentence ids are equated. Sometimes, in practice, the usage of this criterion can lead to losing some of the templates.

The correct orthography or the order of the lexical items and discontinuities that make up a translation template is computed by scanning the sentences from which the lexical items were retrieved. Sometimes there are several possibilities to choose the sentences from which the order is retrieved. A solution is to pick up the one that gives you the minimum number of discontinuities, but in some (test) cases this might not be the perfect solution.

Having the collocation trees for both languages, the text fragments and variables need to be aligned (the Alignment of Text Fragments and Variables Phase). This produces translation templates that are flexible enough for the recombination phase when used in MT, where TL translations are produced. As a result a bilingual lexicon of phrasal translations is obtained.

First, templates are extracted without any semantic information, following the above-mentioned algorithm. In the next phase, after the extracted templates are carefully examined, semantic concepts are attached the lexicon entries, and the original corpus is annotated with the extracted concepts. Then, the following phase of template extraction is run. This phase is similar to the first one, but the concepts are considered. This way, the output is not made of simple templates, but of templates enhanced with semantic information.

4. Preliminary Evaluation

The results of our preliminary evaluation are included in table 1.

[3] Examples taken from McTait [6].

No	*Data*	*Threshold*	*Templates learned*		*Comments*
			***useful**	****less useful**	
1	sentences: 56 words: 478 **templates: 112**	2	45	67	*News items in English*
2	sentences: 56 words: 478 **templates: 40**	3	21	19	*News items in English* **LOST 15 useful templates, ADDED 1 new template (compared to experiment 1)**
3	sentences: 56 words: 478 **templates: 17**	4	4	13	*News items in English* **LOST 3 useful templates compared to (2), LOST 16 useful templates compared to (2),**
4	sentences: 162 words: 2811 **templates:123**	2	64	59	*News items in English*
5	sentences: 162 words: 2409 **templates: 113**	2	39	74	*News items in Romanian*
6	sentences: 214 words: 3667 **templates: 68**	2	31	37	*News items in Romanian*

Table 1. Preliminary evaluation

We consider *useful to be contiguous or non-contiguous text fragments containing at least one content word and **less useful the extracted sentence fragments containing only functional words. The threshold is the minimum number of sentences the lexical item has to occur. We made several experiments with different number of tokens, and came to the conclusion that the maximum number of templates is learned when the threshold is 2. Another observation made is that for the same number of sentences in different languages, a different number of templates is learned. Of course, in some point of our experiments we have to calculate the precision and the recall. As we are at the beginning of our work, this is still not possible.

5. Further Work and Conclusions

The work is very much under development, so we are just in the first phase of the project. Further we intend to improve the monolingual phase, implement the bilingual and alignment phase of the template extraction mechanism. Then, as soon as this is done, after careful analysis of the extracted templates, we will create the necessary ontology. More tests have to be done, in order to be able to calculate the Precision and Recall.

In this paper we presented a new methodology for TM creation enhanced with semantic information. It can be useful in automatic word sense disambiguation. It prevents

learning false templates, and tends to be language–neutral. In the end it will be integrated also in an EBMT system.

References

[1] J. Allen, "Adapting the concept of 'Translation memory' to 'Authoring memory' for a Controlled Language writing environment", Translating and Computer Conference, London, 1999

[2] H. Gabor, G. Tamas, K. Balazc, "Translation memory as a Robust Example-based Translation System", in EAMT (2004), pp.82-89, 2004

[3] F. Gotti, P. Langlais, E. Macklovitch, D. Bourigault, B. Robichaud, C. Coulombe "3GTM: A Third Generation Translation Memory" in 3rd Computational Linguistics in the North-East (CLiNE) Workshop, Gatineau, Québec, aug 2005

[4] H. Güvenir, I. Cicekli: "Learning translation templates from examples", Information Systems 23, pp. 353-363, 1998.

[5] H. Kaji, Y. Kida, Y. Morimoto: "Learning translation templates from bilingual text" in Coling, pp. 672-678, 1992.

[6] K. McTait, "Translation Patterns, Linguistic Knowledge and Complexity in an Approach to EBMT" in M. Carl, A. Way (eds), Recent advances in Example-based Machine Translation, Kluwer Acad. Publ. pp. 307-338, 2003

[7] R. Schäler "Beyond Translation Memories" Workshop on EBMT, MT Summit, VIII, September 2001.