MetaMorpho TM: A Rule-Based Translation Corpus

Grőbler, Tamás; Hodász, Gábor; Kis, Balázs

MorphoLogic
Orbánhegyi út 5. H-1126 Budapest, Hungary
{grobler;hodasz;kis}@morphologic.hu

Abstract

This paper discusses the aspects of bi-lingual resource processing within a rule-based translation memory (TM) system currently being developed. Translation memories can be viewed as translation tools incorporating parallel corpora, mainly aligned at the sentence level. Usually, these corpora have no linguistic annotation, as commercial TM systems perform queries at the character level, using fuzzy matches.

The proposed translation memory system uses linguistic analysis (morphology and parsing) to determine similarity between two source-language segments, and attempts to assemble a sensible translation using translations of source-language chunks if the entire source segment was not found. This is achieved by integrating a rule-based machine translation (RBMT) engine. The drawback of this approach is language-dependence; however, proper grammar acquisition methods are being developed to speed up grammar preparation for further language pairs.

This paper addresses the problem of adding sufficient linguistic annotation to segment pairs – translation units (TU) – for new segment pairs to integrate with the RBMT scheme. This should be fully automatic because adding a new translation unit to a translation memory must be transparent, without requiring user reaction. The paper discusses a robust enough method to obtain as much linguistic annotation as possible, while keeping the error rate low.

0. Introduction

Translation memories (TM) can be viewed as CAT tools incorporating parallel corpora of existing translations. Traditionally, translation units in such corpora are sentence pairs, and the search method is based on character-level similarity, using fuzzy matches.

We are developing a TM system that applies the machinery of a rule-based machine translation (RBMT) system to compose the target sentence from the stored sub-sentential translation units. MetaMorpho TM is thus a linguistically enriched TM system that uses morphology and syntax in both languages to exploit the contents of the TM database to a greater extent, compared to commercial TM systems. The underlying machine translation system is currently being developed for the English-Hungarian and the Hungarian-English language pairs.

Section 1 of this paper provides an outline of the proposed TM system, showing the characteristics of the architecture and the mechanisms. This defines requirements for the annotation scheme itself, which is discussed in Section 3.

As these mechanisms are still being developed, only preliminary evaluation results are presented in the paper. Section 3, however, discusses evaluation methods in detail, presenting some preliminary results, expectations and estimates.

This paper treats translation memory databases both as lexicons and parallel corpora, as the proposed annotation scheme uses annotation to transform the latter into the former, using methods discussed in Sections 1 and 2.

The availability of proper sentence-level alignment is implied throughout the paper. Though this is task is far from obvious – and the proposed system is indeed accompanied with a sentence aligner, having less emphasis in this paper –, we are focusing on linguistic annotation. In special cases, when a human translator is actually working on a translation, alignment is inherent because the translation tool determines the scope of the current translation unit. Within the proposed scheme, proper linguistic annotation should still be performed in these cases.

1. Outline of the proposed TM system

Translation memory systems maintain a database of existing translations. Such databases are practically sentence-aligned but unannotated parallel corpora. The success of a TM system depends entirely on the lookup structure associated with the parallel corpus. In commercial TM applications, the lookup structure is a fuzzy index, which helps the system find source segments not entirely identical to the current source segment (i.e. on which the translator is currently working). The similarity measure is based on the character codes, and does not take into account the linguistic properties of either the stored segments or the current segment.

Another problem of commercial TMs is that they handle the translations on an 'as-is' basis: if a source segment is found at whatever level of similarity to the current one, the stored translation is inserted to the current target text, leaving it to the translator to adjust the translation to the contents of the current source segment. In this scheme, no smaller unit than a single segment (usually a sentence) can be looked for in the database.

1.1. Integration of RBMT

The proposed scheme is being built around a rule-based machine translation module named MetaMorpho. Provided the appropriate grammar lexicon, the MetaMorpho module is able to determine the structure of the source segment, and produce an automatically generated translation. The key benefit of this module – the one exploited in the translation memory scheme – is that the atomic unit of a grammar is a lexicalized syntactic pattern. Therefore, the grammar does not consist of abstract rules, but mainly syntactic patterns that hold the properties of an idiomatic or otherwise lexically constrained phrase. Thus collocations with a non-compositional meaning or translation can still be translated correctly, with all their necessary variations taken into account [1] [2].

The fundamental design of the proposed translation memory scheme assumes that there should be a single lookup method for the TM database, namely, the MetaMorpho

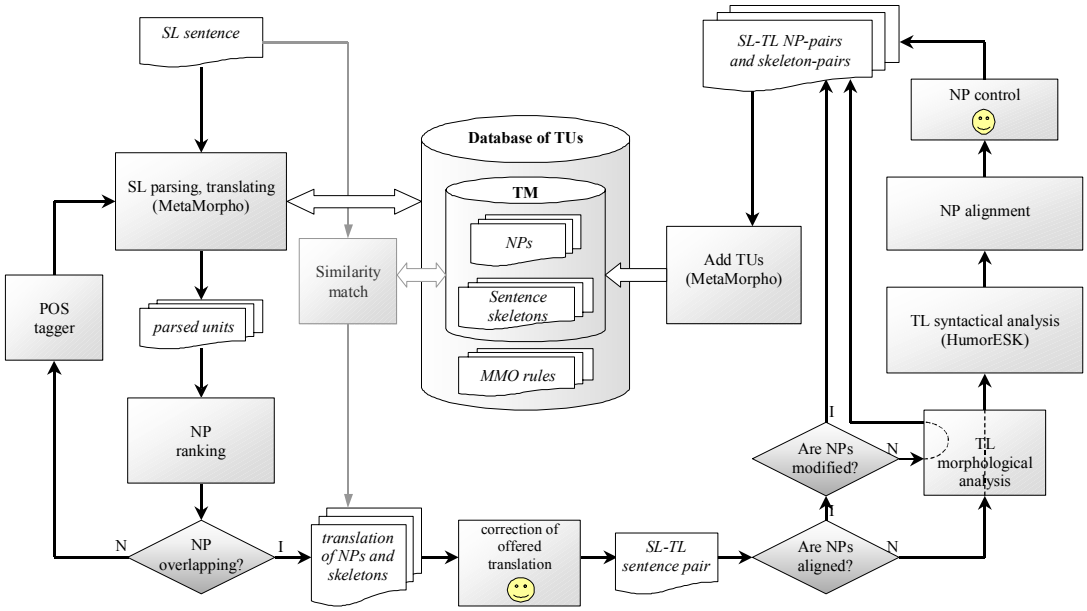

Figure 1. System Outline

translation engine [3] [4]. This poses several special requirements to translation memory operation, especially to the process of annotating new translation units, explained in Section 2. The TM system thus incorporates the machine translation engine: the core grammar lexicon is always available, and matches can still be found, even with an empty translation memory database.

The basic operation of the proposed TM engine is described below. The atomic actions are:

(1) the attempt to translate a single source segment, and

(2) adding the new translation unit (a pair of a source and target segment) to the translation memory once the human translator confirmed it. (See Figure 1.)

1.2. Attempting to translate the source segment

The proposed TM system follows the following protocol:

(1) Attempt to find an exact match. Skip all subsequent steps if found.

(2) Perform linguistic analysis (stemming, morphological analysis and parsing) on the source segment. Determine basic building blocks of the segment, and attempt to find translation for the smaller blocks. Assemble the translation according to a sentence skeleton, adjusting morpho-syntactic properties of certain words if necessary. A sentence skeleton is a pattern that includes the smaller building blocks as single symbols, and those parts of the sentence that could not be part of those building blocks.

The latter is a recursive step: the smaller building blocks undergo the same protocol. If this step is successful – the system is managed to assemble a translation using the building blocks and the skeleton –, skip all subsequent steps.

Note that some gaps may remain in the composite translation, and the operation can still finish with success. Experience with fully automatic translation (see step 3) shows that a human translation even with gaps could be more useful than a target segment translated in a fully automatic manner.

Also note that step (2) is performed entirely by the machine translation module as there are no operations outside the scope of its mechanisms. Both the smaller building blocks and the sentence skeleton can be described as underspecified (or, from another aspect, lexically constrained) grammar patterns. However, the granularity of parsing is significantly smaller here: we need to limit the number of levels in a parse tree to minimize mismatches due to parsing errors.

The discussion of building blocks becomes more specific in Section 2, where we explain the pre-defined sentence structure used in the TM annotation.

(3) If there is not an exact match, and the composition of a translation was unsuccessful, the last resort of the system is to fall back to the machine translation mechanism: it attempts to automatically translate the source segment in its entirety, and provides the user with this result.

At this point, the user receives one or more possible translations: their task is to select one, and, if necessary, adjust it to more closely correspond to the source segment.

1.3. Adding a new translation unit to the translation memory

A translation unit is a pair of one source and one target segment, assuming that the target segment is the translation of the source segment.

In a sense, adding a new translation unit is performed entirely independently from the translation of a source segment. The translation confirmed by the human translator has no connections to the result composite or automatic translations, previously provided by the TM engine. Though there are methods to track the activity of the user, and determine which basic building blocks they retained while adjusting the translation, at the moment we assume that the translation unit is entirely new.

The protocol to follow in this case is outlined below:

(1) Perform linguistic analysis of both the source and the target segment. Determine basic building blocks and segment skeletons.

(2) Perform alignment of the basic building blocks. This alignment process is similar to the one we are using for sentence-level alignment.

At this point, we have aligned pattern pairs from both the segment skeletons and the aligned building blocks, which have to be converted into the format used by the machine translation engine. Each pair is stored as a single translation rule [5] [6].

2. The annotation scheme

2.1. Requirements

According to Section 1, annotation means linguistic analysis of a source-target segment pair. As there are many ways to parse a sentence, a language-aware translation memory must use an annotation scheme that meets the following requirements:

(1) Analysis results in comparable patterns.
(2) Smaller patterns are translated in good quality even by automatic machine translation.
(3) Smaller patterns are relatively well exchangeable within sentences, i.e. they can be represented as a closed sub-structure, which, if altered, does not usually alter the larger structures within the sentences.

For this reason, we chose to use a three-level sentence structure. The obvious choices for these three levels were (1) words, (2) noun phrases (NPs), (3) sentences.
The following 3 subsections outline how this structure is obtained.

2.2. Word-level annotation

After tokenization and segmentation, each word is automatically lemmatized, and their morpho-syntactic properties are determined – in both the source and target segments. Stemming and morphological analysis is performed by MorphoLogic's Humor module, with an optional disambiguating POS-tagger integrated. In the latter case, morpho-syntactic annotation will be disambiguated.
This step results in one or more 'grammatical' patterns consisting of the morpho-syntactic category tag and the lemma of each word, the latter as a lexical constraint. Thus a basic translation pattern is obtained. An example:

Input: *The big dog saw two cats.*

```
the[DET]
big[ADV],big[ADJ]
dog[ADV],dog[N],dog[V]
saw[N],saw[V],see[V][PAST]
two[NUM]
cat[N][PL]+period[PUNCT]
```

2.3. NP annotation

In most sentence structures, arguments in verbal structures can be substituted with different phrases, as long as they take the same grammatical role. The wording and the internal structure of such arguments can be entirely different.
Arguments in English verbal constructions usually take the form of a prepositional phrase, consisting of a preposition and a noun phrase, with the latter being a replaceable construction, and the former implementing the syntactic role for the NP specific to the position within the verbal construction. Moreover, the MetaMorpho module provides fairly high-quality translations for English NPs, while the translation of larger structures can be problematic.
In Hungarian (as our primary language pair is English-Hungarian), a 'PP' takes the form of a casemarked or a post-positional NP. In the former case, the casemarker is suffixed to the head noun of the NP. Strictly speaking, there are no prepositional phrases in Hungarian because there are no prepositions. The casemarked or post-positional NP is the closest Hungarian correspondent to an English NP.
Note that the task here is to separate the 'pure' NP from the grammatical elements that determine its role in the verbal or predicative construction. The way to achieve this is language-dependent, but a language-independent framework can be implemented for it, if we specify the proper subset by using grammatical tags and features only – these are precisely the data used by the MetaMorpho engine for each node in the parse trees.
This task requires a high-precision NP chunker for both the source and the target languages. This chunker is in fact the NP-parsing mechanism integrated into the machine translation engine. However, it does not preserve the internal structure of the NPs. A shallow structure is retained only: it consists of the sequence of morpho-syntactic tags, lemmas and other features of word forms. The multi-level structure is omitted because the intermediate levels are different for each NP, therefore they are unsuitable for subsequent comparisons. Note that this process generates patterns for a translation memory database.
An example of NP chunking and TM-specific NP structures (in the MetaMorpho format):

Input: *The big dog saw two cats.*
The longest NPs found:

```
EN.NP-FULL 50 (NP 47)
DET lex="the"
ADJ lex="big"
N lex="dog" num=SG
```

```
EN.NP-FULL 282 (NP 280)
NUM lex="two"
N lex="cat" num=PL
```

2.3. Sentence skeletons

When morphological analysis is complete, there is a basic sentence-level pattern where the atomic symbols are lemmatized word forms. Once NP boundaries are identified, subsequences corresponding to NPs can be substituted with an NP symbol. In the example above, it takes the following form:

```
EN.S-FULL 363
NP 47
V lex="see" form=F2
NP 280
PUNCT lex="period"
```

Thus, NP gaps are created where virtually any other NP can be substituted. The sentence skeleton is a pattern where functional constituents like verbs, prepositions and

other non-NP words are retained as typeful lemmas, while the actual NPs are substituted with an NP gap.
Note that this is a step of abstraction: the sentence skeleton and the surface NPs are separated. If there are more than one sentence skeletons and NPs, they can be combined in any other way.
PPs, case marks and other elements that specify the role of the NP within a verbal construction, must be retained in the sentence skeleton as requirements, while the appropriate symbols and features must be marked within the NP so that they can be adjusted when inserted into a skeleton in a specific role.

2.4. NP alignment

When adding a new translation unit to the translation memory, both the source and the target segment must be analyzed. There could be an assumption that an NP in the source segment must have a translation in the target segment – this is applied when a translation is assembled by the TM engine. However, due to the nature of human translation – more precisely, the semantics-based human transfer operations – this must not be assumed when processing a translation unit confirmed by a human translator.
There are a few heuristic methods to match source NPs to target NPs: the surface features determining the arguments' roles can be matched to each other (in the source and target languages), and dictionary-based methods can be applied to content words within the NP patterns.
It is not an absolute requirement to fully align all NPs in a translation unit. It is sufficient to only add successfully aligned source-target pairs, and discard those NPs that could not be assigned a pair.

3. Methods of Evaluation

The approach described in this paper attempts at providing significantly higher quality in the translation memory field than achieved by commercial products. Evaluation must thus provide evidence for the hypothesis that linguistic annotation and an RBMT engine can provide quality improvement.
If we restrict our discussion to resource preparation, we must assess the precision of the automatic annotation. Three linguistic operations are performed:

(1) POS tagging (or morphological analysis, in the simpler case)

(2) NP chunking,

(3) NP alignment.

These operations must be assessed by comparing their results to reference values.
On a larger scale, the recall and the precision of the translation memory itself must be measured. It presents a challenge as there is little information available on measurements of existing translation memories.
The recall of a translation memory is a vague concept: when compared to the source text, it always depends on the size and contents of the TM database, which in turn depends on the individual user. It is more useful to compare the hits to the contents of the TM database: the recall of a TM system can be measured by assessing how many of the stored translation units have a chance greater than 50% being retrieved, when testing it on a sufficiently large corpus.
If we are measuring recall this way, there is an argument in favour of the language-aware TM engine: by common sense, the shorter the source segment, the more probable it is to be eventually found. It is obvious that the language-aware translation memory stores shorter segments: on the one hand, it stores substrings of the input segment, on the other hand, it stores simplified patterns such as sentence skeleton where the full variability of NPs is collapsed into a single NP gap.
The precision of a translation memory can be measured by the time the user has to spend with correcting translations offered by the system. There were no end user tests conducted as of the time of writing, however, we can again provide an argument. It is inherent to the language-aware TM to provide combined translations where the translations offered are adapted to the source segment instead of offering an entire target segment unchanged.

4. Conclusion

This paper presented a language-aware translation memory scheme, with special attention to resource processing problems, namely, those of automatic annotation of translation units.
In the authors' view, the only way to achieve substantial quality improvement in translation memories is the introduction of language-aware methods, of which the present development is an example with a lot of work still to do.
However, even at this early point, the paper could demonstrate an unconventional use of otherwise well-known techniques, and provide solid arguments in favour of the proposed scheme.

References

[1] Turcato, D. & F. Popowich (2001): 'What is Example-Based MT?' *Proceedings of the Workshop on Example-Based Machine Translation* [http://www.eamt.org/summitVIII/ workshop-papers.html]

[2] Schäler, R. (2001): 'Beyond Translation Memories.' *Proceedings of the Workshop on Example-Based Machine Translation* [http://www.eamt.org/ summit-VIII/workshop-papers.html]

[3] Prószéky (1996): 'Syntax As Meta-morphology', *Proceedings of COLING-96,* Vol.2, 1123–1126. Copenhagen, Denmark.

[4] Prószéky, G. and L. Tihanyi, (2002): 'MetaMorpho: A Pattern-Based Machine Translation Project'. *Translating and the Computer 24,* ASLIB, London.

[5] Carl, M. (2001): 'Inducing Translation Grammars from Bracketed Alignments.' *Proceedings of the Workshop on Example-Based Machine Translation* [http://www.eamt.org/summitVIII/ workshop-papers.html]

[6] Koichi Takeda (1996): 'Pattern-Based Context-Free Grammars for Machine Translation', *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics.* Santa Cruz, USA.