Translation Memory as a Robust Example-based Translation System

*Hodász, Gábor**; Grőbler, Tamás*; Kis, Balázs**

* MorphoLogic, Budapest
{grobler,kis}@morphologic.hu

** Pázmány Péter Catholic University, Budapest
Department of Information Technology
hodasz@morphologic.hu

Abstract

This paper introduces a new approach to translation memories. The proposed translation technology uses linguistic analysis (morphology and parsing) to determine similarity between two source-language segments, and attempts to assemble a sensible transltion 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.

The paper discusses the basic processes of the proposed TM. Then the underlying machine translation engine is described. This is followed by an outline of the inegrated translation support tool where the proposed TM architecture is to become a core component.

We argue that both the recall and the precision of a language-aware translation memory is higher than those of character-based systems. To this end, the paper introduces an evaluation scheme currently being prepared for use for testing the TM, followed by some estimated figures, the latter providing arguments in favour of the proposed TM scheme.

1. Introduction

Commercially available translation memories do not involve linguistic knowledge. Instead, they determine similarity between translation units on the basis of pure mathematical distance calculations, with the majority of the algorithms using fuzzy indexes. Although most such systems recognize and handle non-translatable passages (such as dates, numbers, some abbreviations), and incorporate terminology management modules as well, existing translation memory systems do not systematically treat and recognize morpho-syntactic or even syntactic similarities, if the translation units themselves are too different in terms of characters. This paper presents a proposed system addressing these particular problems, one that attempts to assemble translations from sub-sentence units stored in its database, both in the source and the target languages.

The proposed language-aware translation memory program is being implemented as a language-dependent translation support tool, first developed for the English-Hungarian language pair. The development process aims at surpassing the recall and precision of existing systems, resulting in a tool that offers translations more often, with the suggestions being closer to the desired translation. In the first version, the translation is assembled from stored translations of noun phrases and the morpho-syntactic skeleton of the source unit. The morpho-syntactic skeleton is a sequence of lemmas and morpho-syntactic parses of the words in the source unit, with a symbolic NP slot at the place of each noun phrase. This scheme may result ambiguities – an undesired phenomenon in CAT systems –, but the best translation will be a far closer approximation of the desired one than in existing systems. Therefore, in the devised user interface, the user will be able to select the best translation from multiple suggestions. However, in the subsequent revision phase – when the human translator transforms the suggestion into the desired translation –, fewer corrections will be necessary.

In order to exploit the advantages of rule-based machine translation (RBMT), the system incorporates a parser/translator module, named MetaMorpho, developed by the authors' team. Thus the suggestions are produced from stored translations and core linguistic patterns used by the MT module. If the translation memory is used extensively, new translation patterns will be created in the thousands or the tens of thousands, which, given the close connection between the TM engine and the MT module, can be recycled to enhance the quality of the MT part as well.

Section 2 of this paper discusses the basic processes of the proposed TM system, showing the characteristics of the architecture and the mechanisms. This section also describes the procedure of insert-

ing a new translation unit into the translation memory database.

Section 3 provides details on the linguistic matching of source and target segments, as well as incoming source segments and stored translation units.

Section 4 describes the MetaMorpho translation mechanism, which is an efficient blend of an example-based and a transfer-based architecture.

Section 5 provides implementation-specific details of the development project, describing the integration of the proposed translation memory tool into full-scale translation tools.

Section 6 concludes the paper by describing a proposed evaluation scheme for the translation memory, and providing arguments in favour of the proposed translation memory scheme.

2. The basic processes of the proposed TM scheme

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.

2.1. Integration of RBMT

The proposed TM 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 (Turcato et al., 1984; Schäler, 2001).

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 translation engine (Prószéky,1996; Prószéky–Tihanyi, 2002). This poses several special requirements to translation memory operation, especially to the process of annotating new translation units, explained in detail in Section 3. 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 a new translation unit (a pair of a source and target segment) to the translation memory once the human translator confirmed it. (See Figure 1.)

2.2. 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: 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 op-

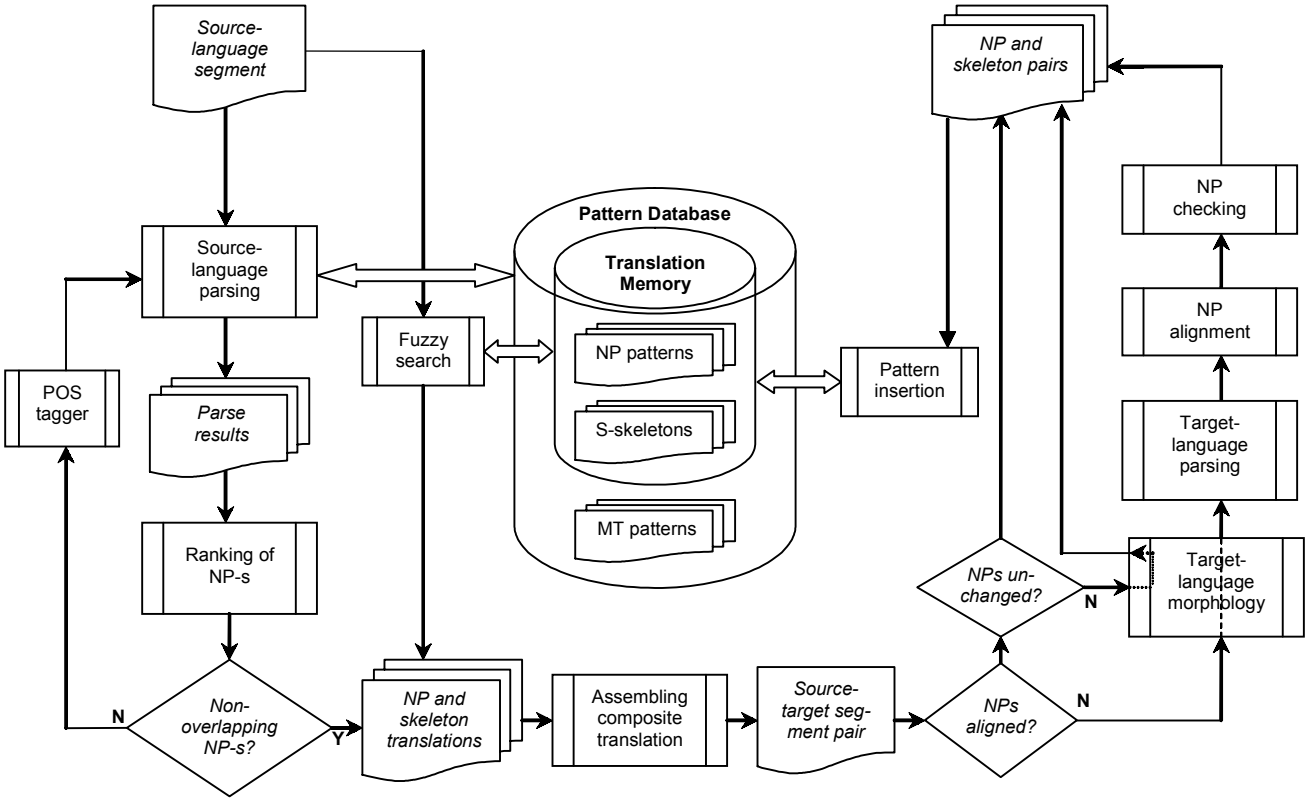

Figure 1. The basic processes of MetaMorpho TM

erations 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 3, where we explain the predefined sentence structure used in the TM enrichment.

(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. (See Section 5, where implementation-specific details are also discussed.)

At this point, the user is offered one or more possible translations: they can now select the correct one, and, if necessary, adjust it to correspond to the source segment more closely.

2.3. 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 entirely independent from the translation of a source segment. We cannot assume that the translation confirmed by the human translator is by any means related to composite or automatic translations previously offered by the TM engine. Though there are methods to track the activity of the user, and determine which basic building blocks were retained while adjusting the translation, it is more efficient to assume that the confirmed translation unit is entirely new.

Thus the TM system follows the procedure below:

(1) Performs linguistic analysis of both the source and the target segments, determining basic building blocks and segment skeletons.

(2) Performs alignment of the basic building blocks. This alignment process is similar to the one we are using for sentence-level alignment (explained in section 5).

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 (Carl, 2001; Takeda, 1996).

3. Linguistic matching

As there are many ways to parse a sentence, a language-aware translation memory must use a parsing 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 alter the larger structures within the sentences.

We decided to use a three-level sentence structure, deliberately skipping intermediate levels that would eventually occur with deep parsing. The obvious choices for these three levels were (1) words, (2) noun phrases (NPs), (3) sentences.

3.1. Word-level analysis

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.

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]
```

3.2. NP chunking

In most sentence structures, arguments in verbal structures can be substituted with different phrases, as long as they take the same grammatical role. However, 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. The casemarked or post-positional NP is the closest Hungarian correspondent to an English NP.

Here the difficulty lies with the task to separate the 'pure' NP from the grammatical elements that determine its role in the verbal or predicative construction. This procedure – as implemented – is language-dependent. However, a language-independent framework must be implemented around this process, by specifying 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. Intermediate levels are omitted because they are different for each NP, therefore unsuitable for subsequent comparisons.

An example of NP chunking and TM-specific NP structures (in the MetaMorpho formalism):

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
```

3.3. Sentence skeletons

Morphological analysis of an entire source and target segment results in a low-level sentence-level pattern where the atomic symbols are lemmatized word forms. Once NP boundaries are identified, subsequences corresponding to NPs can be substi-

tuted 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"
```

As a result, NP gaps will occur in the pattern 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.

With this step, 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. The system can thus offer a composite translation which never actually occurred in the texts stored in the translation memory.

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 constraints, 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.

3.4. NP-level 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. Rather, we elected to add successfully aligned source-target pairs only, discarding NPs that could not be assigned a pair. Discarded NPs must still be retained at least in an activity log because they provide an important resource for subsequent evaluation.

4. The underlying machine translation system

The proposed TM architecture – named MetaMorpho TM – relies on a rule-based machine translation engine: the latter is MetaMorpho itself, developed by the authors' team (Prószéky–Tihanyi, 2002).

Within the MetaMorpho system, target-language structures are created simultaneously with the process of parsing a source segment. The operation following the parsing process is a simple walkthrough of the target forest.

MetaMorpho combines the procedures of a transfer-based MT engine – using generalized rules at a high level of abstraction – and those of an example-based one, working with very specific patterns at a very low level of abstraction.

Generalized rules and highly specific patterns are all fit into the same grammar formalism. MetaMorpho uses a grammar formalism resembling and supposedly equivalent to PATR-II (Shieber et al., 1983). There cannot be defined an exact dividing line between rules and patterns. Every item of the grammar can be a generalized rule from one aspect and a specific pattern from another.

Rules – or patterns – consist of symbols with a feature structure. Every rule or pattern contains constraints on specific features of the constituent symbols. The more features constrained for a symbol, the more specific the pattern is. A typical constraint is where we specify a condition on the lemma (or lexical stem) of one or more symbols in the rule or pattern. This makes the system able to handle idiomatic expressions or other collocations more efficiently.

Typically, we designate an item of grammar a 'pattern' if one or more constituent symbols have a lexical constraint; an item without lexical constraints (i.e., where no symbols have their lemmas constrained) is rather thought of as a 'rule'. However, the grammar of this system is entirely homogeneous: the parsing algorithm does not differentiate between 'patterns' (specific rules from one aspect) and 'rules' (generalized patterns from another aspect), and they are not stored in a different way.

Bearing in mind that all grammar items have a source and the target component – because the target 'tree' is being built simultaneously with the parse forest –, and there can be different constraints on each constituent symbol within an item, we provide an example for a rule or pattern below:

```
*NX=approach+to:12
EN.NX[ct=COUNT] = N(lex=”approach”)
+ PPOBJ(lex=”to”)
```

```
HU.NX = PPOBJ[case=GEN] +
N[lex="megközelítés"]
; example: This is a really nice
approach to religion.
```

The grammar (i.e. the pattern lexicon) may contain multiple items – with very different constraints – that are applicable to the same linguistic unit. With this approach to parsing, ambiguity is an inevitable but undesired phenomenon, and one must find an efficient method to curb it. The MetaMorpho MT system is able to handle situations where one pattern or rule overrides another.

Thus a hierarchy can be defined among patterns applied to the same linguistic unit at the same grammatical level. With a pattern, one can specify a 'killer' rule, i.e. a reference to another pattern. If both patterns were applied to the same piece of the input, the former pattern will override the latter one. Overriding patterns must be manually specified; however, it is a general rule of thumb among the developers that more specific patterns should override generalized ones. Here is an example (now omitting the target-language components):

```
;NP =NP+of+NX:121
EN.NP=NP + PREP(lex="of") + NX
HU.NP=...
; example: a cup of coffee

;NP='a bottle of'+NX:122
EN.NP=DET(lex="a") +
NX(lex="bottle") + PREP(lex="of") +
      NX(num=SG,ct=UNCOUNT)
HU.NP=...
!121
; example: a bottle of wine
```

The second pattern will override the first one. This is specified by the '!121' line after the second pattern. One can also see that the general rule of thumb was followed here: the second pattern is more specific because the first part ('a bottle of') is fixed. When a sentence like 'I asked for a bottle of wine.' is parsed, only pattern 122 will be applied to the phrase 'a bottle of wine'.

Within a rule or pattern, a symbol is specified by one compulsory syntactic or morpho-syntactic label and, optionally, one or more feature constraints. With regard to the analysis methods discussed in Sections 2 and 3, one can see that both basic POS-tagged sentence patterns and sentence skeletons with NP gaps can be represented as a MetaMorpho-compliant rule or pattern.

Thus the MetaMorpho TM translation memory implements a model that can be viewed as a transition from the example-based and the rule-based approach to translation. The translation memory database stores translation units in the same formalism as the one used by the MetaMorpho MT engine.

5. Integration and implementation

The translation memory engine is only a core module of a full-featured translation tool. Without a proper database management layer, terminology management, an aligner module and an intuitive user interface, it can be used for little more than corpus processing.

So far, the TM engine and the alignment module have been implemented. The user interface is still being worked on.

Translation units for the TM are stored in a relational database – similarly to many commercially available TM systems. Terminology management is built around the same database management scheme. Domain or subject management is also in place: both the translation memory and the terminology database can be partitioned in an implicit way, by assigning each translation unit one or more position within a domain hierarchy. If the domain of the source text is then specified, the system is able to select the correct patterns (displaying domain-specific characteristics of wording and style) for the translation.

Success of the language-aware TM depends on the quality of the NLP modules implemented for both the source and the target language. The proposed TM scheme is also designed for robustness: should the linguistics-based process fail, it can fall back to a 'traditional' TM procedure. To this end, a traditional fuzzy index is also created. The fall-back protocol is the following:

(1) Attempt exact match;

(2) Try to assemble a composite translation using the TM database and the grammar of the MT engine;

(3) Attempt fuzzy match (with a rather high threshold);

(4) Attempt fully automatic translation.

If any of the above steps is successful, the translation procedure is finished. It is very important to note that human-confirmed patterns always take precedence over those in the grammar of the MT engine.

The MetaMorpho TM scheme is enriched by a proprietary alignment module, implementing a new approach to parallel text alignment. It defines a hybrid method exploiting and unifying the advantages of existing alignment strategies.

The above scheme is implemented as a closely integrated suite of tools, sharing the same basic NLP

components and using well-defined processes. Within each process, the input and the output of each phase can be exported as XML. This applies to the contents of the translation memory as well. In addition, both the alignment module and the TM engine are fully TMX-compliant.

The techniques described in this paper are language-dependent. Though the NLP modules working in the scheme are entirely data-driven, a great effort is required to produce the appropriate linguistic databases. Thus developers needed to choose one language pair to work with; with regard to Hungary's imminent accession to the EU, the proposed TM system is first developed for the English-Hungarian language pair.

6. Methods of evaluation

The proposed approach attempts at providing significantly higher quality in a translation memory than achieved by commercially available products. Evaluation must thus provide evidence for the hypothesis that linguistic annotation and an RBMT engine can provide quality improvement.

Because we state that linguistic modules will provide the desired quality improvement, we must asses the linguistic operations performed during translation and inserting of translation units:

(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.

The process of evaluating the proposed system has only begun. It is being tested on a sample English-Hungarian parallel corpus of texts on computing, with a size of approx. 1.2 million words per language, 2.5 million words altogether.

7. Conclusion

This paper presented a language-aware translation memory scheme, with a detailed description basic processes implemented in the proposed scheme, and providing an overview of integration of the TM into a larger context.

The paper states that the only way to achieve substantial improvement in translation memory quality is the integration of language-aware methods. The present development is an example of the latter approach.

The proposed scheme is an unconventional use of otherwise well-known techniques. The paper provides solid arguments in favour of the approach – with the caveat that evaluation is still in progress.

References

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]

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

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

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

Shieber, S. M., H. Uszkoreit, F. C. Pereira, J. Robinson, and M. Tyson (1983). The formalism and implementation of PATR-II. *In J. Bresnan, editor, 23 Research*

on Interactive Acquisition and Use of Knowledge. SRI International, Artificial Intelligence Center, Menlo Park, California, USA.

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

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]