[*Translating and the Computer 24*: proceedings of the…International Conference…21-22 November 2002, London (Aslib, 2002)]

MetaMorpho: A Pattern-Based Machine Translation System

Gábor PRÓSZÉKY & László TIHANYI
MorphoLogic
Késmárki u. 8.
Budapest 1118
Hungary

{proszeky,tihanyi}@morphologic.hu

Introduction

This paper describes an efficient real-time comprehension assistance and machine translation method. Combining the advantages of example-based (EBMT) and rule-based machine translation (RBMT), a new paradigm, pattern-based translation is presented. A system based on these principles that features an innovative user-friendly interface has been built. Called MetaMorpho, the system has been tested for English-Hungarian translation, and showed very promising results both in translation quality and speed.

Examples or Rules?

EBMT was suggested in the '80s as an alternative approach to machine translation than RBMT (Nagao, 1984). EBMT and RBMT can be considered the two sides of the same coin. Since then, several authors have pointed out that the performance of EBMT can be considerably improved by adding linguistic background knowledge to the system (McTait 2001; Carl 2001). The architecture is a fairly standard RBMT with a hint of EBMT found in many systems since the 1990s. The goal of our research was to find an optimum between EBMT and RBMT in terms of practical applicability: translation quality and speed. The resulting system, MetaMorpho, is closer to the example-based model than to the rule-based one. The idea is similar to Furuse & Iida's (1992) proposal, which has been later used by several other systems. EBMT is frequently considered a statistics-based, probabilistic process, whereas RBMT is often thought of as a fixed, traditional, deterministic approach. In contrast, we believe that EBMT and RBMT are just the two extremes of a generalized model. In our model, there is an arbitrary number of possible transitions between the two. Our "examples" are not necessarily directly extracted from corpora, or produced by statistical analysis. Rather we opted to build a database of structural segments, which have been generated from various sources: lexicons, dictionaries and corpora. These automatically generated items have been complemented by manually produced structures. In the following, these structural segments are referred to as patterns. These patterns which could correspond to a word or to a partially developed phrase (e.g. idiom, collocation) show some similarity to the translationally equivalent patterns used in HICATS/EJ (Kawasaki et al. 1992) and TAG (Joshi et al 1975). Their translator knowledge base consisting of patterns is mainly utilized to translate idiomatic or nonstandard expressions. Our approach treats patterns as basic tools for describing both standard and idiomatic behavior of sentences, clauses, phrases and – what's more – lexical information. If a pattern comes with all of its attributes specified, then it is an example in traditional terminology. If no attributes of a pattern are specified, then the pattern becomes a rule. Thus both examples and rules are considered as special patterns. Our approach puts the emphasis on the transitions between the two: on patterns that have some elements filled in, but which are not fully specified. A key issue in our model is how to manage these generalized patterns.

The Grammar

Apart from two distinctive features, the grammar used by our system can be termed as a simple unification grammar. The first special feature is the ability of patterns with specific conditions to override more general ones, literally "killing" the results of the latter. This is what we term the "we-know-better" principle. The second feature is the fact that the system operates with pattern pairs instead of rules: for every pattern for the source language, there is a set of corresponding patterns that determine the local structure of the target tree immediately below that node. This is called the interpretation of the source tree, which is—as opposed to the analysis—a top-down operation. For every root symbol created, a target tree is built following the structure of the source tree and only altering it within the scope of a single node at a time.

The analysis of the input is performed in three steps. First the input is segmented into "words," which will serve as the actual input sequence (terminal symbols or tokens) of the parser. The morphological analyzer determines the attributes of these symbols including their lexical form, case, conjugation etc. Second, the bottom-up parser analyzes this input sequence and if it is recognized as a correct sentence, comes up with one or more root symbols. Third, target trees are created by going down the hierarchy node by node and expanding the interpretational rules for each node. The terminal symbols at the leaves of this tree are fed into the morphological generator to produce the output sentence.

Our method is neither direct nor interlingual, and it is also opposed to transfer solutions in the sense that there is no need to "transfer" an abstract structure at any level (Fig. 1): we create our analysis with the final output in mind, and can produce the result in a very straightforward manner, without any need for complex independent transfer methods. MetaMorpho is most similar to the direct method: it does not share its features with the primitive word-by-word translations, but mainly because it is a structure-to-structure translation without having an interlingua or separate generation steps after parsing (as in the transfer method).

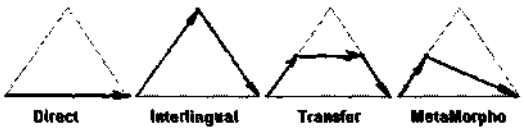


Figure 1: Strategies of MT solutions

The Translation Description Language

Every terminal and non-terminal symbol (or what is equivalent, the corresponding node in the syntax tree under construction) has a well-defined, type-specific set of features. The number of features varies between zero and a few dozen, depending on the category. These features can either take their values from a finite set of items (e.g., values of Num can be SG, PL or nil), or represent a string (e.g., the lexical form of a token).

It is important to note that no structural, semantic or lexical information is amassed in the features of symbols: the interpretation of the input is contained in the syntax tree itself, and not in the features of the node on the topmost level. As soon as parsing of the input is complete, we can create the output by a simple and deterministic interpretation of the syntax tree.

Though the grammar does have target-specific features, these properties are not dominant. Thus the grammar is general enough on the source side to allow switching to another target language with relatively little effort. It means that around 85 % of the patterns can be translated into a new target language without changing the existing grammar of the source language. There is, however, another 15 % where the relation to between the original source and the original target is different from the original source and the new target language. For example, the two sentences *My friends wanted me to be happy* and *My friend wanted me to be glad* have the same structure in the English grammars, but in an English—Hungarian translation system they should be different, because *to be happy* is an analytical structure in

Hungarian *(boldognak lenni),* but *to be glad* is a simple verb *(örülni).* The differences should be covered by that part of the English grammar which is target language dependent. Changing the source language, however, would basically require rewriting the grammar. What this means in terms of real life applications will be discussed later.
To build an efficient system and to keep the description of the linguistic database easy to manage, we created a translation description language (TDL) in XML, which consists of pattern-based rule pairs. It is this description language that defines the elements of patterns, their relationships and the parallel target (output) structure. Patterns are stored in an XML document, which is interpreted by the actual parser. In this way we clearly separate the grammar from the parser, which enables us to use the same parser with any set of patterns, e.g., different source and target language pairs.

Morpho-Syntactic Processing and the "We Know Better" Principle

Morphological analysis is performed by the Humor analyzer (Prószéky & Tihanyi, 1995), which is based on surface patterns. The basic strategy of Humor is inherently suited to parallel execution: search in the main dictionary, secondary dictionaries and affix dictionaries can be performed parallelly. Humor can also be run in a fault-tolerant mode: it overcomes simple orthographic errors and mistyping, which is crucial in the case of real-life applications. In the case of a highly inflecting language like Hungarian, where the number of word-forms for a single word is well over 100, a reliable morphological generator is a crucial part of any translation tool. The advantage of Humor is that it can be used as a generator as well as an analyzer.

A pattern can be "productive," which means it contains little or no lexical information. Such a pattern would be, for instance, **`VP=TV(vti=VT)+DOBJ`**, which describes the fact that a transitive verb and an object can form a verbal phrase. Such patterns are usually called rules. Lexicalized patterns, on the other hand, contain specific lexical information, e.g., **`VP=TV(lex="count")+PPOBJ(lex="on")`**, which expresses that *count* can have a prepositional object with *on* as its valence. Such patterns— called examples in many systems—can override more general ones, meaning that all subtrees containing symbols that were created by the eliminated pattern are deleted. If a pattern consists of a single word only, it is typically called a *lexical item.* Obviously, this is a continuum without formal distinctions depending on how specified the pattern is. Patterns overriding others are referred to as the "we know better" principle, and it is a very powerful method of incorporating lexical information into the grammar, eliminating the need for a lexicon in the traditional sense. It also enables us to disambiguate a word's sense based on its syntactical context. The "we know better" mechanism is the standard way to organize grammars today, namely, in unification grammars: rules are arranged from more specific to less specific. The difference, however, between this and our system is that in our system every symbol that is created and is not eliminated by an overriding pattern is retained even if it does not form part of a correct sentence's syntax tree. Not only is this extremely useful for debugging purposes, but it allows for "best guesses" when no interpretation for a whole sentence is found in real-life applications.

Target Language Interpretation

Patterns come in pairs, not one by one. This is what makes it possible to create an interpretation of the source tree in the target tree, rearranging its elements within the scope of a single node and its children. (Fig. 2)

Let us see how the English **`VP=TV+DOBJ`** rule with the Hungarian **`VP=DOBJ+TV`** counterpart (shown by Fig. 2.) works. The English structure consisting of a transitive verb and a direct object can be translated into Hungarian both as DOBJ+VP and VP+DOBJ. The target order depends partly on the definiteness of the original object. For example, in the sentence *Bears eat honey* the DOBJ *(honey)* is indefinite, and the verb's translation *(eszik)* can incorporate its object, allowing it to occupy the pre-verbal (focus) position:

```
EN.VP [hubeflag=TV.enbeflag, hvti=DOBJ.trobj, dopers=DOBJ.opers, ppoval=NO,
       TV.neg, TV.enbeflag, TV.tensed, TV.conj, TV.lex, TV.num, TV.pers,
       TV.pass, TV.perf, TV.cont, TV.time, TV.Modif, TV.mflag, vti=VT,
       DOBJ.radj, DOBJ.rcomp],
> TV(enbeflag=NO, vti=VT, maydefault=YES, pass=NO)
> DOBJ(uid="0", det!=NO, temp=NO)

HU.VP(focus=NO, EN.DOBJ.reqfocus=YES)
> DOBJ
> TV [focus=YES, dopers=EN.DOBJ.opers, VP.hubeflag, VP.tense, VP.e, VP.num,
      VP.pers, VP.neg, VP.vhead, VP.vhflag, VP.vhtype, VP.cond, VP.hat,
      vti=EN.DOBJ.trobj]
```

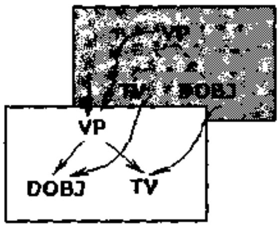
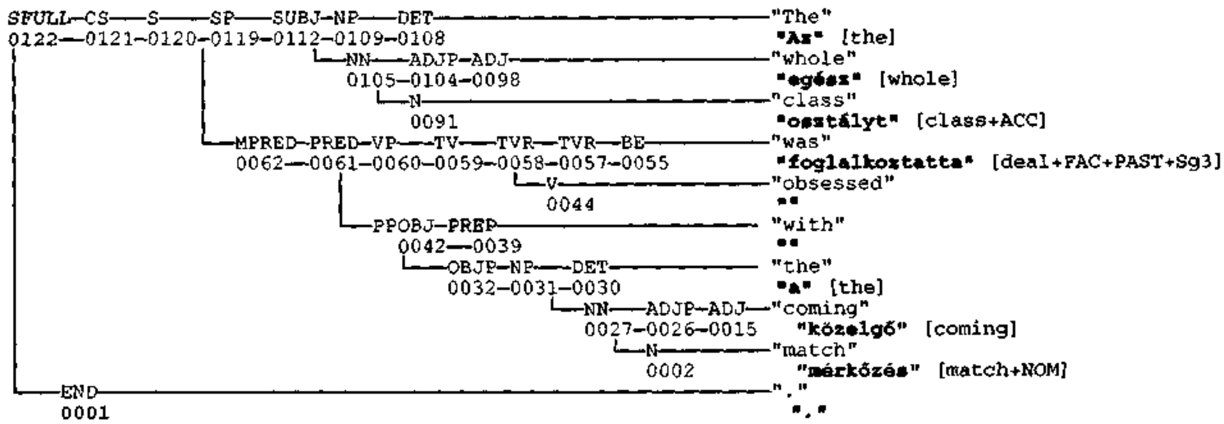


Figure 2: The MetaMorpho pair of an English source pattern (grey) and a Hungarian target pattern (white)

Three subtree interpretations are possible: permutation of the node's children, deletion of one or more children from the target tree, and insertion of terminal symbols. The interpretational rules specify the values assigned to the children's features based on the parent's features in the target tree under expansion and the corresponding features of the symbols in the source tree.

EN: *The whole class was obsessed with the coming match.*



HU: Az egész osztályt foglalkoztatta a közelgő mérkőzés.

Figure 3. Sample translation from English into Hungarian

Implementation of MetaMorpho

The principles and modules discussed above are used in a real-life application, translating from English into Hungarian. Called MetaMorpho, the system is designed to provide a translation quality that should be

high enough to serve both comprehension assistance and authoring needs. MetaMorpho modules have been written in C++.

For the purpose of syntactical description we use a pattern-based formalism, which is based on the syntax model described in (Prószéky 1996). A finite syntax is a finite set of finite descriptions of sentences. Parsing, therefore, can be realized as a sort of lexical lookup of rules. The only online operation is a unifiability check for each possible entry that matches the input sequence.

As the first step to create the basic linguistic modules, core patterns have been written and tested in the development environment. Fortunately, the number of these core type patterns for a language is surprisingly low: around thousand for English. These basic patterns serve as examples for the more specific ones. Lexical patterns, as a first step, have been derived from existing lexicons and collocation databases. This is a crucial part of the project since the building of such an inventory of lexical patterns from scratch would require several man-years.

Many systems described in the literature could not reach their full potential because it was very difficult to expand their lexicon. We have implemented a TDL rule entry environment in XML that has an easy-to-read formalism. Patterns stored in XML format can easily be converted into the parser formalism. We have to note, nonetheless, that even though our coders found the strict descriptive formalism acceptable and had extensive sample sets to work with, it required a great effort to coordinate their work and obtain coherent results. At any rate, coding was followed by thorough manual testing.

As opposed to the low number of core patterns, the number of lexical patterns is well in the hundreds of thousands. Assuming that one pattern can be stored in 100 bytes and a typical PC can be expected to have 100 megabytes of free RAM, the maximum number of patterns that can be used by the system is around 1 million.

MetaMorpho can presently be used with two user interfaces: a typical input-output based one for the developer and a special interface for end users. For the latter we use the technology of MoBiMouse (Clark 2000), which doesn't require more user activity than positioning the mouse pointer over the sentence to be translated, and the translation appears in a pop-up window.

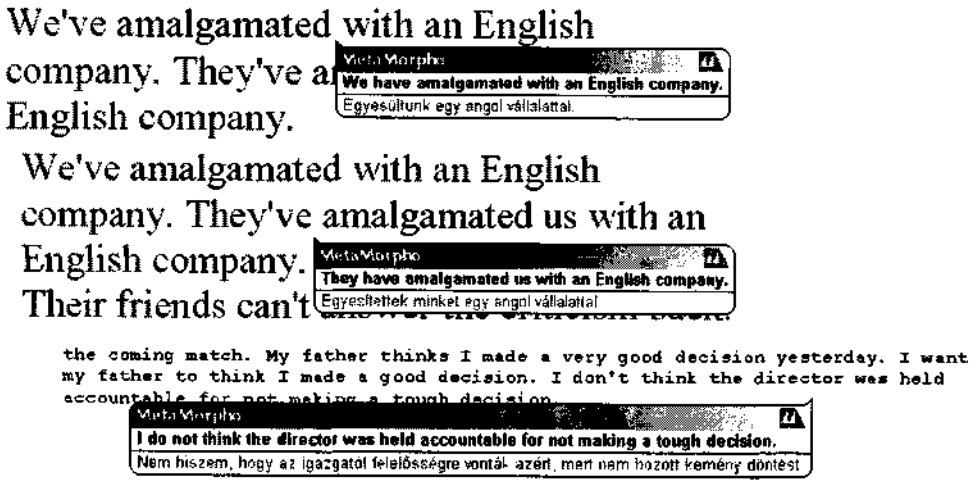


Figure 4: Sample translations of the English-Hungarian system using the "just point" method

Future Work and Conclusion

We are planning to increase MetaMorpho's fault tolerance by combining it with our spell-checker (Prószéky 1995) in the near future. The method is also expected to support professional human translators

by expanding the abilities of translation memories. Taking advantage of MetaMorpho's ability to translate incomplete sentences (structures only), we could translate the variant part of the sentence and thus improve the efficiency of translation memories. MetaMorpho currently fetches only the first target equivalent from the lexical patterns. This could be changed by reordering the target equivalents according the context. We are working on a topic recognition module running the same way as language identifier programs do but identifying the sublanguage (business, medicine, sport, etc.) having a well recognizable terminology within a single language, such as English. Once the topic identifier determines the topic, it sets MetaMorpho's lexical patterns' target equivalent ranking accordingly.

MetaMorpho is an innovative system in many ways. It is based on a uniform description of lexical and structural information. Lexically underspecified patterns correspond to rules of other systems, lexically specified patterns are partially lexical templates (idioms, multi-word expressions, terminologies) or simple lexical entries (words, morphemes). The main reason why this idea has not yet been applied is memory limitations of the computers of the previous decades. Today we can count of min. 100 MB memory, that is several hundreds of thousands patterns. When today's most famous MT systems were born, markup languages did not exist. XML technology and widely used inheritance-based systems help us to implement the above ideas very effectively.

MetaMorpho represents a generalization of the EBMT model, using equivalents of long structures (patterns) whenever possible for the translation and applying a "we know better" approach to get the best fitting translation. Parsing is performed by a system that combines source language parsing and target language interpretation in one single task.

MetaMorpho is also an open development platform: developers in need for a similar translation tool in their preferred language pair (and in possession of necessary resources) may contact the present developers and contribute to the future improvement of the system.

MetaMorpho features MoBiMouse, a user-friendly pop-up window interface to provide comprehension assistance to real users in real time. If the input is grammatically correct, the system should provide correct translation, and if the input cannot be analyzed, the system should provide the translation of all the separate correct structures it can identify. MetaMorpho works according to the comprehension assistance principles: when the user wants to understand some text in a foreign language, sometimes even to look up a single word in a dictionary can be of great help (Nerbonne et al 1997), but MetaMorpho can provide translations of long structural parts of the text, or, in most cases, for whole sentences.

Acknowledgements

Thanks to Gábor Ugray, Balázs Kis, András Földes and István Endrédy for their contribution to MetaMorpho's technical and linguistic implementation.

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]

Clark, B. (2000) MoBiMouse, the first no-click translation tool. *Journal of Language and Documentation*

Joshi, Aravind K., Leon Levy, & Masako Takahashi (1975). Tree Adjoining Grammars. *Journal of Computer and System Sciences*, 10(1). 136-163.

Kawasaki, Z, F. Yamano, N. Yamasaki (1992) Translator Knowledge Base for Machine Translation Systems. *Machine Translation* 6(4), 265-278

McTait, K. (2001) Linguistic Knowledge and Complexity in an EBMT System Based on Translation Patterns. *Proceedings of the Workshop on Example-Based Machine Translation* [http://www.eamt.org/summitVIII/workshop-papers.html]

Nagao, M. (1984). A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In: A. Elithorn and R. Banerji (eds.): *Artificial and Human Intelligence*, pp 173-180, North Holland, Amsterdam.

Nerbonne, John, Lauri Karttunen, Elena Paskaleva, Gábor Prószéky, Tiit Roosmaa (1997) Reading More into Foreign Languages. *Proceedings of the 5th Conference on Applied Natural Language Processing*, 135-138. Washington, USA

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

Prószéky, G. & B. Kis: (1999a) *Szdmitogeppel emberi nyelven.* SZAK, Bicske.

Prószéky, G. & B. Kis (1999b) Agglutinative and Other (Highly) Inflectional Languages. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 261-268. College Park, Maryland, USA (1999)

Prószéky, G. & Tihanyi L. (1995). Humor: High-Speed Unification Morphology and Its Applications for Agglutinative Languages. *La tribune des industries de la langue*, No. 10. 28-29, OFIL, Paris.

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

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]