The MetaMorpho translation system

Attila Novák, László Tihanyi and Gábor Prószéky
MorphoLogic
Orbánhegyi út 5, Budapest 1126, Hungary
{novak,tihanyi,proszeky}@morphologic.hu

Abstract

In this article, we present MetaMorpho, a rule based machine translation system that was used to create MorphoLogic's submission to the WMT08 shared Hungarian to English translation task. The architecture of MetaMorpho does not fit easily into traditional categories of rule based systems: the building blocks of its grammar are pairs of rules that describe source and target language structures in a parallel fashion and translated structures are created while parsing the input.

1 Introduction

Three rule-based approaches to MT are traditionally distinguished: direct, interlingua and transfer. The direct method uses a primitive one-stage process in which words in the source language are replaced with words in the target language and then some rearrangement is done. The main idea behind the interlingua method is that the analysis of any source language should result in a language-independent representation. The target language is then generated from that language-neutral representation. The transfer method first parses the sentence of the source language. It then applies rules that map the lexical and grammatical segments of the source sentence to a representation in the target language.

The MetaMorpho machine translation system developed at MorphoLogic (Prószéky and Tihanyi, 2002), cannot be directly classified in either of the above categories, although it has the most in common with the transfer type architecture.

2 Translation via immediate transfer

In the MetaMorpho system, both productive rules of grammar and lexical entries are stored in the form of patterns, which are like context-free rules enriched with features. Patterns may contain more-or-less underspecified slots, ranging from general productive rules of grammar through more-or-less idiomatic phrases to fully lexicalized items. The majority of the patterns (a couple of hundreds of thousands in the case of our English grammar) represent partially lexicalized items.

The grammar operates with pairs of patterns that consist of one source pattern used during bottom-up parsing and one or more target patterns that are applied during top-down generation of the translation. While traditional transfer and interlingua based systems consist of separate parsing and generating rules, in a MetaMorpho grammar, each parsing rule has its associated generating counterpart. The translation of the parsed structures is already determined during parsing the source language input. The actual generation of the target language representations does not involve any additional transfer operations: target language structures corresponding to substructures of the source language parse tree are combined and the leaves of the resulting tree are interpreted by a morphological generator. We call this solution "immediate transfer" as it uses no separate transfer steps or target transformations.

The idea behind this architecture has much in common with the way semantic compositionality was formalized by Bach (1976) in the from of his rule-to-rule hypothesis, stating that to every rule of syntax that combines constituents into a phrase pertains a corresponding rule of semantics that

Proceedings of the Third Workshop on Statistical Machine Translation, pages 111–114,
Columbus, June 2008. ©2008 Association for Computational Linguistics

combines the meanings of the constituents. In the case of phrases with compositional meaning, the pair of rules of syntax and semantics are of a general nature, while in the case of idioms, the pair of rules is specific and arbitrary. The architecture implemented in the MetaMorpho system is based on essentially the same idea, except that the representation built during analysis of the input sentence is not expressed in a formal language of some semantic representation but directly in the human target language of the translation system.

3 System architecture

The analysis of the input is performed in three stages. First the text to be translated is segmented into sentences, and each sentence is broken up into a sequence of tokens. This token sequence is the actual input of the parser. Morphosyntactic annotation of the input word forms is performed by a morphological analyzer: it assigns morphosyntactic attribute vectors to word forms. We use the Humor morphological system (Prószéky and Kis, 1999; Prószéky and Novák, 2005) that performs an item-and-arrangement style morphological analysis. Morphological synthesis of the target language word forms is performed by the same morphological engine.

The system also accepts unknown elements: they are treated as strings to be inflected at the target side. The (potentially ambiguous) output of the morphological analyzer is fed into the syntactic parser called Moose (Prószéky, Tihanyi and Ugray, 2004), which analyzes this input sequence using the source language patterns and if it is recognized as a correct sentence, comes up with one or more root symbols on the source side.

Every terminal and non-terminal symbol in the syntactic tree under construction has a set of features. The number of features is normally up to a few dozen, depending on the category. These features can either take their values from a finite set of symbolic items (e.g., values of case can be `INS`, `ACC`, `DAT`, etc.), or represent a string (e.g., `lex="approach"`, the lexical form of a token). The formalism does not contain embedded feature structures. It is important to note that no structural or semantic information is amassed in the features of symbols: the interpretation of the input is contained in the syntactic tree itself, and not in the features of the node on the topmost level. Features are used to express constraints on the applicability of patterns and to store morphosyntactic valence and lexical information concerning the parsed input.

More specific patterns (e.g. *approach to*) can override more general ones (e.g. *approach*), in that case subtrees containing symbols that were created by the general pattern are deleted. 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 syntactic tree. Each pattern can explicitly override other rules: if the overriding rule covers a specific range of the input, it blocks the overridden ones over the same range. This method can be used to eliminate spurious ambiguities early during analysis.

When the whole input is processed and no applicable patterns remain, translation is generated in a top-down fashion by combining the target structures corresponding to the source patterns constituting the source language parse tree.

A source language pattern may have more than one associated target pattern. The selection of the target structure to apply relies on constraints on the actual values of features in the source pattern: the first target pattern whose conditions are satisfied is used for target structure generation. To handle complicated word-order changes, the target structure may need rearrangement of its elements within the scope of a single node and its children. There is another technique that can be used to handle word order differences between the source and the target language. A pointer to a subtree can be stored in a feature when applying a rule at parse time, and because this feature's value can percolate up the parse-tree and down the target tree, just like any other feature, a phrase swallowed somewhere in the source side can be expanded at a different location in the target tree. This technique can be used to handle both systematic word order differences (such as the different but fixed order of constituents in possessive constructions: *possession of possessor* in English versus *possessor possession + possessive suffix* in Hungarian) and accidental ones (such as the fixed order of subject verb and object in English, versus the "free" order of these constituents in Hungarian[1]).

Unlike in classical transfer-based systems, however, these rearrangement operations are al-

[1] In fact the order is determined by various factors other than grammatical function.

ready determined during parsing the source language input. During generation, the already determined rearranged structures are simply spelled out. The morphosyntactic feature vectors on the terminal level of the generated tree are interpreted by the morphological generator that synthesizes the corresponding target language word forms.

The morphological generator is not a simple inverse of the corresponding analyzer. It accepts many alternative equivalent morphological descriptions of each word form it can generate beside the one that the corresponding analyzer outputs.

4 The rule database

The rules used by the parser explicitly contain all the features of the daughter nodes to check, all the features to percolate to the mother node, all the features to set in the corresponding target structures and those to be checked on the source language structure to decide on the applicability of a target structure. The fact that all this redundant information is present in the run-time rule database makes the operation of the parser efficient in terms of speed. However, it would be very difficult for humans to create and maintain the rule database in this redundant format.

There is a high level version of the language: although it is not really different in terms of its syntax from the low-level one, it does not require default values and default correspondences to be explicitly listed. The rule database is maintained using this high level formalism. There is a rule converter for each language pair that extends the high-level rules with default information and may also create transformed rules (such as the passive version of verbal subcategorization frames) creating the rule database used by the parser.

Rule conversion is also necessary because in order to be able to parse a free word order language like Hungarian with a parser that uses context free rules, you need to use run time rules that essentially differ in the way they operate from what would be suggested by the rules they are derived from in the high level database. In Hungarian, arguments of a predicate may appear in many different orders in actual sentences and they also freely mix with sentence level adjuncts. This means that a verbal argument structure of the high level rule database with its normal context free rule interpretation would only cover a fraction of its real world realizations. Rule conversion effectively handles this problem by converting rules describing lexical items with argument structures expressed using a context free rule formalism into run time rules that do not actually combine constituents, but only check the saturation of valency frames. Constituents are combined by other more generic rules that take care of saturating the argument slots. This means that while the high level and the run time rules have a similar syntax, the semantics of some high level rules may be very different from similar rules in the low level rule database.

5 Handling sentences with no full parse

The system must not break down if the input sentence happens not to have a full parse (this inevitably happens in the case of real life texts). In that case, it reverts to using a heuristic process that constructs an output by combining the output of a selected set of partial structures covering the whole sentence stored during parsing the input. In the MetaMorpho terminology, this is called a “mosaic translation”. Mosaic translations are usually suboptimal, because in the absence of a full parse some structural information such as agreement is usually lost. There is much to improve on the current algorithm used to create mosaic translations: e.g. it does not currently utilize a statistical model of the target language, which has a negative effect on the fluency of the output. Augmenting the system with such a component would probably improve its performance considerably.

6 Motivation for the MetaMorpho architecture

An obvious drawback of the architecture described above compared to the interlingua and transfer based systems is that the grammar components of the system cannot be simply reused to build translation systems to new target languages without a major revision of the grammar. While in a classical transfer based system, the source language grammar may cover phenomena that the transfer component does not cover, in the MetaMorpho architecture, this is not possible. In a transfer based system, there is a relatively cheaper way to handle coverage issues partially by augmenting only the source grammar (and postponing creation of the corresponding transfer rules). This is not an option in the MetaMorpho architecture.

The main motivation for this system architecture was that it makes it possible to integrate machine translation and translation memories in a natural way and to make the system easily extensible by the user. There is a grammar writer's workbench component of MetaMorpho called Rule Builder. This makes it possible for users to add new, lexical or even syntactic patterns to the grammar in a controlled manner without the need to recompile the rest, using an SQL database for user added entries. The technology used in Rule-Builder can also be applied to create a special combination of the MetaMorpho machine translation tool and translation memories (Hodász, Grőbler and Kis 2004).

Moreover, existing bilingual lexical databases (dictionaries of idioms and collocations) are relatively easy to convert to the high level rule format of the system. The bulk of the grammar of the system was created based on such resources. Another rationale for developing language pair specific grammars directly is that this way distinctions in the grammar of the source language not relevant for the translation to the target language at hand need not be addressed.

7 Performance in the translation task

During development of the system and its grammar components, regression testing has been performed using a test set unknown to the developers measuring case insensitive BLEU with three human reference translations. Our usual test set for the system translating from Hungarian to English contains 274 sentences of newswire text. We had never used single reference BLEU before, because, although creating multiple translations is expensive, single reference BLEU is quite unreliable usually producing very low scores especially if the target language is morphologically rich, like Hungarian.

The current version of the MetaMorpho system translating from Hungarian to English has a BLEU score of 22.14 on our usual newswire test set with three references. Obtaining a BLEU score of 7.8 on the WMT08 shared Hungarian to English translation task test set was rather surprising, so we checked single reference BLEU on our usual test set: the scores are 13.02, 14.15 and 16.83 with the three reference translations respectively.

In the end, we decided to submit our results to the WMT08 shared translation task in spite of the low score. But we think, that these figures cast doubts on the quality of the texts and reference translations in the test set, especially in cases where both the English and the Hungarian text were translated from a third language, so we think that the scores on the WMT08 test set should be evaluated only relative to other systems' performance on the same data and the same language pair.

References

Emmon Bach. 1976. An extension of classical transformational grammar. In Saenz (ed.) *Problems of Linguistic Metatheory: Proceedings of the 1976 Conference,* 183–224. East Lansing, MI: Michigan State University.

Gábor Hodász, Tamás Grőbler and Balázs Kis. 2004. Translation memory as a robust example-based translation system. In Hutchins (ed.), 82–89.

John Hutchins (ed.) *Broadening horizons of machine translation and its applications.* Proceedings of the 9th EAMT Workshop, 26–27 April 2004. La Valletta: Foundation for International Studies.

Gábor Prószéky and Balázs Kis. 1999. Agglutinative and other (highly) inflectional languages. In Robert Dale & Kenneth W. Church (eds.) *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 261–268. Morristown, NJ: Association for Computational Linguistics.

Gábor Prószéky and Attila Novák. 2005. Computational Morphologies for Small Uralic Languages. In: A. Arppe, L. Carlson, K. Lindén, J. Piitulainen, M. Suominen, M. Vainio, H. Westerlund, A. Yli-Jyrä (eds.): *Inquiries into Words, Constraints and Contexts Festschrift in the Honour of Kimmo Koskenniemi on his 60th Birthday,* 116–125. Gummerus Printing, Saarijärvi/CSLI Publications, Stanford.

Gábor Prószéky and László Tihanyi. 2002 MetaMorpho: A Pattern-Based Machine Translation System. In: *Proceedings of the 24th 'Translating and the Computer' Conference*, 19–24. ASLIB, London, United Kingdom.

Gábor Prószéky, László Tihanyi and Gábor Ugray. 2004. Moose: A robust high-performance parser and generator. In Hutchins (ed.), 138–142.