EBMT: Example Based Machine Translation

Alberto Manuel Brandão Simões — ambs@di.uminho.pt

Example-Based Machine Translation (EBMT) is a ***statistical based method*** for ***Machine Translation***. The idea behind this approach to MT is that if we take all the translations already done in the world, probably we will need to translate just a few more sentences, or just translate some words.

So, EBMT uses a ***database of translations***, called examples (this database is in many cases just parallel corpora, but its complexity can raise to more complex data structures like syntactic trees), where it will search for translations similar to the one being performed. After extracting these examples, they will be merged in a full and, hopefully, correct translation[7].

Although based on a simple concept, this translation method is one of the main research areas in MT. New techniques are appearing in the ***storing methods*** with examples generalization, ***matching*** using word morphological information and ***adapting*** the examples using rules. In this document we will try to discuss each one of these steps of MT and how previous work done with word-alignment of parallel corpora can be useful for this task.

We propose to develop ***prototypes*** for some pieces of this tool, in special regarding Portuguese ↔ English translation, and research what techniques for each one of the four components can be more effective for the ***Portuguese language***. Finally, an ***evaluation process*** will be defined so we can discuss the results of applying the different techniques during EBMT.

Examples Database

Data Type

It is important to specify a correct way to store examples. Also, the definition of example is not trivial.

$$\begin{aligned} tmdb &= transMemory : tm \times \\ &\quad transPattern : tp \times \\ &\quad dictionary : dic \\ tm &= alignUnit \overset{m}{\leftrightarrow} alignUnit \\ tp &= pattern \overset{m}{\leftrightarrow} pattern \\ dic &= word \rightharpoonup wordInfo \\ wordInfo &= trans : word \rightharpoonup real \times \\ &\quad occur : int \end{aligned}$$

The traditional definition of a translation memory database (*tmdb*) include just pairs of sentences to be used directly (in the called Memory Based Machine Translation). In this definition we call them *alignUnits* because they do not need to be full clauses.

Taking sentences together it is possible to remove common parts and create place-holders. To these generalized align units we call patterns[2].

Finally, if we divide more and more the alignUnits, we get pairs of words. The translation database should include a dictionary[8] (in this case, a probabilistic translation dictionary[6]) for word-to-word translations.

Generalization of Examples

Given the scarcity of parallel corpora, the examples should be generalized and processed in order to generate more examples.

The first simple way is to detect translation units. This can be done using chunkers, syntactic sentence information and multi-term or compound verbs information.

Splitting sentences into words is similar to the corpora word alignments. A tool like NATools[4] can be used to extract this alignment which results in a probabilistic translation dictionary.

Patterns generalize examples so we can use them in sentences with similar constructs and words, but that are not exactly the same. Generalizing techniques[2] include:

- word chunking: detection of words used in similar contexts;
- entity name recognition;
- part-of-speech tagging, to create word classes;
- generalization removing different portions of similar sentences

More than just storing examples, it is also important to find ways to create them. Although there is freely available parallel corpora, sometimes they do not exist in the required quantities. One way to solve that problem is using techniques to extract examples from the Internet [1].

$$\begin{aligned} & translate : sentence \times tmdb \longrightarrow sentence \\ & translate(s, db) \stackrel{\text{def}}{=} \\ & \quad \underline{let}\ l1 = split(s) \\ & \quad\quad l2 = <match(db, x) \mid x \in l1> \\ & \quad \underline{in}\ \ recombine(l2) \end{aligned}$$

Splitting the Source

It is very improbable that we find the full sentence the system is trying to translate as an example. So, the system should split the sentence in smaller segments.

$$\begin{aligned} & split : sentence \longrightarrow segment^{\star} \\ & split(s) \stackrel{\text{def}}{=} \\ & \quad \ldots \end{aligned}$$

The way sentences should be split is not trivial. We have to split the sentence in sequences that exist in the examples database. Also, these extract do not need to be full phrases as linguists see them: just simple word sequences that pulled together form the original sentence. This splitting can be done in using different techniques:

- use chunkers to create word segments;
- use syntactic sentence information — using annotated corpora (not necessarily parallel one), like CETEMPúblico[3].
- multi-term and compound verbs detection — specially if we use a similar technique to create examples.

This process should be iterative so we can match different sequences of words against the examples and use the ones that give better results. It is specially important because some methods to split the sentences can divide constituents that should be translated together.

Matching Fragments

With the sentence segments, we need to find their translations on the examples. For that, we will match those segments against the examples. This process should be fast, but versatile. The match should be done in a fuzzy like approach.

$$\begin{aligned} & match : tmdb \times segment \longrightarrow segment \\ & match(db, s) \stackrel{\text{def}}{=} \\ & \quad \ldots \end{aligned}$$

For each sequence of words being matched we must check how much probable that translation is. For that, the probabilistic dictionary can be used. This way, we can return a value to the splitter module which can try with another word sequences (bigger or smaller) which can give better results.

Also, the sequence of words can match a pattern. In that case, the place holder portion should be matched again against the examples database.

The matching step can also use some semantic knowledge. Word-sense disambiguation both while matching sentences (matching only if their sense is the same) and choosing a translation from the dictionary (choosing a translation with the same sense) can be very helpful.

Recombining Translation Fragments

After the examples segments extraction, they need to be recombined. Unfortunately, to recombine the segment translations is not so simple as their concatenates. In fact, some phenomena like gender and number issues as well as words order should be analyzed.

$$\begin{aligned} & recombine : segment^{\star} \longrightarrow sentence \\ & recombine(s) \stackrel{\text{def}}{=} \\ & \quad \ldots \end{aligned}$$

Normally this step is performed using a set of rules together with a morphological analyzer[5]. The rules can be of different complexity accordingly with the problem it tries to solve.

Some of these rules can be learned from existing parallel corpora. Some other rules (like gender and number concordance) can be written in a general way using morphological analyzers.

[1] José João Almeida, Alberto Manuel Simões, and José Alves Castro. Grabbing parallel corpora from the web. In *Sociedade Española para el Procesamiento del Lenguaje Natural*, 29, pages 13–20, Sep. 2002.

[2] Ralf D. Brown. Transfer-rule induction for example-based translation. In Michael Carl and Andy Way, editors, *Workshop on Example-Based Machine Translation*, pages 1–11, September 2001.

[3] Paulo Alexandre Rocha and Diana Santos. Cetempúblico: Um corpus de grandes dimensões de linguagem jornalística portuguesa. In Maria das Graças Volpe Nunes, editor, *V Encontro para o processamento computacional da língua portuguesa escrita e falada (PROPOR 2000)*, pages 131–140, Atibaia, São Paulo, November 2000.

[4] Alberto M. Simões and J. João Almeida. Natools – a statistical word aligner workbench. *SEPLN*, Sep. 2003.

[5] Alberto Manuel Simões and José João Almeida. `jspell.pm` – um módulo de análise morfológica para uso em processamento de linguagem natural. In *Actas da Associação Portuguesa de Linguística*, pages 485–495, 2001.

[6] Alberto Manuel Brandão Simões. Parallel corpora word alignment and applications. Master's thesis, Escola de Engenharia - Universidade do Minho, 2004.

[7] Harold Somers. Review article: Example based machine translation. *Machine Translation*, 14(2):113–157, 1999.

[8] Davide Turcato and Fred Popowich. What is example-based machine translation? In Michael Carl and Andy Way, editors, *Workshop on Example-Based Machine Translation*, pages 43–48, September 2001.