Intelligent Information Systems 9999
ISBN 666-666-666, pages 1–8

A new tool for the bilingual text aligning at the sentence level

Krzysztof Jassem and Jarosław Lipski

Adam Mickiewicz University, Poznań, Poland

Abstract

A new bilingual text aligning tool has been developed, which expands the possibilities offered by the Gale-Church algorithm and the Moore algorithm. So far, the tool has been used in a commercial translation memory system and a student statistical MT project. The modular architecture of the program allows for flexibility – the tool may be adapted to various needs. The program is open for further development as free / open source software. The paper discusses potential fields of use and further development of the program.

Keywords: alignment, segmentation

1 Definitions

There are several terms in this paper that we would like to define to avoid confusion[1].

Segment is a fragment of a text document allocated on the basis of a given criterion. Depending on use, a segment may be a word, a phrase, a sentence, a line, a paragraph or even a whole document. *Segmentation* is a process of separating a text into segments, during which no characters may be omitted.

Source text is the first element of the ordered pair of texts, not necessarily the text that was the source for translation. *Target text* is the second element of the ordered pair of texts. *Source segment* is a segment allocated in the source text. *Target segment* is a segment allocated in the target text.

Mapping is an ordered pair of lists: the list of source segments and the list of target segments. For example, a 2-1 mapping is a pair: <a two-segment source list; a one-segment target list>.

Alignment is the list of mappings; the source and target segments in these mappings appear in the same order as they occur in corresponding texts, but some of the segments may be omitted.

Bilingual text aligning at the sentence level is an automated process that results in an alignment between two texts in which segments are defined as sentences.

[1] The definitions presented here are simplified to decrease the paper length. For more accurate definitions see Lipski 2007, chapter 1.

2 Goals

The main goal for the *mALIGNa* project was the use of the aligning tool in *heTMan*. *heTMan* is a translation memory management system that contains a module of translation by analogy called *ANAT*[2]. Its engine is built into a Machine Translation system *Translatica*[3]. *heTMan* is also a crucial part of *TranslAide*[4], a Computer-Assisted Translation system under development. The *mALIGNa* is used in *hetMan* for obtaining the translation memory on the basis human-translated texts stored in various formats, including *Microsoft Word .DOC* format.

mALIGNa was also applied in the university study project called *Exprimo*[5], which aimed at the creation of a complete statistical translation system. Polish, German and English corpora were gathered by a web crawler[6] from the multilingual internet sites, and *mALIGNa* was supposed to handle multiple input formats such as *HTML* and *TXT*.

Another goal was to improve the existing algorithms. The program implements variations of the Gale-Church algorithm[7] and the Moore algorithm[8]. The algorithms are re-written in such a way so that linguistic knowledge could be used in order to obtain more accurate results for smaller texts.

The program interoperates with other available linguistic tools such as *Giza*[9].

Our solution was optimized for the sake of corpora including Polish texts.

Why not just use one of reference implementations[10] provided for Gale-Church and Moore alignment algorithms? First of all, we needed to have more consistent, expandable and portable library, which supports various input and output formats. The reference implementations were more of proof of concepts than final products. We needed a complete platform to test new ideas and the existing codes were hard to use due to monolithic design, complication, old-fashioned programming style[11], not enough configuration options and not enough important components[12].

The modern and more complete bilingual text aligners exist, but most of them are commercial and have prohibitive licenses. One of the notable exceptions is *hunalign*[13], which is free. We intend to share our achievements with the informa-

[2] The system translates sentences via the analogy principle. See Jassem and Gintrowicz 2007.

[3] Translatica is a product of Poleng company. See *http://www.poleng.pl*.

[4] TranslAide is a product of Poleng company. See *http://www.poleng.pl*.

[5] Exprimo is a simple statistical machine translation system, created as a student project by Jacek Gintrowicz, Wojciech Misiurka, Monika Rosińska, Michał Szymański and Jarek Lipski. See *http://mt.wmid.amu.edu.pl/*.

[6] See Rosińska 2007.

[7] Gale-Church algorithm is one of the first successful length-based algorithms. See next section and Gale and Church 1991.

[8] Moore algorithm is a modern alignment algorithm that makes use of word correspondences. See next section and Moore 2002.

[9] Giza is a tool for calculating statistical translation models.
See *http://www.fjoch.com/GIZA++.html*.

[10] The reference implementations are provided with Gale-Church and Moore algorithms. See Gale and Church 1991 and *http://research.microsoft.com/users/bobmoore/*.

[11] For Gale-Church algorithm the programming language is non standard-compliant *C*, and for Moore algorithm it is *Perl*.

[12] Such as parsing of different formats, segmentation. See the alignment process below.

[13] *hunalign* is a bilingual text aligner at the sentence level that implements variation of the Moore algorithm and can make use of dictionaries. See *http://mokk.bme.hu/resources/hunalign*.

tion society as well. Therefore we provide our application as free / open source software, hoping that it will be further developed and the results will be helpful for other applications. The project is licensed under a MIT-style license[14], the source code can be downloaded from the project page on *Sourceforge.net – http://sourceforge.net/projects/align*.

3 Used Methodology

3.1 Implemented algorithms

We re-implemented the Gale-Church algorithm and the Moore algorithm basing on papers and reference implementations. The resulting implementation is:

- more robust – covers more phases of alignment process, handles more file formats,
- more modular – each processing phase can be customized,
- better coded – richly commented code written in widely used Java programming language,
- easier to maintain and extend – thanks to object-oriented methodology.

3.1.1 Gale-Church algorithm

The Gale-Church algorithm takes advantage of the fact that longer sentences in one language tend to be translated into longer sentences in the other language. A probability is assigned to each proposed mapping, based on the ratio of the sum of lengths[15] of source segments and the sum of lengths of target segments in this mapping. This probability measure is used to find the alignment which has the maximum total probability calculated as the product of probabilities of all its mappings. The algorithm is called the length-based alignment algorithm, as it relies only on segment lengths and not on segment contents.

3.1.2 Moore algorithm

In the Moore algorithm the probability of each proposed mapping is calculated on the basis of three linguistic models: the translation model[16], the source language model and the target language model[17]. Similar approach is usually taken in Statistical Machine Translation (SMT)[18]. Contrary to most SMT algorithms the Moore algorithm does not need external models. The models are calculated from

[14] See *http://www.opensource.org/licenses/mit-license.php*.

[15] In Gale-Church algorithm the segment length is measured in characters. It can be also measured in words.

[16] Also called probabilistic dictionary, which is a type of dictionary where for each word there is a list of translations with probabilities assigned to them. See Lipski 2007, chapter 1.

[17] The source and target language models used by Moore algorithm are very simple unigram models. The probability of a sentence in these models is a product of frequencies of each individual word in this sentence in the given language.

[18] "Statistical Machine Translation is a machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual text corpora" (from Wikipedia). For more details see Brown et al. 1993, Knight 1999.

the most probable mappings in the alignment of the input texts, obtained via a length-based alignment algorithm, which is executed first.

The aligning process in the Moore algorithm can be divided in four phases:

1. The length-based algorithm similar to Brown[19] algorithm is used to obtain initial alignment.
2. The highest probability alignment is selected from the initial alignment. Only one-to-one, high probability mappings are preserved.
3. On the basis of the alignment, translation and language models are trained according to the IBM Model 1[20].
4. The final aligning is executed. In this phase the product of the SMT probability and the length-based probability is used as a measure of each proposed mapping. The searched best alignment is the alignment which maximizes the expected number of individual correct mappings. To find this alignment forward-backward procedure[21] is used.

3.2 The aligning process

We have divided the process of aligning into independent phases. According to specific needs some of the phases of the process may not occur or occur more than once. Figure 1. shows the data flow in the process.

The important feature of the scheme is the same data type – the alignment – throughout the whole process (starting from the output of the parsing module). This allows for alternative paths in the process.

3.2.1 Parsing

The first phase of alignment is parsing the input documents in order to obtain the common data type. Various parsers are used for input formats such as: *TMX*[22], *HTML* and plain text. The *DOC* documents are supported by *heTMan*.

3.2.2 Segmenting

The next phase is the segmentation of both source and target texts. As the task is not trivial[23] and depends on input texts properties such as text languages, the rule based splitter devoted for the task was implemented. Rules are stored in the world-wide standard format called *SRX*[24]. Each rule consists of two regular expressions

[19] This length-based algorithm is similar to Gale-Church algorithm. See Brown et al. 1991.

[20] For more informations about the IBM models see Brown et al. 1993, Knight 1999.

[21] The forward-backward procedure is connected with Hidden Markov Models (HMM). For more details see Moore 2002, Rabiner 1989.

[22] TMX (stands for Translation Memory eXchange) is a standard format used for storing bilingual aligned corpora. See *http://www.lisa.org/standards/tmx/tmx.html* and Lipski 2007, chapter 4.

[23] The segmentation task is not trivial because periods are not always used to mark sentence boundaries – they can also occur in numerical expressions, dates, abbreviations and so forth. According to Gale and Church 1991 in the Wall Street Journal for example only 53% of the periods identify sentence boundaries.

[24] SRX (stands for Segmentation Rules eXchange) is a standard format used to describe segmentation rules. See *http://www.lisa.org/standards/srx/srx.html* and Lipski 2007, chapter 3.

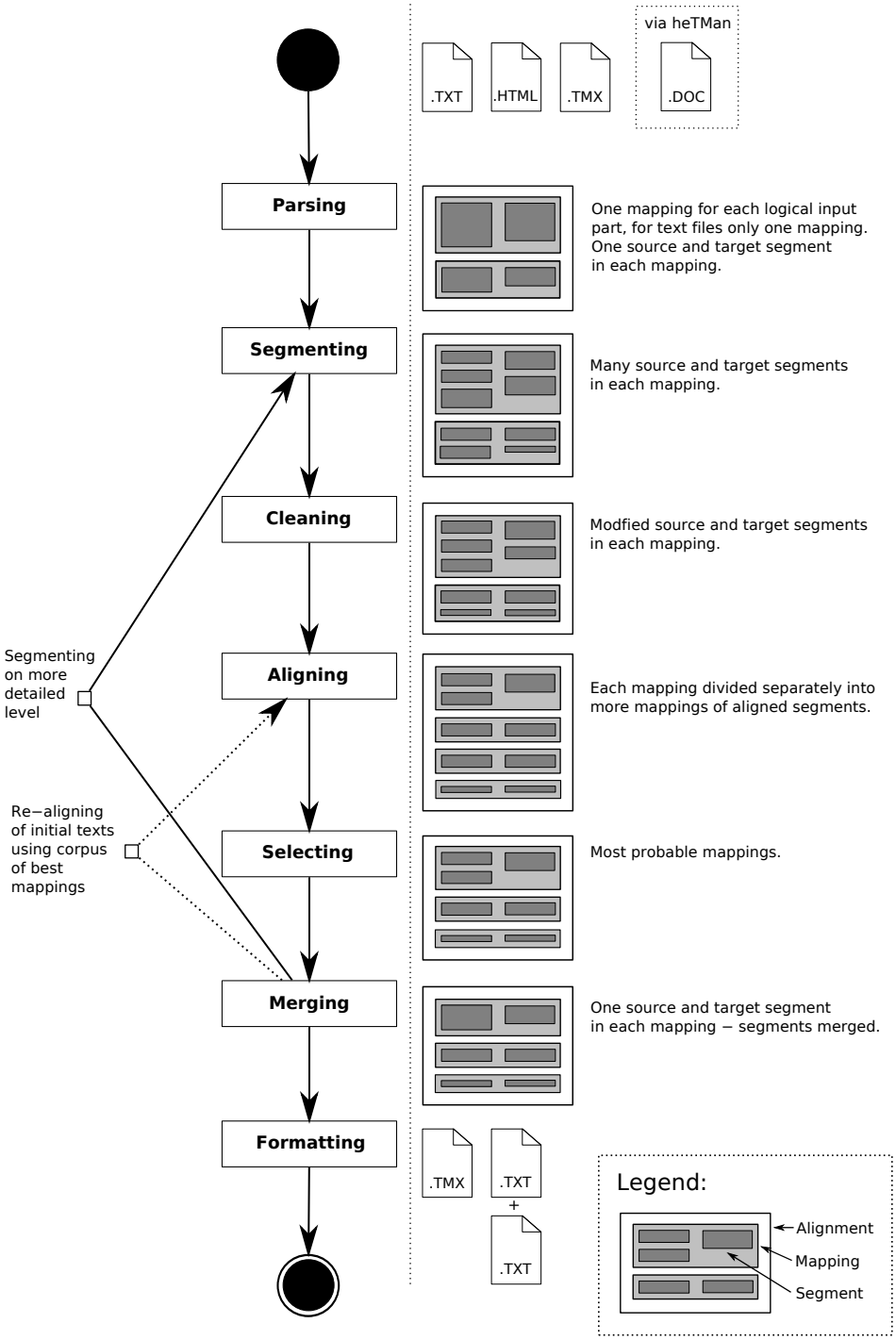

FIGURE 1: The aligning process.

describing texts before and after segment breaks. The number of mappings in the alignment remains unchanged during segmenting; only the number of segments in each mapping can change. The segmentation module may be invoked directly after parsing or after the phase of aligning at a given level. For example, once the texts have been aligned at the paragraph level, the segmentation procedure may be called in order to divide texts into sentences and then the aligning takes place at the sentence level preserving paragraph boundaries.

3.2.3 Cleaning

This phase cleans source and target segments from unwanted characters such as whitespaces at the ends and beginnings of segments that remain after the segmentation phase.

3.2.4 Aligning

This is the key phase of the process. During this phase corresponding segments are aligned separately in each mapping. The output alignment may be pushed back to the segmentation phase and aligned once more at a denser level. The user may choose between different algorithms including variations of the Gale-Church algorithm and the Moore algorithm.

3.2.5 Selecting

This phase selects best mappings according to some criteria. The user may choose for example to select only one-to-one mappings. There are also other possibilities, like selection of most likely mappings (e.g. those, which exceed a threshold of probability or a given fraction of best mappings[25]).

3.2.6 Merging

During this phase all source and target segments in each mapping are merged into one source and one target segment. Sometimes additional characters, such as whitespaces, are inserted between merged segments. This phase does not need to be executed if the output format is allowed to store mappings other than 1-to-1.

3.2.7 Formatting

The last phase consists in formatting the alignment to the desired format. The format can be either two *TXT* files with the same number of lines (as used by *Giza*), or a standard format for storing translation memory – *TMX*. A third format has also been developed, that is a simple format for presenting the results in a human-readable form.

[25]For example, 80% best mappings is an often used constant.

3.3 Pipeline

Each class performing one of the inner phases (segmentation, cleaning, aligning and selection) implements the same interface which takes an alignment as an argument and returns a modified alignment as a result. Such classes are sometimes called *Filters* because their input and output is of the same data type and they modify (e. g. "filter") their input. Furthermore, we call the classes performing the first phase *Parsers*, and the classes for the last phase – *Formatters*.

In the text interface of the align program each phase is invoked by a separate command, which reads the alignment list from the standard input and writes the output alignment list to the standard output. The commands may be linked by means of the pipe character to pass data between them. Usually, the first command in the pipeline calls the parser, then come one or more filters, and eventually – the formatter. This allows for the flexibility of the program. For example, to align text using the Gale-Church algorithm the following commands have to be executed[26]:

```
maligna parse -c txt poznan-pl.txt poznan-de.txt |
maligna modify -c sentencesplit |
maligna modify -c trim |
maligna align -c viterbi -a poisson -n word -s iterband |
maligna select -c onetoone |
maligna format -c txt poznan-pl-align.txt poznan-de-align.txt
```

3.4 Comparator

To avoid evaluating results by hand the comparator tool has been developed. The comparator is included in the project. It can compare two alignments, show the differences and calculate *precision* and *recall*[27].

The above metrics are used when a so-called *reference alignment* (usually executed by a human) is available. A reference alignment is assumed to contain exclusively the correct mappings between segments. We denote the estimated alignment as A and the reference alignment as B.

Precision defines the number of the correct mappings as compared to all mappings of the estimated alignment:

$$precision = \frac{|A \cap B|}{|A|}$$

Recall defines the number of the correct mappings as compared to all mappings of the reference alignment:

$$recall = \frac{|A \cap B|}{|B|}$$

[26] More examples can be found in *mALIGNa* documentation.

[27] Precision and recall are widely used measures to evaluate alignment algorithms. For more details about them see Krynicki 2006, Rosen 2005.

3.5 Optimizations

The complexity of alignment process is $n \cdot m$ where n is the number of segments in the source text and m is the number of segments in the target text, and the crucial operation is the calculation of the mapping probability. For texts containing thousands of segments there is need to optimize the algorithms.

The approach taken by Gale-Church is to align on many levels of accuracy (paragraphs, sentences). This is followed by the presented implementation thanks to modularity of alignment process.

The Moore algorithm uses a technique called "beam search"[28] to shrink the search space for the best alignment so that not all potential mappings probabilities need to be calculated. The technique is also used in our implementation.

References

Peter F. Brown, Jennifer C. Lai, Robert L. Mercer (1991), Aligning sentences in parallel corpora, Proceedings of the 29th annual meeting on Association for Computational Linguistics.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, Robert L. Mercer (1993), The Mathematics of Statistical Machine Translation: Parameter Estimation, Computational Linguistics, Volume 19, Issue 2.

William A. Gale and Kenneth Ward Church (1991), A Program for Aligning Sentences in Bilingual Corpora, Meeting of the Association for Computational Linguistics.

Krzysztof Jassem and Jacek Gintrowicz (2007), Using Regular Expressions in Translation Memories, Proceedings of the International Multiconference on Computer Science and Information Technology.

Kevin Knight (1999), A Statistical MT Tutorial Workbook, unpublished.

Grzegorz Krynicki (2006), Compilation, Annotation and Alignment of a Polish-English Parallel Corpus, PhD Thesis, Adam Mickiewicz University, Poznań.

Jarosław Lipski (2007), Urównoleglanie tekstów dwujęzycznych na poziomie zdania, Master's Thesis, Adam Mickiewicz University in Poznań.

Robert C. Moore (2002), Fast and Accurate Sentence Alignment of Bilingual Corpora, AMTA '02: Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users.

Lawrence R. Rabiner (1989), A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE.

Alexandr Rosen (2005), In search of the best method for sentence alignment in parallel texts, Computer Treatment of Slavic and East European Languages: Third International Serminar, Bratislava.

Monika Rosińska (2007), Collecting Polish-German Parallel Corpora in the Internet, Proceedings of the International Multiconference on Computer Science and Information Technology, Volume 2, XXIII Autumn Meeting of Polish Information Processing Society

[28] For more details on this technique see Moore 2002 and Lipski 2007, chapter 2.2.