Bitextor's participation in WMT'16: shared task on document alignment

Miquel Esplà-Gomis, Mikel L. Forcada
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant, E-03690 Sant Vicent del Raspeig, Spain
{mlf,mespla}@dlsi.ua.es

Sergio Ortiz-Rojas, Jorge Ferrández-Tordera
Prompsit Language Engineering,
Av. Universitat, s/n, Edifici Quorum III, E-03202 Elx, Spain
{sergio,jferrandez}@prompsit.com

Abstract

This paper describes the participation of Prompsit Language Engineering and the Universitat d'Alacant in the shared task on document alignment at the First Conference on Machine Translation (WMT 2016). Two systems have been submitted, corresponding to two different versions of the tool Bitextor: the last stable release, version 4.1, and the newest one, version 5.0. The paper describes the main features of each version of the tool and discusses the results obtained on the data sets published for the shared task.

1 Introduction

Parallel data harvesting has become a critical problem for many cross-lingual tasks in natural language processing. These data are the basis of many approaches, specially in the case of corpus-based machine translation (MT). One of the main sources of new parallel data is the Internet; in fact, many solutions have been proposed for exploiting specific websites by learning features of their structure. Some popular examples of corpora built by mining specific websites are the Europarl Corpus (Koehn, 2005), which exploits the European Parliament website, or the TED2013 corpus (Cettolo et al., 2012), that mines bitexts from the TED talks website,[1] a site that provides videos of public speeches and their transcriptions translated into several languages. Nevertheless, defining methodologies to surf the Web and identify parallel documents in any website is still an open problem. Some of the earliest tools developed for this purpose are STRAND (Resnik and Smith, 2003) and BITS (Ma and Liberman, 1999). These tools use similarities in the URLs and the content of the webpages to detect parallel documents in a given web domain. Based on these principles, many later approaches have been proposed (Nie et al., 1999; Chen et al., 2004; Zhang et al., 2006; Désilets et al., 2008; San Vicente and Manterola, 2012; Papavassiliou et al., 2013); this paper describes the participation of Prompsit Language Engineering and the Universitat d'Alacant in the shared task on document alignment of WMT 2016, based on one of these tools: Bitextor (Esplà-Gomis and Forcada, 2010).

The rest of the paper is organised as follows: Section 2 describes the main features of Bitextor, highlighting the main differences between versions 4.1 and 5.0. Section 3 describes the steps taken to produce the submissions for the shared task on document alignment in WMT 2016, and discusses the results obtained. Finally, some concluding remarks are provided in Section 4.

2 Bitextor

Bitextor is a free/open-source tool for harvesting parallel data from multilingual websites; it is highly modular and is aimed at allowing users to easily obtain segment-aligned parallel corpora from the Internet. This section summarises the evolution of the tool from its earliest versions, paying special attention to versions 4.1 and 5.0, corresponding to the systems submitted to WMT 2016.

The first version of Bitextor was developed in 2006 as a monolithic library written in C++. The core component of Bitextor to find document and sentence-level alignments was an XHTML sentence aligner called TagAligner,[2] which heavily relied on the HTML structure of the documents to be compared. Some analysis on the performance of this tool versus other approaches was presented in (Sánchez-Villamil et al., 2006). The first version of Bitextor used TagAligner along with some strategies aimed at spotting language identifiers or names in document URLs.

[1] http://www.ted.com

[2] http://tag-aligner.sf.net

At some point, the monolithic nature of Bitextor and its dependence on unmaintained external libraries made it hard for users to install it and get it working. Addressing these issues led to a dramatic restructuring for version 4.0 in order to ease the maintenance and to improve the performance of the tool. Bitextor was fully re-implemented and deeply modularised with parallel processing in mind. The new version was mostly implemented in Python and Bash. Most of the external libraries were replaced: LangID[3] was adopted for language detection; XML/HTML normalisation, previously carried out by W3C HTML Tidy,[4] was done now with the more modern and powerful Apache Tika;[5] and the library Boilerpipe[6] was included in the pipeline to simplify the document structure by removing boilerplate material. As regards the strategies for document alignment, heuristics based on language identifiers in URLs were replaced by the use of bilingual lexicons for shallow indexing of documents. This method reduces the search space when looking for translation candidates for a given document. For a more reliable source of information, Bitextor kept relying on the use of XHTML structural comparison for document alignment.

2.1 Current version: Bitextor 4.1

The architecture of Bitextor in versions since 4.0 is based on a Unix-style pipeline, in which a collection of scripts are connected using text interfaces. This architecture favours the parallelisation of subtasks and eases the maintenance of the tool. Figure 1 represents this architecture, describing the main modules in Bitextor and how they interact. As can be seen, the user is required to provide one or more URLs of websites to be processed, the two languages (L_1 and L_2) for which the parallel corpus will be produced, and a bilingual lexicon in these two languages. The following list describes the modules in Bitextor and how this input data is processed to obtain a translation memory:

1. *Website crawling*: given the URL of a website, it is completely downloaded by means of the tool *HTTrack*,[7] keeping only HTML documents; this module does not produce text output, but downloads a mirror of a webpage.

2. *Webpage normalisation*: downloaded documents are preprocessed with *Apache Tika*[8] and *Boilerpipe*[9] (Kohlschütter et al., 2010) to normalise the HTML markup into XHTML and remove boilerplates. After normalisation, exact duplicates are discarded. This module outputs a list of tab-separated fields, in which every line corresponds to a file. Four fields are included in each line: the MIME type,[10] the character encoding, the local path to the file processed, and the content of the document after normalisation encoded in `base64`;[11] this format is henceforth called `ett`.

3. *Language identification*: this module receives as an input the list of processed documents in format `ett`; the language of each document is detected with *LangID* (Lui and Baldwin, 2012),[12] keeping only those documents in one of the target languages (L_1 or L_2). Before language identification, Apache Tika is used to convert the XHTML content of the document into plain text. The module outputs a list of files in `lett` format, which consists of the same fields than `ett` plus the language identifier of the document and the plain text extracted, encoded in `base64`.

4. *XHTML structure representation extraction*: this module receives a list of files in `lett` format and obtains a string that tries to represent the XHTML structure as follows: (i) every different XHTML tag is replaced by an arbitrary character, and (ii) the sequence W of words between two XHTML tags is represented with a reserved character, which is repeated $\log_2(|W|)$ times to account for the length of the text (in words) in the text block. The objective of this representation is to ease the comparison of the structure of the documents by reducing it to a string comparison. The new field is added to the `lett` input: the resulting format is the `lettr` format.

5. *Indexing of words in webpages*: this module receives a `lett` list of files and a bilingual lexicon and produces an `idx` index containing, for every word in the lexicon, the list of documents in which it occurs. The output of this module consists of a list of words, one

[3]https://github.com/saffsd/langid.py
[4]http://tidy.sf.net
[5]https://tika.apache.org
[6]https://github.com/kohlschutter/boilerpipe
[7]http://www.httrack.com/
[8]http://tika.apache.org/
[9]http://code.google.com/p/boilerpipe/
[10]https://wikipedia.org/wiki/Media_type
[11]https://tools.ietf.org/html/rfc4648#section-4
[12]https://github.com/saffsd/langid.py

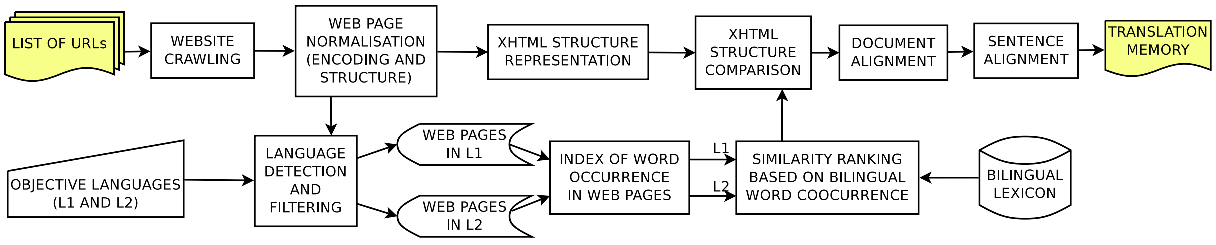
LIST OF URLs
WEBSITE CRAWLING
WEB PAGE NORMALISATION (ENCODING AND STRUCTURE)
XHTML STRUCTURE REPRESENTATION
XHTML STRUCTURE COMPARISON
DOCUMENT ALIGNMENT
SENTENCE ALIGNMENT
TRANSLATION MEMORY
OBJECTIVE LANGUAGES (L1 AND L2)
LANGUAGE DETECTION AND FILTERING
WEB PAGES IN L1
WEB PAGES IN L2
INDEX OF WORD OCCURRENCE IN WEB PAGES
L1
L2
SIMILARITY RANKING BASED ON BILINGUAL WORD COOCURRENCE
BILINGUAL LEXICON

Figure 1: Architecture of Bitextor.

per line, and a numeric identifier for each of the documents in which the word appears; the first identifier corresponds to the line number of the first document in the `lett` list of documents, and the remaining identifiers are offsets to the previous line numbers, to reduce the size of the list. [13]

6. *Similarity ranking based on bilingual word coocurrence*: this module receives the `idx` word index and the `lett` document list and computes a bag-of-words overlapping metric for each pair of documents. This score is used to build a preliminary list of n-best candidates for each document.[14]
7. *N-best lists re-ranking*: the list of the n best candidates obtained for each document is re-ranked by using the similarity metric based on the Levenshtein edit distance between the representation of the XHTML structure of each pair of documents obtained in module 4; the `lettr` list of documents is used in this step.
8. *Document alignment*: Once re-ranked, the n-best lists for both languages are compared, and those documents that are mutually among the best candidates are aligned.[15]
9. *Sentence alignment*: aligned documents are finally aligned at the level of segments by using the tool *hunalign*[16] (Varga et al., 2007). The standard output after this step is a translation memory, but Bitextor can also be run to obtain just a list aligned documents. In this specific case, the list of document pairs obtained in the previous step is filtered by using the reliability score at the level of document pairs produced by hunalign to discard very unlikely document pairs.

Three main bottlenecks can be identified in this structure:

- crawling, given that this process is carried out by an external tool and it is only after the whole website is downloaded that the next step can start;
- obtaining the `lett` list of documents, since until the full list of documents is obtained it is impossible to compute the whole `idx` index of words; and
- n-best list candidates ranking, since documents cannot be aligned until the full ranking is obtained to check which documents are mutually best candidates.

The modules in between these bottlenecks are run in parallel; this allows Bitextor to obtain a high performance in machines with several processors.

2.2 What is new in Bitextor 5.0?

Version 5.0 of Bitextor has dramatically modified the way in which the tool performs two of the most important sub-tasks of its pipeline: crawling of websites and document alignment. This section is aimed at describing the main novelties as regards these modules. Note that the architecture shown in Figure 1 stays the same for Bitextor 5.0, despite the fact that the internal behaviour of the corresponding modules changes.

Web crawling. Until version 4.1, the tool HT-Track was used for downloading websites, and after this tool was done, the rest of the processing was carried out. Version 5.0 of Bitextor implements a new module for crawling websites based on the Python library `creepy`[17] which allows a better control of the crawling process, which can be interrupted at will to perform other processing. The two main advantages of this process are:

[13]For instance, if the word appears in documents 100, 105 and 180 the list would be `100:5:75`.

[14]The standard size of these n-best lists is 10.

[15]It is possible to specify how many documents in the re-ranked n-best list are taken into account: if only the first one is taken into account (the highest one after re-ranking), only mutual best-candidates are finally aligned.

[16]`http://mokk.bme.hu/resources/hunalign`

[17]`https://github.com/Aitjcize/creepy`

- *Better parallelisation of the processing*: with the new crawling module it is easier to control the way in which websites are crawled, allowing to specify the number of parallel threads that can be used. In addition, the need of storing a mirror of the original website locally disappears;[18] instead of this, the documents downloaded are directly stored in the `ett` format, which allows to start webpage normalisation before the whole website is downloaded.

- *Higher control of the crawling process*: the new module allows for a more controlled crawling process. For example, it is possible to avoid following links found in a document that is not written neither in L_1 nor L_2.

Document alignment. The modules of Bitextor involved in the identification of parallel documents in a given website have undergone important changes as well. As described in Section 2.1, previous versions of Bitextor used two sources of evidence to identify candidate pairs of documents: a bag-of-words overlapping metric and a similarity metric based on the structure of the documents, both using the distribution of the text and the XML/HTML structure. In Bitextor 4.1, the first source was used to reduce the search space and create a preliminary ranking of n-best candidates for every document, while the second one, more reliable, was used to re-rank this list. Bitextor version 5.0 keeps the initial strategy for reducing the search space by using the bag-of-words overlapping metric, but adds new sources of evidence and uses a logistic regression approach to combine them for re-ranking the n-best list of candidates. These new sources of information extracted for every candidate pair of documents are:

1. The Jaccard index of the URLs: when comparing documents D_1 and D_2, all the URLs in each document are extracted (using the HTML tag `href`) obtaining the sets U_1 and U_2, respectively; the Jaccard index is then computed as: $\frac{|\{U_1 \cap U_2\}|}{|\{U_1 \cup U_2\}|}$;

2. The similarity of URLs inside the document, represented by the Levenshtein distance (Wagner and Fischer, 1974) between the sequence of the URLs contained both in D_1 and D_2 at the character level;

3. The Jaccard index of the images shared: the URLs of the images in documents D_1 and D_2 are extracted (using the HTML tag `img`) obtaining the collections I_1 and I_2, respectively; the Jaccard index is then computed as: $\frac{|\{I_1 \cap I_2\}|}{|\{I_1 \cup I_2\}|}$;

4. Mutual links: a binary feature that is `true` if both documents are mutually linked, and `false` otherwise; and

5. Document URL distance: the Levenshtein distance (Wagner and Fischer, 1974) between the URLs corresponding to D_1 and D_2.

These new metrics, together with the two original ones (bag-of-words overlap and structure comparison), are used as features[19] in a logistic regression algorithm based on the use of a multilayer perceptron implemented with the Python library *keras*.[20] The logistic regression algorithm is trained to obtain a real number in $[0.0, 1.0]$ where 0.0 means that the documents are totally unrelated and 1.0 means that the documents are parallel. The score obtained by the logistic regressor allows to rank the candidates in the n-best list for every document. One of the main advantages of this approach, apart of being more empirical and less arbitrary than the previous heuristic approach is that it provides a similarity score at the level of document pairs, more reliable and easy to obtain than the one obtained from hunalign, and which does not require to align the documents at the level of segments.

3 Bitextor for document alignment in WMT 2016

This section describes the problem proposed by the organisers of the shared task in document alignment and the two systems submitted by Prompsit Language Engineering and the Universitat d'Alacant, the two main institutions supporting the tool Bitextor.

3.1 Data sets

The organisers of the shared task on document alignment provided a collection of `lett` files containing the collection of documents crawled from several multilingual websites. Two different data sets were provided: a training set consisting of a collection of 49 crawled websites with a total of

[18]This means that one of the bottlenecks specified in Section 2.1 is avoided in this version.

[19]All these metrics are normalised and take values in $[0.0, 1.0]$ except for *mutual link*, which is binary.

[20]`http://keras.io`

573,953 documents, and a test set consisting of a collection of 203 crawled websites, with a total of 1,204,239 documents. For the training set, a collection of gold document alignments was provided for a sub-set of the whole collection of documents.

The objective of the task is to build a collection of documents in English aligned to their translations in French.

3.2 Submitted systems

The only external resource required by Bitextor, a bilingual lexicon, was taken from the project's webpage.[21] The two versions of the tool used to produce the submissions were run with the standard parameters for document alignment, with the only exception of the parameter that specifies the amount of candidates to be taken into account in the n-best list: that was set to consider only the first one (see the description of the *document alignment* module in Section 2.1). As regards the submission based on Bitextor 4.1, its standard pipeline includes a filtering of the document pairs using the score provided by hunalign (see Section 2.1), while Bitextor 5.0 does not use any filtering at the document level.

Given that Bitextor 4.1 uses an heuristic approach, the training set was not used to build this system. However, Bitextor 5.0 does need[22] to train the logistic regressor used to rank the n-best translation candidates for a given document in a website; the training set was therefore used for this purpose in the following way:

1. the websites in the training set, which were already provided in the `lett` format, were processed up to the step in which the n-best lists are built and the features described in Section 2.2 were obtained for every candidate pair of documents, i.e. for the n pairs consisting of a document and each of its n-best candidates;

2. those document pairs for which neither of them was not found in the gold standard provided by the organisation were discarded;

3. for the remaining pairs of documents, those in the gold standard were considered as positive samples (for which the expected output of the logistic regressor is 1.0), while those aligning a document in the gold standard with a different document were considered as negative samples (for which the expected output of the logistic regressor is 0.0).

Following this method, a collection of 30,815 training samples was obtained,[23] which was randomised to use 10% of the samples as development set, and the remaining 90% as training set.

3.3 Results

The results obtained with each version of Bitextor consisted of a collection of 95,760 pairs of documents in the case of Bitextor 4.1 and 157,682 in the case of Bitextor 5.0; that is, the new version detected about 60% more document pairs. The organisers of the task took a sample of 2,402 URL pairs as a gold standard for evaluating the recall of each system. On this evaluation framework, Bitextor 4.1 obtained a recall of about 31%, while Bitextor 5.0 obtained a recall of about 83%. After a careful revision of the results obtained, it is worth mentioning that some errors were detected in the gold standard, which led to considering as wrong some correct document pairs detected by Bitextor. After fixing them, Bitextor 5.0, the best performing version of the tool, obtained a recall of about 87.5%. In addition, some ambiguities were detected in the gold standard that had not been taken into account, such as URL aliases (having two alias that lead to the same document) or language variants (for example, having a document in British English and American English). After adding these ambiguous cases to the gold standard, the recall of Bitextor grew to almost 88%. Finally, it was possible to find websites translated into several languages for which some documents were not translated and are therefore written in the default language (English in most of the cases). This problem is discussed in Section 3.4.3 and, as explained in this section, after boilerplates removal, bitextor would keep only English text from these documents, producing a valid alignment. If we consider these alignments as valid, the recall would reach 89%.

Regarding the quality of the aligned document set obtained, Bitextor 4.1 reached a precision of about 85%, while in the case of Bitextor 5.0, the precision was higher than 90%. Taking into account the errors in the gold standard, this value grows to more than 95%.

These results confirm that the novelties in the new version of Bitextor provide a considerable improvement in the performance of the tool for docu-

[21]`https://sf.net/projects/bitextor/files/lexicons/`

[22]The new release published for Bitextor 5.0 includes a pre-trained regression model, so it does not need to be trained every time it is used.

[23]Since the standard size of the n-best lists is 10, about 10% of the samples were positive, while remaining samples were negative.

ment alignment. When compared to other systems participating in the task, Bitextor obtains a performance that falls in the middle of the ranking. However, taking into account the issues regarding the gold standard described in this section, Bitextor 5.0 would rank among the 5-top systems submitted in the current ranking.[24]

3.4 Error analysis

A deeper look into the data sets provided by the organisation of the shared task and the results obtained with Bitextor uncovered some of the most important problems faced when dealing with document alignment in an environment such as that of multilingual websites. The following are some of the main problems detected in the case of Bitextor.

3.4.1 Webpages not translated.

It is usual to find websites in which some pages are not translated in all the languages offered; this introduces noise into the task, since the tool may be looking for non-existing translations for some documents. If this happens only in a language (usually, the source language in which the original pages are written) it is not a big problem: untranslated webpages are just ignored. However, this issue has a relevant impact in the final accuracy when there are untranslated documents in both languages; in this case, the risk to end up aligning two documents for which no translation is available is higher. A good example of this situation in the test set is the website `http://academiedesprez.org`.

3.4.2 Webpages with little text.

This problem is usual in catalogues in which a template is used and only a few words or phrases (names, prices, measures, etc.) change in the different pages. This makes pages very similar and rises the probability of obtaining wrong alignments, which affect both precision and recall. An extreme example of this problem is the website `http://milltowndowntown.com` in the test set, that contains an extensive collection of pictures, each presented in a webpage without any text. For the purpose of building a corpus of parallel texts, it may be interesting to set a filter to discard those documents that do not contain a minimum amount of text to reduce the noise produced by this kind of webpages.

[24]It is worth noting that fixing these problems in the gold standard would possibly affect the rest of systems, and the whole ranking would need to be built again.

3.4.3 Repeated webpages.

In multilingual websites, it is usual to find that, when an article or a piece of news is not translated, it is shown in the *default* (original) language. As a result, it may happen that two webpages could be basically equivalent with the only exception of some menus or titles that are translated according to the template of the website.[25] In a real-world scenario, any of these "equivalent documents" would be a valid alignment and it would not be an error at all; however, given that the gold standard used for evaluation only provides a valid candidate for every document, this has an impact both in the precision and the recall of the tool. A good example of this problem is the website `https://pawpeds.com`. It is worth mentioning that, even though this issue affects the evaluation results, if the objective of document alignment is to produce a parallel corpus, aligning a document to any of its equivalent translations should not be considered an error at all.

4 Concluding remarks

This paper describes the systems submitted to the document alignment shared task at WMT 2016 by the team consisting of Prompsit Language Engineering and the Universitat d'Alacant. These submissions are based on Bitextor, a free/open-source tool for building parallel corpora from multilingual websites. For this shared task, two different versions of Bitextor were used to produce the two submissions: version 4.1 and 5.0. The results obtained show that the new version of Bitextor is able to identify a noticeably higher amount of parallel documents (about 60% more). In addition, the preliminary results obtained show that version 5.0 performs better than version 4.1 both as regards precision and recall in document classification.

Bitextor is distributed under version 3 of the GNU General Public Licence and can be downloaded from the project's website: `https://sf.net/projects/bitextor/files/bitextor/`.

Acknowledgements: Supported by the European Commission through project PIAP-GA-2012-324414 (Abu-MaTran) and by the Spanish government through project TIN2015-69632-R (Effortune).

[25]This information is usually discarded when boilerplates are removed.

References

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation*, pages 261–268, Trento, Italy.

Jisong Chen, Rowena Chau, and Chung-Hsing Yeh. 2004. Discovering parallel text from the world wide web. In *Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, volume 32, pages 157–161, Dunedin, New Zealand.

Alain Désilets, Benoit Farley, Marta Stojanovic, and Geneviève Patenaude. 2008. WeBiText: Building large heterogeneous translation memories from parallel web content. In *Proceedings of Translating and the Computer*, pages 27–28, London, UK.

Miquel Esplà-Gomis and Mikel L. Forcada. 2010. Combining content-based and URL-based heuristics to harvest aligned bitexts from multilingual sites with bitextor. *The Prague Bulletin of Mathematical Linguistics*, 93:77–86.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the X Machine Translation Summit*, pages 79–86, Phuket, Thailand.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450, New York, NY, USA.

Marco Lui and Timothy Baldwin. 2012. Langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea.

Xiaoyi Ma and Mark Liberman. 1999. BITS: A method for bilingual text search over the web. In *Machine Translation Summit VII*, pages 538–542, Singapore, Singapore.

Jian-Yun Nie, Michel Simard, Pierre Isabelle, and Richard Durand. 1999. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–81, Berkeley, California, USA.

Vassilis Papavassiliou, Prokopis Prokopidis, and Gregor Thurmair. 2013. A modular open-source focused crawler for mining monolingual and bilingual corpora from the web. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 43–51, Sofia, Bulgaria.

Philip Resnik and Noah A. Smith. 2003. The Web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Iñaki San Vicente and Iker Manterola. 2012. PaCo2: A fully automated tool for gathering parallel corpora from the web. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 1–6, Istanbul, Turkey.

Enrique Sánchez-Villamil, Susana Santos-Antón, Sergio Ortiz-Rojas, and Mikel L Forcada. 2006. Evaluation of alignment methods for html parallel text. In *Advances in Natural Language Processing*, pages 280–290. Springer.

Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2007. Parallel corpora for medium density languages. *Amsterdam Studies in the Theory and History of Linguistic Science, Series IV*, 292:247–258.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.

Ying Zhang, Ke Wu, Jianfeng Gao, and Phil Vines. 2006. Automatic acquisition of Chinese–English parallel corpus from the web. In *Advances in Information Retrieval*, volume 3936, pages 420–431. Springer Berlin Heidelberg.