Evaluation of alignment methods for HTML parallel text

Enrique Sánchez-Villamil, Susana Santos-Antón,
Sergio Ortiz-Rojas, and Mikel L. Forcada

Transducens group, Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant, E-03071 Alacant, Spain
{esvillamil,ssantos,sortiz,mlf}@dlsi.ua.es

Abstract. The Internet constitutes a potential huge store of parallel text that may be collected to be exploited by many applications such as multilingual information retrieval, machine translation, etc. These applications usually require at least sentence-aligned bilingual text. This paper presents new aligners designed for improving the performance of classical sentence-level aligners while aligning structured text such as HTML. The new aligners are compared with other well-known geometric aligners.

1 Introduction

Many machine translation applications are based on machine learning on parallel corpora. The amount of parallel text required to obtain accurate translations using these applications is quite high (up to hundreds of megabytes) although it seems possible to generate such large corpora using the Internet. The utility of the corpora increases dramatically when they are aligned at sentence or word levels.

A number of sentence-alignment approaches have been developed during the last years. The first effective approach at aligning large corpora was based on modeling the relationship between the lengths of sentences that are mutual translations (Brown et al., 1991; Gale and Church, 1991, 1993). Chen (1993) used a different approach, based on lexical information to improve accuracy, but it was slower than sentence-length-based algorithms. Some years later, Melamed (1996) developed a method based on word correspondences and supported by external linguistical knowledge.

All these aligners are designed to work with text segmented in sentences. In our case, collections of hundreds of megabytes of downloaded webpages, which are not segmented, have to be aligned at sentence-level. These pages are turned into XML[1] using the `tidy` program,[2] which may be used to turn HTML into XHTML.[3]

[1] `http://www.w3.org/TR/2004/REC-xml-20040204/`
[2] `http://www.w3.org/People/Raggett/tidy/`
[3] XHTML is a stricter and cleaner XML-version of HTML.

The aligners proposed in this paper are being used to generate a large collection of aligned text corpora. The corpora will be segmented, and segments will be aligned to build translation units. The resulting translation units may be used to train translation applications.

In particular, this paper presents a type of aligners that combine sentence-splitting and alignment generation, and take advantage of the structured nature of web documents to improve the accuracy of sentence-aligned text in the absence of linguistic knowledge. The aligners are compared to classical approaches in the experiments.

2 Notation

In this paper, we define the *alignment* as a sequence of edit operations, that is, a sequence of insertions, deletions and substitutions of segments.[4] Let $L = (l_1, l_2, ..., l_{|L|})$ and $R = (r_1, r_2, ..., r_{|R|})$ be two parallel texts split in segments and $S = (s_1, s_2, ..., s_{|S|})$, a sequence of edit distance operations, where s_i can be an insertion (m_i), a deletion (m_d) or a substitution (m_s) of a segment. It is straightforward to obtain the aligned segment pairs (l_i, r_i) using the edit distance sequence. We define A as the function returning the edit-distance alignment of two texts, so that $A(L, R) \longrightarrow S$.

Additionally, we define the alignment distance D that is considered as a measure of the similarity of the texts that have been aligned. The distance D is defined as the addition of the differences in length of all aligned segments:

$$D(S) = \sum_{i=1}^{|S|} \mathrm{abs}(|l_i| - |r_i|) \tag{1}$$

where bars $|\cdot|$ are used to represent the length of a text segment. The m_i and m_d operations where either l_i or r_i would be the empty string are also taken into account.

3 Classical geometric aligners

Geometric aligners are based only in geometric properties of the documents, such as sentence lengths, word lengths, paragraph lengths, etc. They are fully independent of the language because they do not use linguistic information.

Classical geometric aligners were designed to align plain text segmented in sentences. However, they can be adapted to marked-up corpora, such as XHTML, in several ways. The simplest approach would be the removal of all tags in both sides, so that a pair of plain texts would be obtained and would then be split; such aligner is called *Remover*. A more elaborate algorithm would require the substitution of some tags by sentence boundaries[5] (and the removal of the rest

[4] This definition induces a monotone alignment.

[5] The tags replaced are `hr`, `br`, `p`, `li`, `ul`, `ol`, `tr`, `td`, `th`, `div`.

of tags) and the aligner that implements this algorithm is called *Replacer*. Both aligners will be used as a baseline to evaluate the sentence-alignment algorithms presented in this paper.

4 Geometric aligners based on structure

The sentence-alignment algorithms that are presented in this paper combine sentence splitting with the alignment process. The algorithms work with structured text such as XHTML, but can be generalized easily to be applied to other XML-based formats.

These algorithms are based on a classification of tags that guide the initial splitting of the text. After that, a sequence of tags and text segments is extracted to perform the alignment, which will never allow the alignment of tags to text segments.

4.1 Classification of XHTML tags

In order to maximize the alignment accuracy, XHTML tags have been classified in several different categories. Originally, tags were divided in *block* and *inline* tags as in the XHTML DTD, but these partition was refined and the following four categories were defined:

– Structural tags: Tags that compose the structure of the webpage and its graphical representation: `blockquote, body, caption, col, colgroup, dd, dir, div, dl, dt, h1, h2, h3, h4, h5, h6, head, hr, html, li, menu, noframes, noscript, ol, optgroup, option, p, q, select, table, tbody, td, tfoot, th, thead, tr, ul.`
– Format tags: Tags that specify the format of some elements of the webpage: `abbr, acronym, b, big, center, cite, code, dfn, em, font, i, pre, s, small, span, strike, strong, style, sub, sup, tt, u.`
– Content tags: Tags that contain relevant elements (for alignment purposes), which are neither structural nor format tags: `a, area, fieldset, form, iframe, img, input, isindex, label, legend, map, object, param, textarea, title.`
– Irrelevant tags: Tags that are ignored[6] during the alignment process: `address, applet, base, basefont, bdo, br, button, del, ins, kbd, link, meta, samp, script, var.`

Most block tags are structural tags, and most inline tags are format tags. Content tags represent basically inline tags that do not contain format. Tags that are not useful in the alignment process are classified as irrelevant tags.

Such a classification allows to set specific substitution costs regarding to the categories of the tags; for instance, the costs of substitutions involving structural tags will be higher than those of substitutions involving format tags.

[6] In fact, irrelevant tags are simply removed before aligning.

#	**Characters before**	**Characters after**	**Points**
1	-	a number	-0.5
2	-	a blank space	$+0.5$
3	-	a non-capital letter	-0.2
4	-	another dot	-0.5
5	-	a blank space and a capital letter	$+0.5$
6	-	a blank space and a non-capital letter	-0.2
7	a capital letter	-	-0.5
8	a word of 3 characters or less	-	-0.5
9	a blank space	-	$+0.2$
10	a ' or " character	a ' or " character	-0.5
11	another dot	-	$+0.4$

Table 1. Triggers applied to the context of dots to score sentence borders.

4.2 Sentence-splitting heuristics

Text segments have to be split into sentences so that they can be aligned. The splitting algorithm considers many of the tags as sentence boundaries, which often generates small segments that do not even contain a single sentence. After that, sentence splitting algorithms are applied to ensure that the alignment is performed at sentence-level, so that no segment contains more than one sentence.

The algorithms in this paper use heuristics to find sentence boundaries. Initially, all breaking points[7] inside a text segment are located. After that, question marks and exclamation marks are considered sentence boundaries and dots are analysed to detect if they constitute sentence boundaries.

The analysis of dots is based on a list of triggers, which is shown in table 1, that assign scores to breaking points, so that breaking points with a score higher than a threshold, which was defined as -0.2, will be considered sentence borders. The scores associated with the triggers and the threshold have been experimentally adjusted. The threshold was defined to be negative so that if no trigger is executed the breaking point is considered as a sentence border.

4.3 Text alignment

The alignment process is performed at the same time for tags and text segments, that is, the edit distance algorithm is applied to generate the best alignment between tags and text segments.

The edit distance costs were specified according to our alignment constraints and some decisions about the alignment of parallel texts:

– A tag cannot be aligned to a text segment or vice versa.
– A structural tag should not be aligned to a format, content or irrelevant tag, and the cost of the alignment with a different structural tag should be high.

[7] Breaking points are dots (.), question marks (?) and exclamation marks (!).

	Insertion	**Struct. tags**	**Format tags**	**Content tags**	**Text segm.**
Deletion	-	1	0.75	1.25	0.01 $\|r\|$
Struct. tags	1	1.5	1.75	H	H
Format tags	0.75	1.75	0.4	H	H
Content tags	1.25	H	H	H	H
Text segm.	0.01 $\|l\|$	H	H	H	Δ

Table 2. Edit distance costs between different items in the text. See text for a definition of Δ.

– A format tag should not be aligned to a tag of different type, and the cost of the alignment with a different format tag should be low.
– A content tag should only be aligned with the same tag.
– To favor tag alignment, text chunks should only be aligned between them and costs of their aligment should be lower than those that involve tags.

These costs are defined in table 2, where H is a value high enough to be never used in the edit distance process. The value of the symbol Δ will be defined specifically for each aligner.

The *2-in-1 aligner* splits both texts in tags and text sentences and then aligns them. This aligner defines $\Delta = 0.015\ (\text{abs}(|l| - |r|))$, that is, the difference between the text sentences lengths multiplied by a factor. The factor 0.015 has been established to be between the cost of inserting/deleting text characters and the sum of both costs, that was defined experimentally as 0.01 in the table 2.

The *2-steps aligner* splits the text in tags and text segments, which can contain more than one sentence. The first step consists in aligning tags and text segments among them. After that, the second step consists in aligning the sentences contained in the text segments obtained in the first step.

Two variants of this last aligner have been tested: the first one defines $\Delta = 0.015\ (\text{abs}(|l| - |r|))$, that is called *2-steps-L aligner*, where L means length, given that this first variant is based in length differences. The second one defines $\Delta = 0.01\ D(A(l, r))$, that is the alignment distance between the text segments multiplied by a factor, and it is called *2-steps-AD aligner*.[8] The 2-step-L factor is the same used in the 2-in-1 aligner, but the 2-steps-AD factor is the cost of inserting/deleting text characters, so that the substitution cost would remain lower than the sum of the insertion cost plus the deletion costs if the substitution were possible.

5 Experiments

The experiments performed to assess the quality of the aligners are based on several sentence-aligned corpora. These corpora were downloaded from three different websites. Then, all corpora were aligned and manually corrected using

[8] AD stands for *alignment distance*.

a program with a graphical user interface[9] oriented to checking and manipulating alignments.

5.1 Corpora

Three different corpora have been used to evaluate the quality of the alignments. Each of them represents a step forward in alignment difficulty. A measure of the difficulty in the alignment of parallel text could be the number of sentences that are aligned to blank in a manual alignment. The higher the number of blank alignments established, the less parallel the aligned texts are.

The first corpus has been downloaded from the Internet with a collector of parallel corpora called Bitextor (Sánchez-Villamil et al., 2006)[10]. This corpus contains 3.3 megabytes of Spanish–Catalan parallel text that was downloaded from `www.elperiodico.com`, an online daily newspaper.

The second one is a small fragment of the Quixote (196 kilobytes) that was downloaded from the Miguel de Cervantes Digital Library,[11] and it constitutes an Spanish–English corpus.

And the third one is a little collection of parallel text files in Spanish, Portuguese, Italian, Catalan and Galician that compose the help texts of the popular chatting program mIRC.[12] It contains a total of 96 kilobytes distributed in the five languages. All ten possible language pairs were aligned.

The corpus downloaded using Bitextor has 2.14% of sentences aligned to blank, which makes it the easiest one. The Quixote corpus has 19.08% which makes it harder to be aligned, and the 26.42% of the mIRC corpus makes it the hardest one.

5.2 Metrics

The metrics that have been selected to evaluate the quality of the alignment generated by the different aligners are the same as in (Black et al., 1991), that is, the precision, the recall and the F-measure. All of them require a reference alignment to calculate the number of correct alignments and the total of reference alignments. These metrics are based on sentences and are defined as follows:

$$\text{precision} = \frac{\#\ \text{correct alignments}}{\#\ \text{proposed alignments}} \tag{2}$$

$$\text{recall} = \frac{\#\ \text{correct alignments}}{\#\ \text{reference alignments}} \tag{3}$$

$$F = 2 \cdot \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{4}$$

[9] Very similar to that of the `bitext2tmx` bitext aligner, `http://www.sourceforge.net/projects/bitext2tmx/`.
[10] `http://www.sourceforge.net/projects/bitextor/`
[11] `http://www.cervantesvirtual.com/`
[12] `http://www.mirc.co.uk/translations/index.html`

However, in our case, the direct comparison of the results of the aligners would not make sense because there is a significant length diference of the resulting alignments generated by different aligners. Therefore, concatenation of alignments was allowed in the comparison. This means that a pair of aligned segments is correct if: (a) the same pair is found in the reference alignment or (b) the same pair can be built by concatenating pairs of the reference alignment.

Furthermore, an additional metric has been applied to perform the evaluation. This metric is based on considering the number of sentence boundaries, instead of sentences, that were aligned properly. In (Melamed, 1996), Melamed used a method of evaluating bitext mapping algorithms which consists in comparing their output to a hand-constructed reference set of points, which, in our case, are the sentence boundaries. This metric allows to handle successfully the differences in length in the alignments proposed, although does not completely guarantee the correction of the alignments proposed.

5.3 Results

The experiments have been performed using five different aligners. The first two, are the basic geometric aligners *Remover* and *Replacer* that were explained in section 3, which are used as a baseline. The last three are the aligners proposed in this paper, i.e., the 2-steps-AD aligner, the 2-steps-L aligner and the 2-in-1 aligner.

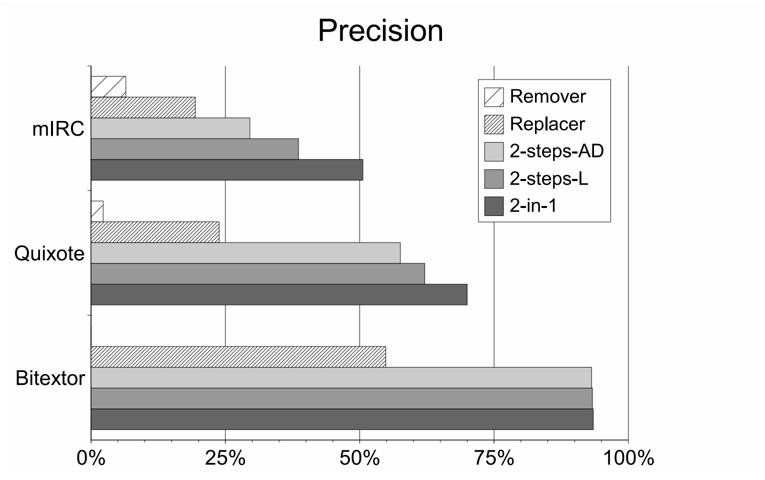

Fig. 1. Precision achieved by the aligners.

The results that we obtained with the metrics based on sentences are shown in figures 1, 2 and 3. As can be seen, the precision and recall of the resulting

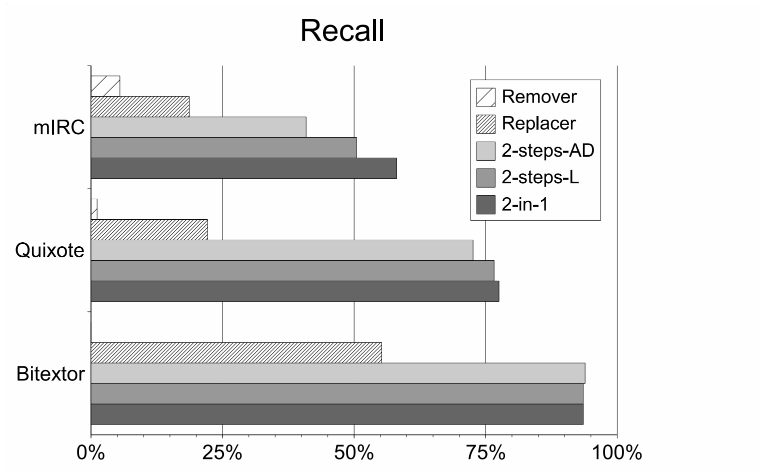

Fig. 2. Recall achieved by the aligners.

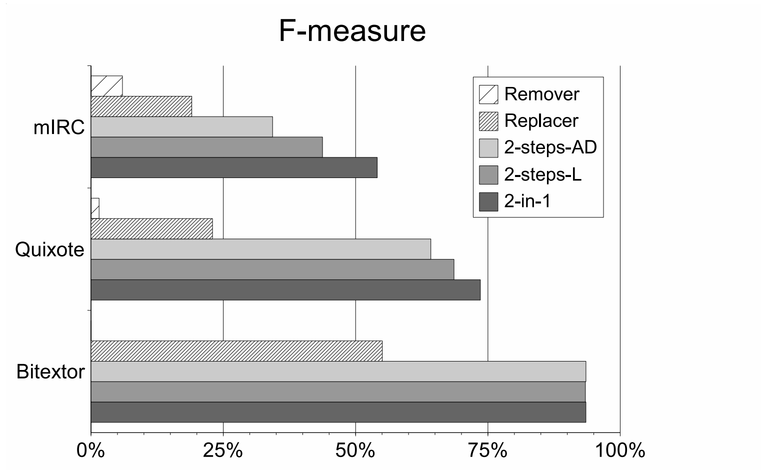

Fig. 3. F-measure achieved by the aligners.

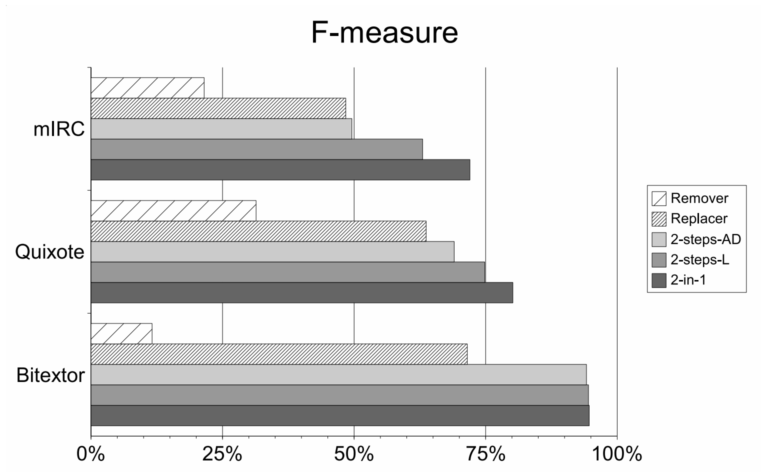

Fig. 4. *F*-measure achieved by the aligners using the sentence-boundary metric.

	Remover	**Replacer**	**2-steps-AD**	**2-steps-L**	**2-in-1**
mIRC	38(65)	33(52)	29(46)	29(46)	29(47)
Quixote	75(166)	40(95)	37(85)	37(85)	37(86)
Bitextor	95(285)	22(31)	21(29)	21(29)	21(29)

Table 3. Average segment length (standard deviation) for each corpus and each aligner in characters.

alignments is quite different for each corpus. As expected, the best results were obtained in the Bitextor corpus, and the mIRC corpus had the worst results. The results of the proposed aligners were clearly better than those of the basic geometric aligners.

Similar results were obtained using the metric based on sentence boundaries, as it is shown in figure 4 (only the *F*-measure is given). The results were slightly better because the sentence-based metric requires the coincidence of two consecutive sentence boundaries,[13] while the sentence-boundaries metric only requires the coincidence of one of them.

The results obtained by the 2-in-1 aligner were the best in the three corpora, with more than 93% of *F*-measure in the Bitextor corpus, more than 73% in the Quixote corpus and more than 58% in the mIRC corpus.

Additionally, it is worth examining the average in segment length of the different aligners, given that they apply different sentence splitting criteria. In table 3 the comparison of the results of the different aligners is shown. As expected, the proposed aligners build shorter sentences than basic geometric aligners. The standard deviation is much higher than the average because the sentence length

[13] Two consecutive sentence boundaries delimit a sentence.

	Remover	**Replacer**	**2-steps-AD**	**2-steps-L**	**2-in-1**
Time	155 s	153 s	318 s	294 s	323 s

Table 4. Time spent by the aligners while processing the Bitextor corpus.

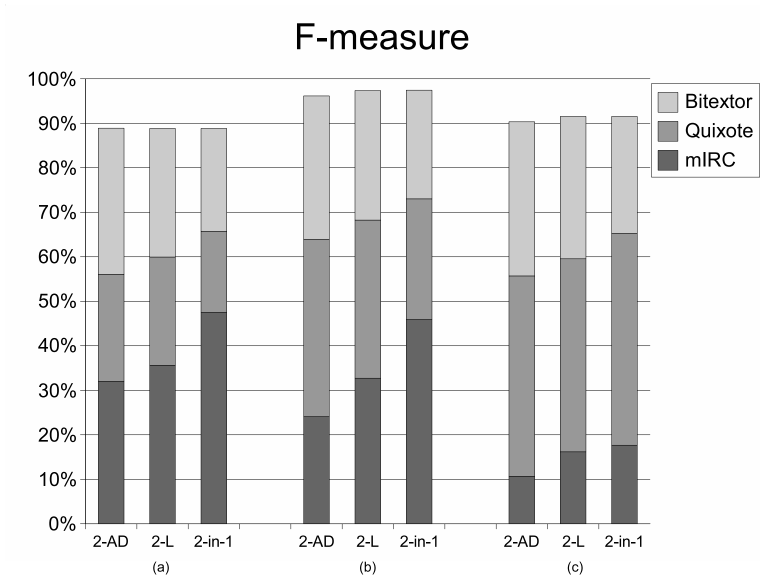

Fig. 5. F-measure achieved by the aligners in related experiments: (a) Ignoring format tags. (b) Ignoring content tags. (c) Ignoring format and content tags.

distribution in common texts is not a standard distribution, as it is explained in (Sigurd et al., 2003).

Nevertheless, as a counterpart, the tag aligners are slower than basic geometric aligners. Table 4 contains the results of the time comparison of the alignment of the Bitextor corpus. The processing times of basic geometric aligners were aproximately one half of the processing times of the tag aligners. The tag aligners align a higher number of items than basic aligners; this is because basic aligners do not consider the tags as items, which explains the significant difference in processing time.

Finally, some related experiments (changing the classification of tags) have been studied and are shown in figure 5. When format tags are considered as irrelevant tags, the results are slightly worse (7–18% worse for the mIRC corpus, 6–16% worse for the Quixote corpus and 4–5% for the Bitextor corpus). When content tags are considered as irrelevant tags, the results are quite different in each of the corpus: 17–22% worse for the mIRC corpus, similar results for the Quixote corpus and 2–4% better for the Bitextor corpus.

6 Conclusions

The alignment algorithms presented in this paper achieve a better level of quality, compared to classical algorithms, as has been shown in the experiments. This strongly suggests that using the tag structure of the webpages is very useful when aligning bitexts.

The quality of the bitexts used to perform the experiments was not optimal, that is, the bitexts were not accurate translations. In many cases, the tag structure of pairs of real parallel text was slightly different. In spite of this, the results have been clearly better than simply filtering the tags before the alignment.

Additionally, the aligned text segments generated by the proposed aligners have a smaller length than those obtained by basic geometric aligners. This may improve the reusability of the resulting translation units in many translation applications.

The comparison of the aligners revealed that the 2-in-1 aligner obtains the best results in all corpora. This suggests that it is not necessary to apply two steps in the aligners given that the results did not improve, but made the algorithm more complex.

The aligners proposed in this paper can be downloaded freely[14] because they have been released under the GNU General Public License.[15]

7 Future work

We are developing translation applications that will be trained with the translation units generated by the aligners. They will translate sentences by choosing the best combination of translation units that compose them.

However, it is not clear which one is the best combination of translation units that compose a sentence-pair, although using the most frequent ones appears to give better results. We are researching different ways of combining the harvested translation units to improve the quality of the translation.

Acknowledgements Work funded by the Spanish Government through grant TIC2003-08681-C02-01.

[14] `http://sourceforge.net/projects/tag-aligner`

[15] `http://www.gnu.org/licenses/gpl.html`

Bibliography

Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hin-dle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). A procedure for quantitatively comparing syntactic coverage of english grammars. In *DARPA Speech and Natural Language Workshop*.

Brown, P. F., Lai, J. C., and Mercer, R. L. (1991). Aligning sentences in parallel corpora. In *Proceedings of the 29th Meeting of the Association for Computational Linguistics*, pages 169–176, Berkeley. University of California.

Chen, S. F. (1993). Aligning sentences in bilingual corpora using lexical information. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 9–16, Morristown, NJ, USA. Association for Computational Linguistics.

Gale, W. A. and Church, K. W. (1991). A program for aligning sentences in bilingual corpora. In *Meeting of the Association for Computational Linguistics*, pages 177–184.

Gale, W. A. and Church, K. W. (1993). A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19:75–102.

Melamed, I. D. (1996). A geometric approach to mapping bitext correspondence. In Brill, E. and Church, K., editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–12. Association for Computational Linguistics, Somerset, New Jersey.

Sánchez-Villamil, E., Tomás, J., and Forcada, M. L. (2006). Building parallel text collections for closely related languages. Unpublished.

Sigurd, B., Eeg-Olofsson, M., and van Weijer, J. (2003). Word length, sentence length and frequency - Zipf revisited. *Studia Linguistica*, 58(1):37–52.