Finding parallel texts on the web using cross-language information retrieval

Achim Ruopp
University of Washington,
Seattle, WA 98195, USA
achimr@u.washington.edu

Fei Xia
University of Washington
Seattle, WA 98195, USA
fxia@u.washington.edu

Abstract

Discovering parallel corpora on the web is a challenging task. In this paper, we use cross-language information retrieval techniques in combination with structural features to retrieve candidate page pairs from a commercial search engine. The candidate page pairs are then filtered using techniques described by Resnik and Smith (2003) to determine if they are translations. The results allow the comparison of efficiency of different parameter settings and provide an estimate for the percentage of pages that are parallel for a certain language pair.

1 Introduction

Parallel corpora are invaluable resources in many areas of natural language processing (NLP). They are used in multilingual NLP as a basis for the creation of translation models (Brown et. al., 1990), lexical acquisition (Gale and Church, 1991) as well as for cross-language information retrieval (Chen and Nie, 2000). Parallel corpora can also benefit monolingual NLP via the induction of monolingual analysis tools for new languages or the improvement of tools for languages where tools already exist (Hwa et. al., 2005; Padó and Lapata, 2005; Yarowsky and Ngai, 2001).

For most of the mentioned work, large parallel corpora are required. Often these corpora have limited availability due to licensing restrictions (Tiedemann and Nygaard, 2004) and/or are domain specific (Koehn, 2005). Also parallel corpora are only available for a limited set of language pairs. As a result, researchers look to the World Wide Web as a source for parallel corpora (Resnik and Smith, 2003; Ma and Liberman, 1999; Chen and Nie, 2000). Because of the web's world-wide reach and audience, many websites are bilingual, if not multilingual. The web is therefore a prime candidate as a source for such corpora especially for language pairs including resource-poor languages.

Resnik and Smith (2003) outlined the following three steps for identifying parallel text on the web:

(1) Locating pages that might have parallel translations
(2) Generating candidate page pairs that might be translations
(3) Structural filtering out of non-translation candidate pairs

In most of the previous work, Step (1) is performed in an ad-hoc manner using structural features that were observed in a limited set of samples of parallel pages. For example a language name in an HTML link is considered a strong indication that the page is also available translated to the language indicated by the link. The reason for this ad-hoc approach is that there aren't any standards as to how web developers structure multilingual web pages on a server. Often developers use language names or identifiers in uniform resource locators (URLs) to distinguish different language versions of a page on a server.

When Step (1) is performed using a commercial search engine, another obstacle to finding candidates for parallel pages comes into play: the results are always relevance-ranked for the end user. In this paper, instead of searching exclusively for structural features of parallel pages, we are adding a dictionary-based sampling technique, based on cross-language information retrieval for Step (1). We compare the URL results from each of our ex-

periments with three different matching methods for Step (2). Finally, for Step (3), we adapted a filtering method from Resnik and Smith (2003) to determine whether or not a page pair is a true translation pair.

To estimate the percentage of parallel pages that are available for a certain language pair in relation to the total number of pages available in each of the two languages, we modified a technique that Bharat and Broder (1998) used to estimate overlaps of search engine indices.

We conducted our experiments on the English-German pair, but the described techniques are largely language-independent. The results of the experiments in this paper would allow researchers to choose the most efficient technique when trying to build parallel corpora from the web and guide research into further optimizing the retrieval of parallel texts from the web.

2 Methodology

The first step in finding parallel text on the web has two parts. The first part, the *sampling* procedure, retrieves a set S_1 of pages in the source language L_1 by sending sampling queries to the search engine. These sampling queries are structured in such a way that they retrieve pages that are likely to have translations. The second part, a *checking* procedure, retrieves a set S_2 of pages in target language L_2 that are likely to contain the translations of pages in S_1. The two procedures are described in Sections 2.1 and 2.2, respectively.

Step (2) *matches up* elements of S_1 and S_2 to generate a set of candidates for page pairs that could be translations of each other. This is explained in Section 2.3.

Step (3), a final *filtering* step, uses features of the pages to eliminate page pairs that are not translations of each other. The detail of the step is described in Sections 2.4. Figure 1 illustrates the different sets of pages and page pairs created by the three steps.

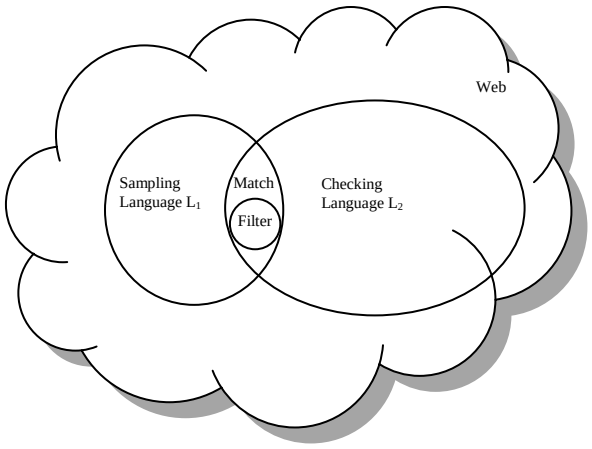

Figure 1. Pages and page pairs involved in the three steps of the algorithm

2.1 Sampling

For the baseline case the sampling should select pages randomly from the search space. To get a random sample of pages from a search engine that we can check for translational equivalents in another language, we select terms at random from a bilingual dictionary.

Instead of using a manually crafted bilingual dictionary, we chose to use a translation lexicon automatically created from parallel data, because the translation probabilities are useful for our experiments. In this study, the translation lexicon was created by aligning part of the German-English portion of the Europarl corpus (Koehn, 2005) using the Giza++ package (Och and Ney, 2003).

The drawback of using this translation lexicon is that the lexicon is domain-specific to parliamentary proceedings. We alleviated this domain-specificity by selecting mainly terms with medium frequency in the lexicon.

We sorted the terms by frequency. According to Zipf's law (Zipf, 1949), the frequency of the terms is roughly inversely proportional to their rank in this list. We choose terms according to a normal distribution whose mean is the midpoint of all ranks. We tuned the deviation to ¼ of the mean, so as to avoid getting very frequent terms into the sample which would just return a large set of unrelated pages, as well as very infrequent terms which would return few or no results.

A single word selected with this normal distribution, together with the `lang:` parameter set to language L_1, is submitted to the search engine to

retrieve a sample (in our experiments 100 pages). The search engine automatically performs stemming on the term.

2.1.1 Source Language Expansion

To obtain a sample yielding more translation candidates, it is valuable to use semantically related multi-word queries for the sampling procedure.

To obtain semantically related words, we used the standard information retrieval (IR) technique of query expansion. Part of the sampling result of single-word queries are summaries, delivered back by the search engine. To come up with a ranked list of terms that are semantically related to the original one-word term, we extract and count all unigrams from a concatenation of the summaries. Stopwords are ignored. After the count, the unigrams are ranked by frequency.

For an n-term query, the original single-word query is combined with the first (n-1) terms of this ranked list to form an expanded query that is submitted to the search engine.

The advantage of this form of expansion is that it is largely language independent and often leads to highly relevant terms, due to the ranking algorithms employed by the search engines.

2.1.2 Language Identifiers in URLs

Once the baseline is established with single and multi-word sampling queries, an additional structural `inurl:` search parameter, which allows querying for substrings in URLs, can be added to increase the likelihood of finding pages that do have translations.

For this paper we limited our experiments to use standard (RFC 3066) two-letter language identifiers for this search parameter: "en" for English and "de" for German.

2.2 Checking

The purpose of the checking procedure is to generate a set of web pages in language L_2 that are potentially translations of pages in the sample obtained in the previous section.

2.2.1 Translating the Sampling Query

The natural way to do this is to translate the sampling query from language L_1 into the target language L_2. The sampling query does not necessarily have a unique one-to-one translation in language L_2. This is where the translation lexicon created from the Europarl corpus comes in. Because the lexicon contains probabilities, we can obtain the m-best translations for a single term from the sampling query.

Given a query in L_1 with n terms and each term has up to m translations, the checking procedure will form up to m^n queries in L_2 and sends each of them to the search engine. Because most current commercial search engines set a limit on the maximum number of queries allowed per day, longer sampling queries (i.e., larger n) mean that fewer overall samples can be retrieved per day. The effect of this trade-off on the number of parallel page pairs is evaluated in our experiments.

Source language expansion can lead to sample terms that are not part of the translation lexicon. These are removed during translation.

If the `inurl:` search parameter was used in the sampling query, the corresponding `inurl:` parameter for language L_2 will be used in the checking query.

2.2.2 Target Language Expansion

An alternative to translating all terms in an expanded, multi-word sampling query (see Section 2.1.1) is to translate only the original single sampling word to obtain top m translations in L_2, and then for each translation do a term expansion on the target language side with (n-1) expansion terms. The benefit of target language expansion is that it only requires m checking queries, where source language expansion requires m^n checking queries. The performance of this different approach will be evaluated in Section 3.

2.2.3 Site Parameter

Another structural search parameter appropriate for checking is the `site:` parameter, which many search engines provide. It allows limiting the query results to a set of pre-defined sites. In our experiments we use the sites of the top-30 results of the sampling set, which is the maximum allowed by the Yahoo! search engine.

2.3 Matching Methods

To obtain page pairs that might be translations of each other, pages in sampling set S_1 are matched up based on URL similarity with pages in corres-

ponding checking set S_2. We experimented with three methods.

2.3.1 Fixed Language List

In the fixed language list matching method, URLs differing only in the language names and language identifiers (as listed in Table 1) are considered a match and added to the set of page pair candidates.

en	de
en-us	de-de
en	ge
enu	deu
enu	ger
english	german
englisch	deutsch

Table 1. Language identifiers and language names for Fixed Language List and URL Part Substitution

An example for a match in this category is http://ec.europa.eu/education/policies/rec_qual/recognition/diploma_en.html and http://ec.europa.eu/education/policies/rec_qual/recognition/diploma_de.html.

2.3.2 Levenshtein Distance

In the Levenshtein distance matching method, if the Levenshtein distance (also known as edit distance) between a pair of URLs from S_1 and S_2 is larger than zero[1] and is below a threshold, the URL pair is considered a match. In our experiments, we set the threshold to four, because for most standard (RFC 3066) language identifiers the maximum Levenshtein distance would be four (e.g. "en-US" vs. "de-DE" as part of a URL).

2.3.3 URL Part Substitution

The third method that we tried does not require querying a search engine for a checking set. Instead, each URL U_1 in the sampling set S_1 is parsed to determine if it contains a language name or identifier at a word boundary. If so, the language name or identifier is substituted with the corresponding language name or identifier for the target language to form a target language URL U_2 according to the substitutions listed in Table 1.

For each resulting U_2, an HTTP HEAD request is issued to verify whether the page with that URL exists on the server. If the request is successful, the pair (U_1,U_2) is added to the set of page pair candidates. If multiple substitutions are possible for a U_1 all the resulting U_2 will be tested.

2.4 Page Filtering

The goal of this step is to filter out all the page pairs that are not true translations.

2.4.1 Structural Filtering

One method for filtering is a purely structural, language-independent method described in Resnik and Smith (2003). In this method, the HTML structure in each page is linearized and the resulting sequences are aligned to determine the structural differences between the two files. Their paper discussed four scalar values that can be calculated from the alignment. We used two of the values in our experiments, as described below.

The first one is called the difference percentage (dp), which indicates the percentage of nonshared material in the page pair. Given the two linearized sequences for a page pair (p_1, p_2), we used Eq (1) to calculate dp, where $length_1$ is the length of the first sequence, and $diff_1$ is the number of lines in the first sequence that do not align to anything in the second sequence; $length_2$ and $diff_2$ are defined similarly.

$$dp\ (p_1, p_2) = \frac{diff_1 + diff_2}{length_1 + length_2} \quad (1)$$

The second value measures the correlation between the lengths of aligned nonmarkup chunks. The idea is that the lengths of corresponding translated sentences or paragraphs usually correlate. The longer a sentence in one language is, the longer its translation in another language should be. For the sake of simplicity, we assume there is a linear correlation between the lengths of the two files, and use the Pearson correlation coefficient as a length correlation metric. From the two linearized sequences, the lengths of nonmarkup chunks are recorded into two arrays. The Pearson correlation coefficient can be directly calculated on these two arrays.

[1] We don't want to match identical URLs.

This metric is denoted as $r(p_1,p_2)$, and its value is in the range of [-1,1]: 1 indicates a perfect positive linear relationship, 0 indicates there is no linear relationship, and -1 indicates a perfect negative linear relationship between the chunk lengths in the two files.

2.4.2 Content Translation Metric

As shown in Resnik and Smith (2003), the structural filtering to judge whether pages are translations of each other leads to very good precision and satisfactory recall.

However, when using the URL part substitution method described in 2.3, many web sites, if they receive a request for a URL that does not exist, respond by returning the most likely page for which there is an existing URL on the server. This is often the page content of the original URL before substitution. Identical pages in the candidate page pair[2] would be judged as translations by the purely structural method and precision would be negatively impacted. There are several solutions for this, one of them is to use a content-based metric to complement the structural metric.

Ma and Liberman (1999) define the following similarity metric between two pages in a page pair (p_1, p_2):

$$c(p_1, p_2) = \frac{Num\ Of\ Translation\ Token\ Pairs}{Num\ Of\ Tokens\ in\ p_1} \quad (2)$$

To calculate this content-based metric, the translation lexicon created in Step (1) comes in handy. For the first 500 words of each page in the page pair candidate, we calculate the similarity metric in Eq (2), using the top two translations of the words in the translation lexicon.

2.4.3 Linear Combination

We combine the structural metrics (*dp* and *r*) and the content-based metric *c* by linear combination:[3]

$$t_{dprc}(p_1, p_2) = \frac{a_{dp} * (1 - dp(p_1, p_2)) + a_r * r(p_1, p_2) + a_c * c(p_1, p_2)}{3} \quad (3)$$

If t_{dprc} is larger than a predefined threshold, the page pair is judged to be a translation.

2.5 Estimating the Percentage of Parallel Pages for a Language Pair

Statistics on what share of web pages in one language have translated equivalents in another language are, to our knowledge, not available. Obtaining these statistics is useful from a web metrics perspective. The statistics allow the calculation of relative language web page counts and serve as a baseline to evaluate methods that try to find parallel pages.

Fortunately there is a statistical method (Bharat and Broder, 1998) that can be adapted to obtain these numbers. Bharat and Broder introduce a method to estimate overlaps in the coverage of a pair of search indices and to calculate the relative size ratio of search indices. They achieve this by randomly sampling pages in one index and then check whether the pages are also present in the other index.

Instead of calculating the overlap of pages in two search engines, we adapted the method to measure the overlap of languages in one search engine. Let P(E) represent the probability that a web page belongs to a set E. Let $P(F_E|E)$ represent the conditional probability that there exist translational equivalents F of E given E. Then

$$P(F_E \mid E) = \frac{Size(F_E)}{Size(E)} \quad (4)$$

$$P(E_F \mid F) = \frac{Size(E_F)}{Size(F)} \quad (5)$$

$Size(F_E)$ and Size(E) can be determined with an experiment using one term samples and checking with a `site:` parameter. Size (F_E) equals the number of page pairs that are determined to be translations by the filtering step. Size(E) is the number of checked sites per sample (30 in the case of Yahoo!) times the number of samples. Size(E_F) and Size(F) are calculated similarly.

3 Experiments

To evaluate the effectiveness of various methods described in Section 2, we ran a range of experiments and the results are shown in Table 2.

[2] The two identical pages could have different URLs.

[3] We use $1-dp(p_1,p_2)$ to turn a dissimilarity measure into a similarity measure.

The first column is the experiment ID; the second column indicates whether source or target expansion is used in Step (1); the third column shows the length of the queries after the expansion (if any). For instance, in Experiment #3, source expansion is applied, and after the expansion, the new queries always have three words: one is the original query term, and the other two are terms that are most relevant to the original query term according to the documents retrieved by the search engine (see Section 2.1.2).

The fourth and fifth columns indicate whether the `inurl:` and `site:` parameters are used during the search. A blank cell means that the search parameter is not used in that experiment. For each query, the search engine returns the top 100 documents.

The next three columns show the numbers of page pairs produced by each of the three matching methods describe in Section 2.3. The last three columns show the numbers of page pairs after the filtering step. Here, we used the linear combination (see Section 2.4.3). All the numbers are the sum of the results from 100 sampling queries. Let us examine the experimental results in detail.

3.1 Sampling and Checking

The evaluation of the sampling and checking procedures are difficult, because the number of translation page pairs existing on the web is unknown. In this study, we evaluated the module indirectly by looking at the translation pairs found by the following steps: the matching step and the filtering step.

A few observations are worth noting. First, query expansion increases the number of page pairs created in Steps (2) and (3), and source and target query expansion lead to similar results. However, the difference between n=2 and n=3 is not significant. One possible explanation is that the semantic divergence between queries on the source side and on the target side could become more problematic for longer queries.

Second, using the `site:` and `inurl:` search parameters (described in 2.1.2 and 2.2.3) increases the number of discovered page pairs. The potential limitation is that `inurl:` narrows the set of discoverable pages to the ones that contain language identifiers in the URL.

Experiment ID	Expansion type	Query length (n)	inurl: Param	site: Param	Number of page pairs (before filtering)			Number of page pairs (after filtering)		
					List	Levenshtein	Substitution	List	Levenshtein	Substitution
1	none	1			5	13	1108	1	1	97
2	Source	2			8	28	1889	3	4	157
3	Source	3			10	42	1975	1	10	124
4	none	1	en/de		58	84	5083	17	22	285
5	Source	2	en/de		72	132	9279	27	31	433
6	Source	3	en/de		100	160	9200	25	31	347
7	none	1			6	18	1099	1	3	92
8	Target	2			4	24	1771	2	3	143
9	Target	3			4	12	1761	0	0	149
10	none	1	en/de		56	93	5041	24	34	281
11	Target	2	en/de		107	161	9131	27	33	426
12	Target	3	en/de		45	72	8395	12	15	335
13	none	1		30	10	258	n/a	6	9	n/a
14	Source	2		30	22	743	n/a	9	32	n/a
15	Source	3		30	46	1074	n/a	12	41	n/a
16	none	1	en/de	30	59	164	n/a	13	15	n/a
17	Source	2	en/de	30	118	442	n/a	28	50	n/a
18	Source	3	en/de	30	171	693	n/a	46	49	n/a

Table 2. Experiment configurations and results

3.2 Matching Methods

Table 2 shows that among the three matching methods, the URL part substitution method leads to many more translation page pairs than the other two methods.

Notice that although the fixed language list method uses the same language name pair table (i.e., Table 1) as the URL part substitution method, it works much worse than the latter. This is largely due to the different rankings of documents in different languages. For instance, suppose a page p_1 in L_1 is retrieved by a sampling query q_1, and p_1 has a translation page p_2 in L_2, it is possible that p_2 will not be retrieved by the query q_2, a query made up of the translation of the terms in q_1.[4]

Another observation is that the Levenshtein distance matching method outperforms the fixed language list method. In addition, it has a unique advantage: the results allow the automatic learning of language identifiers that web developers use in URLs to distinguish parallel pages for certain language pairs.

3.3 Parameter Tuning for Linear Combination of Filtering Metrics

Before the combined metrics in Eq (3) can be used to filter page pairs, the combination parameters need to be tuned on a development set. The parameters are a_{dp}, a_r, and a_c as well as the threshold above which the combined metrics indicate a translated page pair vs. an unrelated page pair.

To tune the parameters, we used data from an independent test run for the en→de language direction. We randomly chose 50 candidate pairs from a set created with the URL part substitution method and manually judged whether or not the pages are translations of each other.

We varied the parameters a_{dp}, a_r, a_c and t_{dprc} over a range of empirical values and compared how well the combined metrics judgment correlated with the human judgment for page translation (we calculated the Pearson correlation coefficient). The results of tuning are shown in Table 3.

a_{dp}	a_r	a_c	t_{dprc}
0.5	1.5	1	> 0.8

Table 3: Parameter and threshold values chosen for linear combination

[4] The search engine returns 100 or fewer documents for each query.

3.4 Evaluation of the Filtering Step

To evaluate the combined filtering method described in Section 2.4.3, we chose 110 page pairs at random from the 433 candidate page pairs in experiment #5 (Language direction en→de, Pairs generated with the URL part substitution method described in 2.3.3). Each of the page pairs was evaluated manually to assess whether it is a true translation pair.

On this set, the combined filter had a precision of 88.9% and a recall of 36.4%. The high precision is encouraging on the noisy test set. The recall is low but is acceptable since one can always submit more sampling queries to the search engine. Resnik and Smith (2003) reported higher precision and recall in their experiments. However, their numbers and ours are not directly comparable because their approach required the existence of parent or sibling pages and consequently their test sets were less noisy.

From the numbers of translation pairs, we can make an estimate of available parallel pages for a language pair, as explained in Section 2.5. For instance, by using the results of experiment #13, the estimate is $P(D_E|E)=0.03\%$ and $P(E_D|D)=0.27\%$ (E for English, and D for German). This indicates that the number of English-German parallel pages is small comparing to the total number of English and German web pages.

4 Conclusion

In this paper we show that despite the fact that there are no standardized features to identify parallel web pages and despite the relevance ranking of commercial search engine results, it is possible to come up with reliable methods to gather parallel pages using commercial search engines. It is also possible to calculate an estimate of how many pages are available parallel in relation to the overall number of pages in a certain language.

The number of translation pages retrieved by the current methods is relatively small. In the future, we plan to learn URL patterns from the Levenshtein matching method and add them to the patterns used in the URL part substitution method. Once more translation pages are retrieved, we plan to use these pages as parallel data in a statistical machine translation (MT) system to evaluate the usefulness of this approach to MT.

Instead of using narrow query interfaces to a public search engine interface, it also might be advantageous to have access to raw indices or crawl data of the engines. Such access will enable us to take advantage of certain page features that could be good indicators of parallel pages.

References

Krishna Bharat and Andrei Broder , 1998, *A technique for measuring the relative size and overlap of public Web search engines*, Computer Networks and ISDN Systems 30(1-7).

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin, 1990, *A statistical approach to machine translation*, Computational Linguist. **16**(2), pp 79-85.

Jiang Chen and Jian-Yun Nie, 2000, *Parallel Web Text Mining for Cross-Language IR, in* Proceedings of RIAO-2000: Content-Based Multimedia Information Access.

William A. Gale and Kenneth W. Church, 1991, *Identifying word correspondence in parallel texts, in* 'HLT '91: Proceedings of the workshop on Speech and Natural Language', NJ, USA, pp. 152-157.

J.W. Hunt and M.D. McIlroy, M.D., 1976, *An Algorithm for Differential File Comparison*, Bell Laboratories, Computer Science Technical Report **41**.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak, 2005, *Bootstrapping Parsers via Syntactic Projection across Parallel Texts*, Special Issue of the Journal of Natural Language Engineering on Parallel Texts **11**(3), pp 311-325.

Philipp Koehn, 2005, *Europarl: A Parallel Corpus for Statistical Machine Translation, in Proceedings of the* 2005 MT Summit.

Xiaoyi Ma and Mark Y. Liberman, 1999, *BITS: A Method for Bilingual Text Search over the Web*, in Proceedings of the 1999 MT Summit, Singapore.

Franz Josef Och and Hermann Ney, 2003, *A Systematic Comparison of Various Statistical Alignment Models*, Computational Linguistics **29**(1), pp 19-51.

Sebastian Padó and Mirella Lapata, 2005, *Cross-linguistic Projection of Role-Semantic Information, in* Proceedings of HLT/EMNLP 2005.

Philip Resnik and Noah A. Smith, 2003, *The Web as a Parallel Corpus*, Computational Linguistics **29**(3), pp 349-380.

Jörg Tiedemann and Lars Nygaard, 2004, *The OPUS corpus - parallel & free, in* 'Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)'.

David Yarowsky and Grace Ngai, 2001, *Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora, in* 'Proceedings of the Second meeting of the North American Chapter of ACL (NAACL 2001)', NJ, USA, pp. 1-8.

George K. Zipf, 1949, *Human Behavior and the Principle of Least-Effort*, Addison-Wesley