PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 2017 39–56

Extracting Parallel Paragraphs from Common Crawl

Jakub Kúdela,[a] Irena Holubová,[a] Ondřej Bojar[b]

[a] Charles University, Faculty of Mathematics and Physics, Department of Software Engineering
[b] Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

Most of the current methods for mining parallel texts from the web assume that web pages of web sites share same structure across languages. We believe that there still exists a non-negligible amount of parallel data spread across sources not satisfying this assumption. We propose an approach based on a combination of bivec (a bilingual extension of word2vec) and locality-sensitive hashing which allows us to efficiently identify pairs of parallel segments located anywhere on pages of a given web domain, regardless their structure. We validate our method on realigning segments from a large parallel corpus. Another experiment with real-world data provided by Common Crawl Foundation confirms that our solution scales to hundreds of terabytes large set of web-crawled data.

1. Introduction

The web is as an ever-growing source of considerable amounts of parallel data that can be mined and included in the training process of machine translation systems. The task of *bilingual document alignment* can be generally stated as follows: Assume we have a set of documents written in two different languages, where a document is a plain text of any length (a sentence, a sequence of multiple sentences, or even a single word). The goal of the task is to collect all pairs of documents in different languages that are mutual translations of each other.

The majority of methods for bilingual document alignment assume that source documents are web pages and they rely on their internal structure or structure of their URLs (e.g. Resnik and Smith, 2003; Esplà-Gomis and Forcada, 2009). By this filtering for similar structure, we can lose a considerable amount of parallel data. Our proposed method is thus not based on page structure comparison. Instead, we search

 Corresponding author: bojar@ufal.mff.cuni.cz
Cite as: Jakub Kúdela, Irena Holubová, Ondřej Bojar. Extracting Parallel Paragraphs from Common Crawl. The Prague Bulletin of Mathematical Linguistics No. 107, 2017, pp. 39–56. doi: 10.1515/pralin-2017-0003.

for parallel paragraphs (or sentences) regardless their organization in the page or in the web site. By moving to these finer units, we have to rely more on the actual content of the paragraphs.

To overcome the well-known problems of text data sparseness, we use bivec (Luong et al., 2015)—a bilingual extension of currently popular word embedding model word2vec (Mikolov et al., 2013). To deal with the possibly large amount of input documents, we make use of recently studied strategies for locality-sensitive hashing (Charikar, 2002; Andoni and Indyk, 2008). Note that any finer alignment of the documents, such as sentence alignment (unless the documents consist of individual sentences), is beyond the scope of our work. However, the methods for obtaining sentence alignment for a document-aligned parallel corpus are well explored (Tiedemann, 2011) and can be easily applied to the output of our method.

Related work has been described by Roy et al. (2016) and Lohar et al. (2016). However, the discussed strategies use the original monolingual word2vec model in contrast to our solution, which makes use of the bilingual bivec model to obtain word vectors in a common vector space. Moreover, neither of the two approaches uses locality-sensitive hashing, which is utilized in our method to achieve better speed performance with regard to scalability.

The rest of the paper is structured as follows: In Section 2 we describe the proposed method. Section 3 provides the results of the experiments and Section 4 concludes and outlines future work.

2. Proposed Method

For our purposes, we refine the specification of bilingual document alignment. Let us assume we have a collection of documents in two languages of interest, organized into *bins*. Each bin holds two sets of documents, one in the source language, the other in the target language, and represents a standalone set of input documents for the original task. For each bin we want to find all the pairs of parallel documents (one in the source language, the other in the target language) present within the bin.

A bin can contain up to millions of documents and it is not required to have a balanced language distribution. Individual bins may vary in size. The binning is a way to restrict the set of considered pairs. No pairs are aligned across different bins, nor between the documents of the same language. The smaller the size of a bin, the better the quality of the resulting alignment (because fewer document pairs need to be considered). It also takes less time and memory to align a smaller bin.

When mining bilingual parallel corpora from the web, we can form a bin for every identified bilingual web domain, simply by taking all paragraphs in the two languages languages of interest, scraped from the web domain. We could be less permissive and create bins spanning several web domains, but at this moment we are leaving this option for the future.

2.1. Training Part I: Bilingual Dictionary, Bilingual Word Vectors

The proposed method is supervised and needs to be trained on an already existing sentence-aligned training parallel corpus (i.e. *seed corpus*) for the language pair we are interested in. For better clarity, we distinguish between two parts of the training process. This section describes the first part, which is depicted in Figure 1. The objective of this part is to preprocess the seed corpus and create a bilingual dictionary together with bilingual word vectors.

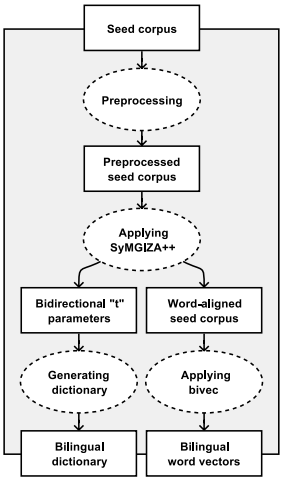

Figure 1: Method: Training part I

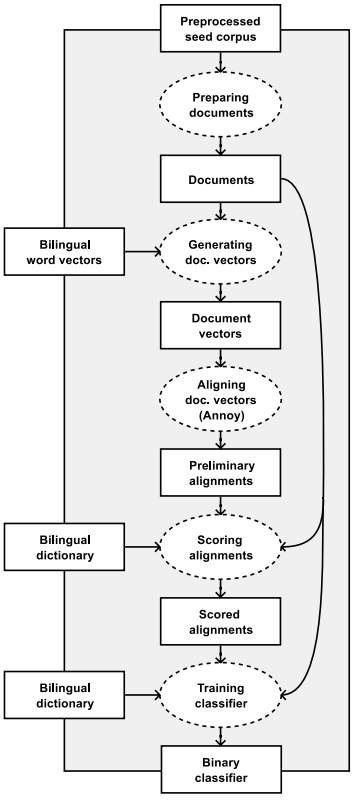

Figure 2: Method: Training part II

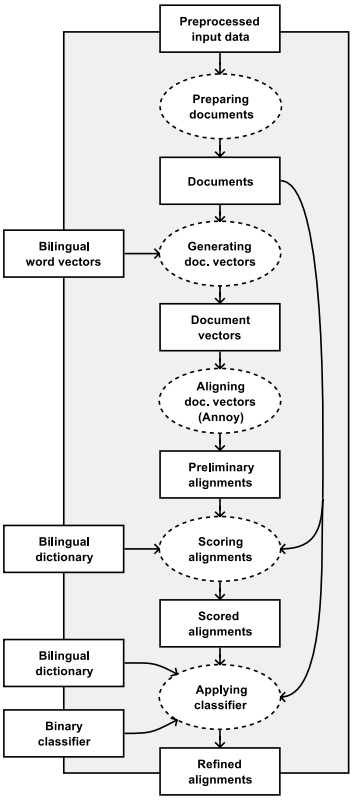

Figure 3: Method: Application

2.1.1. Preprocessing Seed Corpus

The preprocessing may involve tokenization, lemmatization, stemming, truecasing, lowercasing, removing stop words, etc. For individual language pairs, different preprocessing steps might help to gain better results. It is very important that any type of preprocessing done to the seed corpus needs to be also applied to the input data before the alignment process starts.

2.1.2. Applying SyMGIZA++

The resulting corpus from the previous step is further cleaned by removing all such pairs where one of the sentences contains more than 50 tokens or does not contain any letter from any alphabet. Then SyMGIZA++ (Junczys-Dowmunt and Szał, 2012) is executed to obtain a word alignment for the preprocessed and cleaned seed corpus. This step includes preparation of word classes and word co-occurrences which are used in the word alignment process.

SyMGIZA++ is a tool for computing symmetric word alignment models. It is an extension of MGIZA++ (Gao and Vogel, 2008), which is in turn a successor of the historically original program called GIZA++ (Och and Ney, 2003).

2.1.3. Generating Dictionary

The bilingual dictionary is built using the final IBM Model "t" parameters estimated by SyMGIZA++. Each word pair present in both directions having the harmonic mean of the "t" parameters (i.e. *weight*) over a certain threshold is included into the dictionary together with the calculated weight.

2.1.4. Applying bivec

Word embedding is a common name for a set of techniques mapping words or phrases from a vocabulary to distributed word representations in the form of vectors consisting of real numbers in a high-dimensional continuous space. Our method takes advantage of bivec—a bilingual extension of word2vec. It creates bilingual word representations when provided with a word-aligned parallel corpus.

The original word2vec is a group of models producing monolingual word embeddings. These models are implemented in form of neural networks. They are usually trained to reconstruct the contexts of words. A monolingual corpus is needed for the training. The training algorithm iterates over the words in the corpus while considering the surrounding words to be the context of the current word. One word2vec model, called continuous bag-of-words (CBOW), is trained to predict the word when given its context (without the word). Another model, named skip-gram, is trained to predict the context of a given word.

The authors of bivec proposed an extension of the original skip-gram model in the form of a joint bilingual model—bilingual skip-gram. When trained, this model can predict the context of a given word in both languages. In order to train, bivec requires a sentence-aligned parallel corpus and its word alignment. In fact, the word alignment is not strictly necessary, and if it is not provided, the system uses a simple heuristic. However, with the alignment provided, the results are better.

Our method follows the training by further processing the word-aligned seed corpus produced by SyMGIZA++ according to the recommendations of bivec. All the sequences of numbers are replaced with a unified placeholder (the symbol "`0`") and all the unknown symbols (e.g. non-printable Unicode characters) with the specially dedicated `<unk>` tag.

With the word-aligned seed corpus processed, bivec is executed to create the bilingual word vectors (i.e. embeddings) with 40 dimensions. These vectors are known to have a greater cosine similarity for context-related words, even cross-lingually. Although the number of dimensions is an unrestricted parameter, there is a reason why we keep it relatively low. The word vectors are used to calculate the aggregate document vectors with the same number of dimensions. The document vectors are then indexed using Annoy[1]—an implementation of approximate nearest neighbors search. The documentation of Annoy suggests that it works best with the number of dimensions less than 100. On the other hand, the authors of bivec conducted the tests using 40, 128, 256 and 512 dimensions. We have decided to use the only number of dimensions suitable for Annoy that has been tested with bivec.

2.2. Training Part II: Binary Classifier

The second part of the training process is illustrated in Figure 2. In this part, the method attempts to find candidate sentence pairs by realigning the preprocessed seed corpus. This is performed by employing the bilingual word vectors and locality-sensitive hashing. While working with the seed corpus, we know which of the candidate sentence pairs are correct and we exploit this knowledge to train a binary classifier. The trained classifier is then used when applying the trained method on the input data.

2.2.1. Preparing Documents

The method splits all the pairs of sentences from the seed corpus (i.e. documents) into a set of equally large bins. The last bin can be an exception. The size of a bin should be an estimate of its expected size in a real-world use case. In our experiments, we split the corpus into training bins consisting of 50,000 pairs of parallel documents, i.e. 100,000 individual documents. We believe that this amount of documents is a

[1] `https://github.com/spotify/annoy`

good upper-bound estimate of the total number of paragraphs in either of the two languages located on a typical bilingual web domain.

2.2.2. Generating Document Vectors

For each document, an associated vector is generated using the bilingual word vectors obtained in the first part of the training. To calculate the document vector, we utilize the tf-idf (i.e. term frequency - inverse document frequency) weighting scheme. For every unique document $d = (d_1, d_2, \ldots, d_n)$ a vector is generated as:

$$\text{doc\_vector}(d) = \sum_{i=1}^{n} \text{tf-idf}(d_i, d) \times \text{word\_vector}(d_i) \quad (1)$$

where $\text{tf-idf}(d_i, d)$ is a tf-idf of the term d_i in the document d and $\text{word\_vector}(d_i)$ is the bivec word vector for the term d_i. If a word vector does not exist for a given term, a zero vector is used instead.

In a smaller-scale experiment performed with our method, the tf-idf weighting scheme was compared to a plain sum of the word vectors with an equal weight. The method yielded comparatively better results using the tf-idf scheme.

2.2.3. Aligning Document Vectors (Annoy)

For each bin, independently of other bins, the following procedure is performed. A search index is built containing the vectors of all the documents in the target language. To build the search index, the method uses Annoy operating with the angular distance. Then, for every document in the source language, the index is searched to obtain k approximate nearest neighbors to its vector. This returns a list of candidate parallel documents in the target language to the document in the source language. We call them *preliminary alignments*.

Annoy is an implementation of approximate nearest neighbors search. Unlike the exact search methods, it does not guarantee to find the optimum, but in many cases it actually does. This relaxation enables it to require less resources. In particular, it needs less time, which becomes useful when dealing with larger amounts of data. Internally, Annoy uses random projections to build up a forest of search trees—an index structure for the searching process. The algorithm for building the index uses SimHash, which is a locality-sensitive hashing method.

2.2.4. Scoring Alignments

Within the preliminary alignments, the top candidates are not necessarily the optimal ones. Therefore, our method applies a scoring function to reorder the candidates

creating the *scored alignments*. This increases the probability of the optimal documents to appear higher in their candidate lists. Given the document $d = (d_1, d_2, \ldots, d_n)$ and its candidate document pair $c = (c_1, c_2, \ldots, c_m)$, where d_i and c_i are the individual terms in the respective documents, the scoring function is defined as:

$$\text{score}(d, c) = \text{length\_similarity}(d, c) \times \text{weight\_similarity}(d, c) \tag{2}$$

Both the functions length_similarity(d, c) and weight_similarity(d, c) have the range of $[0, 1]$. The idea is that the higher the result they return, the greater the possibility that the pair is parallel. The length_similarity(d, c) function compares the ratio of the documents' lengths. It is based on the probability density function of the Gaussian (normal) distribution:

$$\text{length\_similarity}(d, c) = e^{-\frac{\left(\frac{\text{length}(c)}{\text{length}(d)} - \mu\right)^2}{2\sigma^2}} \tag{3}$$

where $\frac{\text{length}(c)}{\text{length}(d)}$ is the actual ratio of the documents' lengths (i.e. total number of characters) and μ is the expected ratio with the standard deviation σ. The expected ratio with its standard deviation can be estimated using the pairs of parallel sentences from the preprocessed seed corpus. The other function weight_similarity(d, c) is based on the IBM Model 1 (Brown et al., 1993) and uses the bilingual dictionary created in the first part of the training. It is defined as:

$$\text{weight\_similarity}(d, c) = \prod_{i=1}^{n} \sum_{j=1}^{m} \frac{\text{weight}(d_i, c_j)}{m} \tag{4}$$

where weight(d_i, c_j) is the weight of the word pair $\langle d_i, c_j \rangle$ provided by the dictionary if the word pair entry exists, otherwise it equals 10^{-9} (i.e. "null weight").

2.2.5. Training Classifier

We use a binary classifier to decide whether to accept a proposed pair of documents as parallel or not. The chosen model for the classifier is a feed-forward neural network trained by back-propagating errors (Rumelhart et al., 1986). The method uses an implementation provided by PyBrain (Schaul et al., 2010). The classification is based on 4 features. All of these features have the range of $[0, 1]$. Given the document $d = (d_1, d_2, \ldots, d_n)$ and its candidate document pair $c = (c_1, c_2, \ldots, c_m)$, the following text describes all the features.

The first feature length_similarity(d, c) has been already defined in Section 2.2.4. It scores the ratio of the documents' lengths against the expected ratio. The second feature length_confidence(d, c) provides a supplementary information for the first one, which is neither reliable, nor effective when scoring pairs of short documents; however, it is substantial when comparing pairs of long documents:

$$\text{length\_confidence}(d, c) = 1 - e^{-0.01 \times \text{length}(d)} \tag{5}$$

This is a monotonically increasing function providing the model with an information of absolute length of the document d. The higher the length_confidence(d, c) is, the more authoritative the score of the length_similarity(d, c) should be deemed.

The third feature weight_similarity$_2(d_i, c_j)$ is a modified version of the one already defined (i.e. weight_similarity(d, c)). The original version was tested for the purposes of the classification, but the results were poor. The ineffectiveness could be caused by the fact that the original function lacks some proper normalization with respect to documents' sizes and returns extremely small values when comparing larger documents. The modified version is defined as follows:

$$\text{weight\_similarity}_2(d, c) = \frac{\sum_{i=1}^{n} \text{length}(d_i) \times \max_{j=1}^{m} (\text{weight}_2(d_i, c_j))}{\sum_{i=1}^{n} \text{length}(d_i) \times \text{sgn}(\max_{j=1}^{m} (\text{weight}_2(d_i, c_j)))} \tag{6}$$

where length(d_i) is the length of the term d_i and weight$_2(d_i, c_j)$ is defined as the weight of the word pair $\langle d_i, c_j \rangle$ provided by the dictionary if the entry exists; however, if the entry does not exist and the two words are identical then it equals 1, otherwise it returns 0.

Let us explain the reason for the heuristic of weight$_2(d_i, c_j) = 1$ for a pair of identical words not having an entry present in the dictionary. The same set of features is used when applying the trained method on the input data. At that moment, occurrences of new words or special terms (e.g. URLs or email addresses) are expected. The heuristic considers a pair of identical words to be a perfect translation only if the dictionary does not contain other relation.

Moreover, the weights are multiplied by the lengths of words due to an assumption that longer words are usually less frequent, carry more meaning, therefore are more important for the sentence. The definition of weight_similarity$_2(d, c)$ is an arithmetic mean of strongest relations between a source word from d and any of the target words from c, weighted by the lengths of source words. We can interpret weight_similarity$_2$ as: "Given our incomplete word translation dictionary, how likely are these two documents parallel?"

The last feature weight_confidence$_2$ supplements the third one and it can be interpreted as: "To what extent does the dictionary cover the pairs of words in these documents?". The formal definition is the following:

$$\text{weight\_confidence}_2(d, c) = \frac{\sum_{i=1}^{n} \text{length}(d_i) \times \text{sgn}(\max_{j=1}^{m} (\text{weight}_2(d_i, c_j)))}{\sum_{i=1}^{n} \text{length}(d_i)} \tag{7}$$

The process of training of the binary classifier starts by creating a supervised dataset using the scored alignments. For every document in the source language and its top candidate in the target language, a pair of input→output vectors is added into the supervised dataset as follows:

$$\begin{pmatrix} \text{length\_similarity}(d, c) \\ \text{length\_confidence}(d, c) \\ \text{weight\_similarity}_2(d, c) \\ \text{weight\_confidence}_2(d, c) \end{pmatrix} \rightarrow \begin{cases} \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \text{if } \langle d, c \rangle \text{ are parallel} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \text{otherwise} \end{cases} \tag{8}$$

The input vector consists of the 4 defined features, while the output vector encodes whether the documents $\langle d, c \rangle$ are parallel or not. The first value of the output vector represents the probability of the documents to be non-parallel. The second value is complementary to the first one. Before the network is trained, the collected supervised dataset is subsampled to contain an approximately equal number of items representing parallel and non-parallel document pairs. This helps the network to be less biased by the ratio of parallel and non-parallel pairs in the supervised dataset. At this moment, it is also possible to reduce the size of the dataset to shorten the time it takes to complete the training.

For completeness, let us describe the configuration of the network. It has 4 input, 16 hidden and 2 output neurons. The hidden neurons are arranged in a single layer. The input neurons are linear, the hidden layer uses the sigmoid function and the output layer uses the softmax function.

2.3. Application

The process of applying the trained method on the input data is illustrated in Figure 3. It is almost identical with the procedure of the second part of the training (described in Section 2.2). Due to this similarity, the following text does not cover the shared parts.

The input documents have to be preprocessed in the same way as the seed corpus during the training. Then, the preprocessed documents have to be split into bins. When aligning paragraphs from the web, a bin can contain all the paragraphs for both the languages scraped from one bilingual web domain. In this scenario, the names of the web domains can be used as bin identifiers.

2.3.1. Applying Classifier

With the input dataset prepared, the process follows with the same steps as in the second part of the training. First, vectors are generated for all the documents. Then, the document vectors are aligned by searching for nearest neighbors of all the documents in the source language, resulting in preliminary alignments. In the final step, the trained classifier is used to obtain *refined alignments*. For every document in the source language and its top candidate in the target language the trained network is activated in the same way it has been trained. The second value of the output vector represents the confidence that the two documents are parallel. If the confidence is greater than a user-defined threshold, the document pair ends up in the resulting refined alignments (i.e. *extracted corpus*).

3. Experiments

Our experiments are solely focused on the Czech–English language pair. The first experiment is carried out using CzEng 1.0 (Bojar et al., 2012)—a Czech–English sentence-aligned parallel corpus. By realigning the CzEng 1.0 corpus, we can evaluate the quality of the results automatically. The second experiment uses more realistic and noisy data provided by Common Crawl Foundation.[2] The organization produces and maintains an open repository of web-crawled data that is universally accessible and analyzable.

3.1. Prealigned Data (CzEng 1.0) Experiment

The corpus consists of all the training sections (packs 00–97) of CzEng 1.0 in the plain text, untokenized format. It includes 14,833,358 pairs of parallel sentences collected from various domains (e.g. fiction, legislation, movie subtitles, parallel web pages, etc.). By default, the pairs are shuffled, meaning they are not grouped by their domains. The shuffled corpus is split exactly in half into a *head* (i.e. seed corpus) for training and a *tail* for evaluation. The whole procedure is illustrated in Figure 4.

The preprocessing step consists of tokenization and lowercasing. The head is cleaned by excluding all such pairs where one of the sentences contains too many tokens or does not contain any letter from any alphabet. The tail is cleaned by only

[2] http://commoncrawl.org/

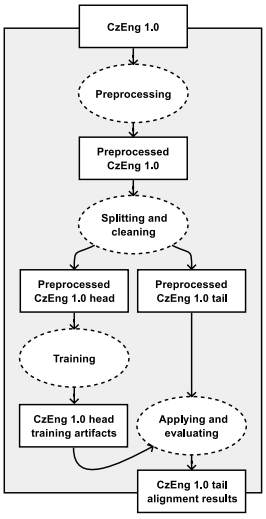

Figure 4: Prealigned Data (CzEng 1.0) Experiment

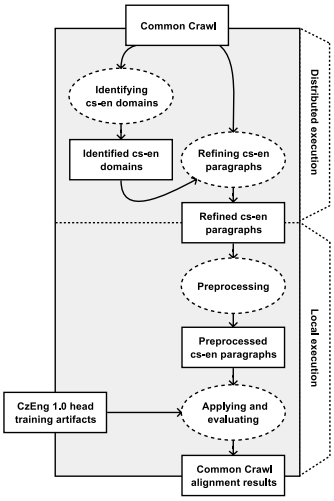

Figure 5: Web Data (Common Crawl) Experiment

applying the latter of the two mentioned criteria. The pairs containing overly long sentences are removed from the head to get better quality word alignment.

During the training, SyMGIZA++ uses the union method for the final symmetrization of the word alignments, while bivec uses the bilingual skip-gram model and runs for 10 iterations. The pairs of parallel sentences from the head are distributed into a set of artificial bins to form a dataset for the training of the classifier. Each bin contains 50,000 pairs of parallel documents from various domains, i.e. 100,000 individual documents. As discussed above, we believe this amount of documents to be a good upper-bound estimate of the total number of paragraphs in either of the two languages (Czech and English) located on a typical Czech–English web domain. Annoy builds search indexes with 500 trees and during each search it inspects up to 20,000 nodes to return 20 candidates. The classifier is trained using approximately 20% of all the available pairs of Czech documents with their corresponding top English candidates. Additionally, the supervised dataset contains nearly as many parallel pairs as non-parallel. The classifier's network is trained for 20 epochs with 1% learning rate.

The trained method is used to realign the tail. The pairs of parallel sentences from the tail are distributed into artificial bins in the same manner as those from the head. In this scenario, each bin simulates a web domain with 50,000 Czech and 50,000 English paragraphs that we want to align. In contrast to real-world websites, these are

perfectly parallel, meaning that all the content is available in both languages. The original alignment of the tail is forgotten and it does not affect the evaluation process is any way. The confidence threshold of the classifier is set to 50%. The refined alignments represent a subset of all the pairs of Czech documents with their top English candidates that the classifier accepts to be parallel.

In the preliminary alignments of the tail, 50.30% of all the top candidates are exact matches. However, this ratio is more satisfactory in the scored alignments—71.30%. This means that the scoring of the preliminary alignments is an important step in the whole process. The overall effectiveness of our method (after the application of the binary classifier), when realigning the tail part of CzEng 1.0, is listed in Table 1. Of all the existing pairs of parallel sentences 63.02% were detected, and 93.74% of all the detected pairs were correct.

Recall (%)	63.02
Precision (%)	93.74

Table 1: Prealigned Data (CzEng 1.0); Experiment: Effectiveness

The computer used for the execution has Intel® Xeon® CPU E5-2630 v3 (20 MB Cache, 2.40 GHz) and 128 GB of memory. Table 2 lists the approximate time durations of the individual steps of the experiment.

3.2. Web Data (Common Crawl) Experiment

The second experiment deals with the non-parallel, real-word, noisy data acquired from the web. The language pair of our interest is again Czech–English. The procedure uses the training artifacts created in the first experiment, namely the dictionary, bilingual word vectors and the trained classifier.

The input data are obtained from the July 2015 dataset provided by Common Crawl Foundation and consist of approximately 1.84 billions of crawled web pages, taking about 149 TB of disk space in an uncompressed format. To store and process this large volume of data we use Hadoop—an HDFS (Shvachko et al., 2010) cluster and the MapReduce (Dean and Ghemawat, 2004) framework.

The procedure of the experiment is illustrated in Figure 5. It starts with a distributed execution running two MapReduce jobs. The first job creates a list of web domains containing at least some Czech and English paragraphs, i.e. contents of <p> HTML tags. Parsing of the HTML is done using jsoup[3] and language detection is per-

[3] https://jsoup.org/

Activity	Duration (hh:mm)
Preprocessing	
Tokenizing and lowercasing	00:08
Splitting and cleaning	00:05
Training part I	
Applying SyMGIZA++	13:21
Generating dictionary	00:10
Applying bivec	01:01
Training part II	
Generating document vectors	00:37
Aligning document vectors (Annoy)	05:52
Scoring alignments	02:49
Training binary classifier	01:29
Application	
Generating document vectors	00:45
Aligning document vectors (Annoy)	07:04
Scoring alignments	04:10
Applying binary classifier	00:47

Table 2: Prealigned Data (CzEng 1.0); Experiment: Time Duration

formed by language-detector.[4] Due to the unsatisfactory effectiveness of language-detector on shorter texts, paragraphs having less than 100 characters are discarded. In the future, the parsing could be modified to consider also other HTML tags.

The list of web domains is further filtered to keep only those having the ratio of Czech to English paragraphs within the interval $(0.01, 100)$. This filtering discards all domains with very unbalanced language distribution. The output of the first job contains $8,750$ identified bilingual domains.

The second MapReduce job extracts the Czech and English paragraphs for all the identified bilingual web domains. In order to provide the second job with the file containing the accepted domains, Hadoop Distributed Cache is utilized. The output of the second job contains $5,931,091$ paragraphs for both languages, namely $801,116$ Czech and $5,129,975$ English. These paragraphs are aligned with our method in a local execution and the results are evaluated. All the settings remain the same as in the first experiment, except the threshold of the classifier is changed to 99%. The precision is favored over the recall. The extracted paragraph-aligned parallel corpus contains

[4] https://github.com/optimaize/language-detector

114,771 pairs from 2,178 domains, having in total 7,235,908 Czech and 8,369,870 English tokens. Table 3 lists the most frequent web domains contributing to the extracted corpus. The size of the extracted corpus is comparable with the amount of Czech–English parallel data acquired by the related project focused on mining the Common Crawl datasets (Smith et al., 2013).

The quality of the extracted corpus is evaluated manually on a set of 500 randomly selected paragraph pairs. The inspected pairs are categorized into the categories displayed in Table 4. A pair of paragraphs is considered to be a human translation if it seems like created by a human. If the translation of the pair seems cumbersome, it is labeled as a product of machine translation. A partial match represents a situation, when one paragraph is incomplete regarding the content of the other one. Everything else is labeled as a mismatch. If we consider the pairs belonging to the two categories of human and machine translation as true positives, then the estimation of precision is 94.60%.

Source Domain	**Paragraph Pairs**	**Ratio (%)**
europa.eu	23457	20.45
eur-lex.europa.eu	15037	13.11
windows.microsoft.com	11905	10.38
www.europarl.europa.eu	8560	7.46
www.project-syndicate.org	2210	1.93
www.debian.org	2191	1.91
support.office.com	1908	1.66
www.esa.int	1308	1.14
www.eea.europa.eu	1299	1.13
www.muni.cz	1206	1.05
⋮	⋮	⋮
Total	114,711	100.00

Table 3: Web Data (Common Crawl); Experiment: Web Domains

Manual evaluation of the method's recall is complicated and therefore it is performed using only one selected web domain—`www.csa.cz`, the official website of Czech Airlines. The input dataset contains 68 Czech and 87 English paragraphs for this domain. These paragraphs are manually aligned, creating the *desirable alignments*. When evaluated, the desirable alignments contain 44 paragraph pairs, of which 42 also appear in the corpus extracted by our method. Additionally, the extracted corpus include 1 extra pair subjectively regarded as a mismatch. Table 5 shows the effectiveness of our method evaluated for the `www.csa.cz` web domain.

Category	Count	Ratio (%)
Human translation	466	93.20
Machine translation	7	1.40
Partial match	13	2.60
Mismatch	14	2.80
Total	500	100.00

Table 4: Web Data (Common Crawl); Experiment: Evaluation (500 Paragraph Pairs)

Recall (%)	95.45
Precision (%)	97.67

Table 5: Web Data (Common Crawl)
Experiment: Effectiveness (`www.csa.cz`)

The Hadoop cluster used for the distributed execution of the two MapReduce jobs consists of 3 management nodes and 24 worker nodes. The management nodes run components like front-end, HDFS NameNode and MapReduce History Server. Every node of the configuration has Intel® Xeon® CPU E5-2630 v3 (20 MB Cache, 2.40 GHz) and 128 GB of memory. The total disk space available on the cluster is 1.02 PB. The HDFS operates with a replication factor of 4. The rest of the procedure, i.e. the local execution, is done on one node of the cluster. Table 6 contains the approximate time durations of the individual steps of the experiment.

4. Conclusions and Future Work

The majority of methods for bilingual document alignment search for pairs of parallel web pages by comparing the similarity of their HTML structures. Our method does not depend on any kind of page structure comparison. We are able to efficiently identify pairs of parallel segments (i.e. paragraphs) located anywhere on the pages of a web domain, regardless of their structure.

To verify the idea of our method, we have performed two experiments focused on the Czech–English language pair with both prealigned and real-world data. These experiments show satisfactory results, implying that the proposed method is a promising baseline for acquiring parallel corpora from the web.

Nevertheless, there is still some room for improvement. First of all, our method does not consider word order at any stage during the aligning process. The scoring function and the features of the classifier could be extended to take word order into ac-

Activity	Duration (hh:mm)
MapReduce framework	
Identifying cs-en domains	11:58
Refining cs-en paragraphs	11:38
Local Execution	
Tokenization and lowercasing	00:09
Generating document vectors	00:58
Aligning document vectors (Annoy)	01:13
Scoring alignments	03:42
Applying classifier	00:39

Table 6: Web Data (Common Crawl); Experiment: Time Duration

count. Then, there is an asymmetric nature of our method, meaning that it yields different results if the source and the target languages are swapped. The method could perform the alignment for both directions and the results could be symmetrized. This might help to achieve an even higher precision.

Another possibility would be to extend our method with some kind of structural comparison, for instance, in form of a new feature for the classifier, that would compare the structural origin of the input documents (e.g. XPath of <p> tags, in case of aligning paragraphs from the web).

Finally, we have used our method only in a single-node environment so far. This is largely because we have worked with relatively small sets of documents (not more than $15,000,000$). However, the method is designed to be able to run in distributed fashion. Bins with input documents represent independent and isolable tasks. With the method trained in a local execution, these tasks could be distributed across multiple nodes of a cluster. This could increase the throughput of our method, and hence decrease the overall execution time.

Acknowledgements

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 644402 (HimL). This work has been using language resources stored and distributed by the LINDAT/CLARIN

project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Bibliography

Andoni, Alexandr and Piotr Indyk. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM*, 51(1):117–122, Jan. 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327494. URL http://doi.acm.org/10.1145/1327452.1327494.

Bojar, Ondřej, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12)*, pages 3921–3928, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association. ISBN 978-2-9517408-7-7.

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.*, 19(2):263–311, June 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972474.

Charikar, Moses S. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM. ISBN 1-58113-495-9. doi: 10.1145/509907.509965. URL http://doi.acm.org/10.1145/509907.509965.

Dean, Jeffrey and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1251254.1251264.

Esplà-Gomis, Miquel and Mikel L. Forcada. Bitextor, a free/open-source software to harvest translation memories from multilingual websites. In *Proceedings of the workshop Beyond Translation Memories: New Tools for Translators MT*, Ottawa, Ontario, Canada, August 2009.

Gao, Qin and Stephan Vogel. Parallel Implementations of Word Alignment Tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-932432-10-7. URL http://dl.acm.org/citation.cfm?id=1622110.1622119.

Junczys-Dowmunt, Marcin and Arkadiusz Szał. SyMGiza++: Symmetrized Word Alignment Models for Machine Translation. In Bouvry, Pascal, Mieczyslaw A. Klopotek, Franck Leprévost, Malgorzata Marciniak, Agnieszka Mykowiecka, and Henryk Rybinski, editors, *Security and Intelligent Information Systems (SIIS)*, volume 7053 of *Lecture Notes in Computer Science*, pages 379–390, Warsaw, Poland, 2012. Springer. URL http://emjotde.github.io/publications/pdf/mjd2011siis.pdf.

Lohar, Pintu, Debasis Ganguly, Haithem Afli, Andy Way, and Gareth J.F. Jones. FaDA: Fast Document Aligner using Word Embedding. *The Prague Bulletin of Mathematical Linguistics*, 106(1):169–179, Oct. 2016. ISSN 0891-2017. doi: 10.1515/pralin-2016-0016. URL https://doi.org/10.1515/pralin-2016-0016.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. Bilingual Word Representations with Monolingual Quality in Mind. In *NAACL Workshop on Vector Space Modeling for NLP*, Denver, United States, 2015.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

Och, Franz Josef and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Comput. Linguist.*, 29(1):19–51, Mar. 2003. ISSN 0891-2017. doi: 10.1162/089120103321337421. URL http://dx.doi.org/10.1162/089120103321337421.

Resnik, Philip and Noah A. Smith. The Web As a Parallel Corpus. *Comput. Linguist.*, 29(3): 349–380, Sept. 2003. ISSN 0891-2017. doi: 10.1162/089120103322711578. URL http://dx.doi.org/10.1162/089120103322711578.

Roy, Dwaipayan, Debasis Ganguly, Mandar Mitra, and Gareth J. F. Jones. Representing Documents and Queries as Sets of Word Embedded Vectors for Information Retrieval. *CoRR*, abs/1606.07869, 2016. URL http://arxiv.org/abs/1606.07869.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038%2F323533a0.

Schaul, Tom, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

Shvachko, Konstantin, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop Distributed File System. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-1-4244-7152-2. doi: 10.1109/MSST.2010.5496972. URL http://dx.doi.org/10.1109/MSST.2010.5496972.

Smith, Jason R., Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt Cheap Web-Scale Parallel Text from the Common Crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1383, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-1135.

Tiedemann, J. *Bitext Alignment*. Synthesis digital library of engineering and computer science. Morgan & Claypool, 2011. ISBN 9781608455102. URL https://books.google.cz/books?id=IMCIGSMB5k0C.

Address for correspondence:
Ondřej Bojar
bojar@ufal.mff.cuni.cz
ÚFAL MFF UK (Linguistics)
Malostranské náměstí 25
11800 Praha
Czech Republic