Word Clustering Approach to Bilingual Document Alignment (WMT 2016 Shared Task)

Vadim Shchukin[1,2] **Dmitry Khristich**[2] **Irina Galinskaya**[2]
[1]Yandex School of Data Analysis,
[2]Yandex
{rj42,khristich,galinskaya}@yandex-team.ru

Abstract

Our participation in Bilingual Document Alignment shared task at WMT16 focuses on building a language-independent, scalable system for aligning documents based on content as opposed to using webpage meta information. The resulting system is capable of producing scored n-best lists of candidate pages and can therefore be adapted to tasks where either precision or recall is maximized. We conduct a series of experiments that show the effectiveness of the system without any specific tuning.

1 Introduction

Training statistical machine translation systems involves using two kinds of textual data: mono- and bilingual. While mining monolingual data is rather straightforward, determining pairs of parallel documents is a rather complicated task for a variety of reasons.

First of all, the largest source of text documents — the World Wide Web — has most of its parallel data in an unstructured form, meaning that it is often impossible to determine parallel pairs using meta info only. While a set of documents within a particular webdomain may be structured, the structure itself varies between domains and is therefore hard to exploit. This lack of stucture in the Web forces a mining system to compare every source language document to every target language document from the corpus, thus leading to quadratic complexity and making such straightforward algorithms not applicable to mining parallel data from large web corpora containing billions of documents.

Existing parallel data mining approaches deal with these problems in different ways.

Methods focused on meta info such as document URL (Resnik and Smith, 2003), publication dates or document structure, may work well on small structured corpora but suffer from sparsity and unreliability of meta info in the Web. One of the advantages of such methods is a lesser computational complexity — simple URL matching, for example, can be performed in linear time and doesn't even require to store HTML bodies as it only operates on URLs.

Another approach is to analyze document contents only, making zero assumptions about the document structure or meta info. This approach is more versatile but at the same time more resourse-demanding and tends to suffer from bad scalability. Applying it to big Web corpora requires implementation of special techniques that reduce the quadratic complexity of a naive algorithm to something manageable, preferrably making the number of document comparisons linear.

2 Previous work

Our approach is based on two papers working with different aspects of content-based document alignment. The first of them (Uszkoreit et al., 2010) aims at reducing the amount of pairwise comparisons of documents, while the other (Fukushima et al., 2006) speeds up the comparisons themselves. We describe both methods below.

2.1 Shingles and near-duplicate detection

Uszkoreit et al. (2010) describe a large scale parallel data mining method.

First, the system transforms a given multilingual input corpus into a monolingual one by translating every document into English using a baseline statistical machine translation system.

After that, candidates of parallel document pairs are extracted by applying a near-duplicate detection algorithm to the translated corpus. This re-

Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 740–744,
Berlin, Germany, August 11-12, 2016.

quires two different sets of n-grams (shingles) to be extracted from each document:

- **Matching n-grams** are used to construct the candidate sets, meaning that the system only considers pairs of documents that have at least one common matching n-gram. The key trick here is that we discard every matching n-gram whose frequency exceeds some fixed threshold. If the order of matching n-grams is sufficiently large, this operation prunes only a small fraction of the matching n-grams, and most importantly makes the number of pairwise document comparisons linear.

- **Scoring n-grams** are used only in the computation of a score for a given pair of documents. Every scoring n-gram is assigned a score equal to its inverse global document frequency in the input corpus. As the score of an n-gram is inversely proportional to its frequency, scoring n-grams with very high frequencies may be safely pruned, increasing performance. The score of a pair of documents is computed as cosine similarity of two corresponding vectors in the vector space of scoring n-grams.

In the next stage, candidate sets are built using matching n-grams, then pairs of documents from every set are scored using scoring n-grams, producing scored n-best lists for every document.

In the final stage, pairs are symmetrized, leaving only those where each document is a part of the other's n-best list.

The described method scales well as all steps can be parallelized, has linear computational complexity and provides high quality on big unstructured collections of documents. However, its quality is dependent on the quality of the baseline machine translation system and using a high quality baseline usually makes the first step — translation of every document in corpus — a very computationally complex task.

2.2 Word clustering

Fukushima et al. (2006) present an approach to the task of judging whether a pair of texts is parallel or not. The proposed algorithm scores a pair of documents based on the number of word pairs from the documents that are mutual translations of each other.

In the first step, the algorithm maps every noun from both languages to a special 'semantic ID' (non-nouns are ignored). The goal is to assign the same ID to every pair of words that are translations of each other.

To assign semantic IDs, the algorithm builds a word graph using a bilingual dictionary: nodes represent words and edges connect pairs of words that are translations of each other. Then, a threshold on the size of a connected component is selected and every component larger than the threshold is recursively divided into two smaller parts with an equal number of nodes. The process continues until every component is smaller than the threshold.

Graph partitioning is performed using a simple greedy algorithm. For a given connected component, it divides nodes into two equal groups such that the number of edges between the groups is minimized.

After the partitioning is complete, every component is assigned a unique semantic ID.

In the next step, every document from the corpus is preprocessed, converting each word to its corresponding semantic ID. The converted representations are then used to compare pairs of documents.

The method is reported to significantly speed up the document comparison without losing accuracy.

One of the disadvantages of this method is that it treats all edges of the word graph equally, while in reality some of the translations are more probable, and therefore more valuable than the others.

3 Our approach

The outline of our method is as follows. First, we run a bilingual word clustering algorithm similar to the one described in Section 2.2. Then, we preprocess the bilingual input corpus converting each word to its cluster ID. This operation produces a 'monolingual' corpus in a 'language' of cluster IDs which we then use as input data for the near-duplicate detection algorithm described in Section 2.1, thus skipping the computationally expensive step of machine-translating the entire input corpus.

Our approach to the bilingual word clustering problem is described in detail below.

3.1 Weighted word clustering

To form word clusters, we require a phrasetable of the corresponding translation direction as input

data. This phrasetable can be built from the parallel data mined using some simple baseline method like URL mathching or, alternatively, the previous iteration of our algorithm.

In the first stage, we filter the phrasetable keeping only phrases where both source and destination parts consist of a single word. The result is used to form a graph with words as nodes and phrases as edges. Previously, Fukushima et al. (2006) used a dictionary as input and built an unweighted word graph. Our approach is to make a weighted graph using statistics from the phrasetable, namely phrase observation counts:

- $N_{src}(f)$ — the count of the source phrase f,
- $N_{tgt}(e)$ — the count of the target phrase e,
- $N(f, e)$ — the co-occurence count of the source phrase f and the target phrase e.

The resulting graph will most likely have one giant connected component containing most of the graph's vertices. Therefore, to form meaningful word clusters some of the edges have to be removed. We propose to use a variation of layered graph clustering algorithm (Algorithm 1).

It is an iterative process that takes some graph G as input and examines all connected components one by one. If the current component satisfies some fixed clustering criterion, a new word cluster is formed, assigned a unique ID and the component is removed from the graph. Otherwise, it takes a fraction of the edges of the current graph that have the worst weights, removes them, and runs recursively on the new graph. The process continues until the graph is empty.

Removing a constant fraction of edges during every step makes the complexity of the algorithm linear: $\Theta(E)$, where E is the number of the edges in the graph, i.e. the number of single word phrases in the input phrase table.

Whether the algorithm is capable of producing word clusters that have as many related (and as few unrelated) words as possible, depends on the choice of the weighting function and the connected component criterion. The weighting function that worked well during our experiments on various data, is as follows:

$$weight(f, e) = \frac{N^2(f, e)}{N_{src}(f) \cdot N_{tgt}(e)} \quad (1)$$

As for the connected component criterion — we chose the one that simply checks that the component has less than S nodes. S can be tuned on the training set.

Algorithm 1 Weighted Word Clustering

Input: graph G, cluster size threshold S, fraction of weak edges to remove F
Output: set of word clusters C

```
function CLUSTER(G, S, F)
    C ← ∅
    for each connected component c ⊆ G do
        if |c| ≤ S then
            C ← C ∪ {c}
        else
            remove F% of weak edges from c
            C ← C ∪ CLUSTER(c, S, F)
    return C
```

As we mentioned earlier, during the next step, the generated cluster IDs are used to substitute all the words in the input corpus.

Intuitively, this captures more information from the original corpus than the actual machine translation used in (Uszkoreit et al., 2010), because the result of the described transformation — a sequence of cluster IDs — represents many possible translations of every source document into target language and vice versa.

Besides, replacing machine translation with our method significantly improves overall performance of the system. First of all, the process is less demanding memory-wise as it doesn't require loading of phrase tables, language models, etc.; instead, only the cluster dictionary is used which is small (<100Mb of plain text in total for both languages even when using a phrase table built on a huge Web corpus). Second, it is also much faster as it basically consists of a single hashtable lookup per input word.

4 Data sets

The training data provided by WMT16 organizers consists of a set of 1,624 EN-FR URL pairs from 49 webdomains and all the pages crawled from the same domains. The crawled data for each page consists of the URL, language ID, mime type, encoding, HTML and text, of which our system only used URLs, language IDs and texts. The organizers also identified spans of French text and produced English translations using MT which we

also didn't use.

As will be explained further, we did not perform any specific parameter tuning and only used training data for quality analysis and to ensure that no mistakes were made.

For testing, 203 additional crawls of new web-domains were provided, distinct from the ones in the training data in the same format. The final evaluation was performed using a subset of 2402 URL pairs from the test data.

5 System details

Our system could use the provided training data in two ways. First, we could mine parallel data from it using some baseline algorithm to build the input phrase table used in the word clustering algorithm. Instead, we used an in-house phrasetable built from a large Web corpus. Second, it could be used to fine-tune parameters such as upper threshold on word cluster size, but our experiments on multiple data sets for different language pairs showed that, once these parameters are set to some adequate values, tuning them does not have a big impact on the result, effectively making the system language- and domain-independent.

The chosen parameter values are:

- maximum size of a word cluster = 90,
- order of matching n-grams = 5,
- order of scoring n-grams = 3,
- upper threshold on matching n-gram frequency = 2000.

6 Results and analysis

Simple evaluation on the training data achieves a recall of 81.47 (Here and below, test data results are almost identical to train; for exact values on test, please refer to the tables). However, analysis of the results on the training set uncovered a number of problems in the data that made this result an underestimation. Some of these problems are:

- incorrect language detection,
- empty pages or pages with crawling errors,
- duplicate and near-duplicate pages.

While the first two kinds of errors mostly don't affect our system's performance as long as there are no such errors in test set pairs, the third problem turns out to be quite serious.

	train		test	
position	count	recall	count	recall
1	1482	91.26	2233	92.96
2	96	97.17	110	97.54
3	18	98.28	8	97.88
4	3	98.46	4	98.04
5	1	98.52	5	98.25
6	0	98.52	1	98.29
7	2	98.65	0	98.29
8	1	98.71	0	98.29
9	0	98.71	1	98.33
10	0	98.71	0	98.33
none	21		40	
total	1624		2402	

Table 1: Reference document positions and n-best recall on the train and test data sets.

Some duplicate pages have exactly the same text content and only differ in some insignificant parameter in the URL, some are redirects, others only differ in a couple lines of boilerplate text (e.g., 'page viewed X times'), etc. Naturally, such sets of duplicates and near-duplicates negatively affect results of systems based on content analysis.

Also worth noting is the 1-1 rule enforced by the competition, which doesn't count pairs that include any of the URLs from the pairs accepted previously. This restriction significantly lowers the recall if the data contains near-duplicates of the pages from the reference pairs (which is almost always the case when working with crawled web-pages). Evaluating our system on training data without the 1-1 rule yields a recall of 91.26.

To provide further analysis, we set our system to output n-best lists of size 10 for every source document. Table 1 shows the distribution of the positions of the reference documents in the generated n-best lists. As you can see, considering 3 best condidates per source document yields a recall of 98.28 while 10-best recall is 98.71.

We further investigate 121 source documents whose references were scored 2nd to 10th. For these source documents we examine the intersection of the best scored candidate and the reference document (see Figure 1).

The results show a big amount of full duplicates (100% intersection) and near-duplicates (high val-

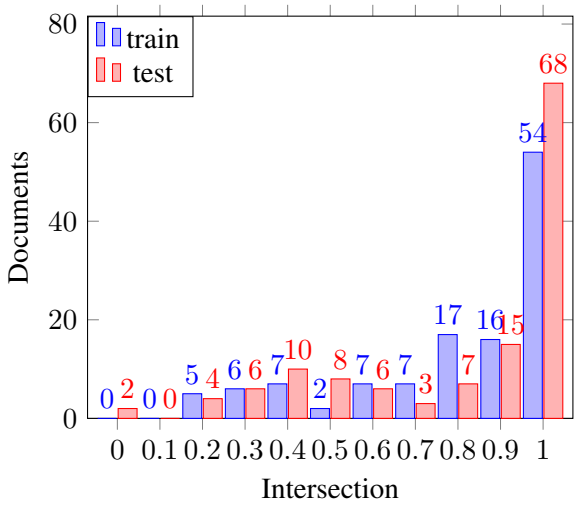

Figure 1: Intersection of the top-1 and the reference document in 10-best lists where reference document is on the 2nd - 10th place.

metric	train	test
1-1 rule recall	81.47	84.14
1-best recall	91.26	92.96
3-best recall	98.28	97.88
10-best recall	98.71	98.33
>80% similarity recall	96.61	96.71

Table 2: Quality on the training set using different metrics.

ues of intersection) in the generated n-best lists. This also brings us to a conclusion that most of the time the best scored candidate is not completely worthless but in fact can be used to mine parallel sentences from as it is very similar to the reference.

Considering top-1 scored documents that are not references but have 80% or more intersection with the reference 'correct' (which seems very reasonable), will achieve a recall of 96.61.

The most notable results for the training and test set are summarized in table 2.

7 Summary

We presented an effective, scalable and versatile approach to mining parallel data from big corpora of any nature. The method is based on textual content analysis and doesn't make any assumptions about the structure of the input data. Assuming the required input phrase table already exists, the system can work without any additional training data. Additionally, the parameters of the algorithm do not require any specific tuning, making it language- and domain-independent. We demonstrated that the system works well and achieves high values of recall on the provided data.

References

Ken'ichi Fukushima, Kenjiro Taura, and Takashi Chikayama. 2006. A fast and accurate method for detecting English-Japanese parallel texts. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability*, MLRI '06, pages 60–67, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Jakob Uszkoreit, Jay M. Ponte, Ashok C. Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1101–1109, Stroudsburg, PA, USA. Association for Computational Linguistics.