Quick and Reliable Document Alignment via TF/IDF-weighted Cosine Distance

Christian Buck
University of Edinburgh
Edinburgh, Scotland
`christian.buck@ed.ac.uk`

Philipp Koehn
Center for Language and Speech Processing
Department of Computer Science
Johns Hopkins University, Baltimore, MD
`phi@jhu.edu`

Abstract

This work describes our submission to the WMT16 Bilingual Document Alignment task. We show that a very simple distance metric, namely Cosine distance of tf/idf weighted document vectors provides a quick and reliable way to align documents. We compare many possible variants for constructing the document vectors. We also introduce a greedy algorithm that runs quicker and performs better in practice than the optimal solution to bipartite graph matching. Our approach shows competitive performance and can be improved even further through combination with URL based pair matching.

1 Related Work

The process of finding bilingual data online has been investigated since the early days of the world wide web (Resnik, 1999). In this work we are concerned with the problem of finding pairs of documents, a problem that can be structured in the following steps:

1. **Candidate Generation** The naive approach of considering all possible pairs of websites is often not applicable on a web scale, even when limiting the scope to a single webdomain. To overcome computational complexity previous work has focused on (i) matching pairs of URLs (Resnik and Smith, 2003) by removing language identifiers such as `&lang=en` or `/fr/` from URLs, (ii) considering only documents that either link to each other or that share a *parent page* (Resnik, 1999) that links to them, (iii) following links on already aligned documents (Shi et al., 2006), (iv) querying a search engine for possible translations (Ruopp and Xia, 2008), and (v) rephrasing the task as near-duplicate detection after translating all non-English content to English (Uszkoreit et al., 2010).

   (Ture et al., 2011) map all document vectors into a target language space and use an approximation of cosine distance based on locally-sensitive hashing (LSH) together with a sliding window algorithm to efficiently collect similar pairs.

2. **Document alignment** After possible pairings have been generated any distance function that compares two documents can be used to remove unlikely candidates. Common choices include (i) edit-distance between linearized documents (Resnik and Smith, 2003) (ii) cosine distance of idf-weighted bigram vectors (Uszkoreit et al., 2010), and (iii) probability of a probabilistic DOM-tree alignment model (Shi et al., 2006).

2 Approach

In this work we deal mainly with the second problem, document alignment, and just allow all possible source/target pairings. Thus, the task can be formalized as such: We are given a set of possible pairings

$$C = \{(d_s, d_t) \mid d_s \in D_s, d_t \in D_t\}$$

where C is the set of candidates, $d_s \in D_s$ are source language documents and $d_t \in D_t$ are target language documents. The task is to find a subset of $C' = \{(d_{s,i}, d_{t,i}), \ldots\} \subset C$ such that $d_{s,i}$ is a translation of $d_{t,i}$ (and vice versa) and the number $|C'|$ of such pairings is maximized.

We consider all source/target pairings that come from the same webdomain so that $C = D_s \times D_t$. This yields a fully connected bipartite graph with source and target pages being the partitions. By using a scoring function defined on edges the graph becomes weighted. We allow every page to occur not more than once in C', i.e. we do not allow 1:n or m:1 connections:

$$d_{s,i} = d_{s,j} \Leftrightarrow d_{t,i} = d_{t,j} \quad \forall\, (d_{s,i}, d_{t,i}), (d_{s,j}, d_{t,j}) \in C'$$

2.1 Selecting pairs

After computing a score for every edge of the bipartite graph, a matching of maximum weight can be found in $\mathcal{O}(\max(|D_s||D_t|)^3)$ by solving the assignment problem using the Kuhn-Munkres algorithm (Munkres, 1957). We expect every page of the non-dominant language to have a translated counterpart, thus $\min(|D_s|, |D_t|)$ pairs are generated.

In section 3.3, we compare the optimal assignment to a greedy solution by incrementally choosing the edge with the highest score and removing all other edges pointing to respective vertices. The greedy algorithm stops once no edges are left and produces the same number of pairs as the optimal solution but only requires $\mathcal{O}\left(|D_s||D_t| \times \log(|D_s||D_t|)\right)$ time to sort the score matrix.

3 Experiments

In total, the training dataset consists of 1624 document pairs from 49 web domains. The number of annotated aligned document pairs per web domain ranges from 4 to over 200.

Our experiments that led to the selection of the method used on the evaluation data are all based on a fixed and random split into train and development (dev) data: we split the data set into training (998 document pairs in 24 web domains) and test (626 document pairs in 25 web domains). The former is used for extensive experimentation, the latter to select the best approach for our shared task submission.

3.1 Performance considerations

Our approach requires us to produce a dense matrix of feature values which seems prohibitively expensive given the high number of possible pairings. In practice, even for the largest webdomains in our data, requiring the scoring of roughly 1B possible pairs, we are able to produce all values quickly enough that the run-time is dominated by I/O and preprocessing steps such as tokenization.

ngram size	$n=1$	$n=3$	$n=5$
Number of unique n-grams			
Used for scoring	53k	1.2M	1.7M
Ignored because freq < 2	11k	351k	658k
Non-zero entries in (sparse) document matrix			
Source (English)	1.5M	5.7M	6.1M
Target (French)	0.4M	1.4M	1.4M
Time per processing step (single-threaded)			
Read tokenized corpus	117s	117s	117s
IDF estimation	15s	26s	30s
Document vectors	33s	86s	91s
Pairwise distances	8s	11s	20s

Table 1: Runtime details for generation of 971M pairwise cosine similarity features for n-grams of size $\{1, 3, 5\}$ on `virtualhospice.ca`. N-grams which occur fewer than 2 times are filtered from the corpus. Single-threaded execution on 2.66Ghz Xeon CPU.

As can be seen from Table 1, a total of 1.2M 3-grams types are used for scoring pages from `virtualhospice.ca` which holds 43.5k English and 22.3k French pages. Loading the corpus, estimating the idf weights, and populating the sparse document matrices with roughly 7M entries both take about 2 minutes. On the other hand, producing the $43.5k \times 22.3k = 971M$ pairwise distances only accounts for 11 seconds.

Speed and, more importantly, memory consumption can be further improved by pruning all n-grams that occur fewer times than a set threshold in the corpus. We find empirically that maintaining a very low minimum count cutoff somewhere below 10 is crucial for maintaining high recall, as shown in Figure 1.

3.2 TF-IDF weighting

In the literature (Manning et al., 2008) a number of different weighting schemes based on tf/idf have been proposed with the overall goal to assign lower scores to terms (or n-grams) that are less discriminatory for document comparison.

However, these approaches usually aim at document retrieval, i.e. finding relevant documents given a large (in comparison to the overall document size) number of search terms. In the setting of near duplicate detection, our *query* is a complete document and other weighting schemes may apply.

To empirically evaluate the fitness of different approaches we implement the following weighting schemes for term frequency (tf).

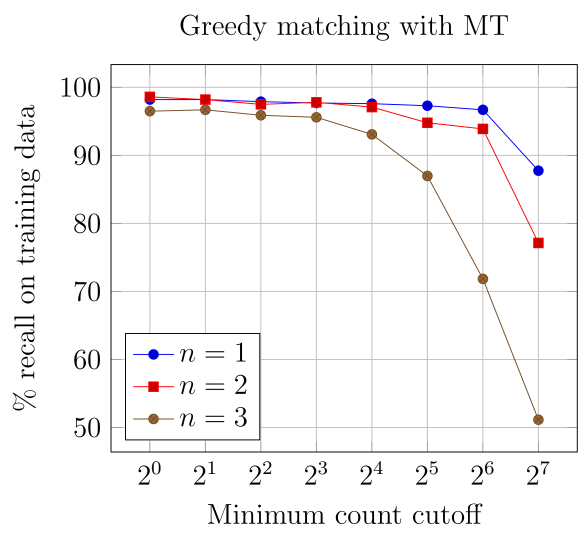

Figure 1: Recall on training set using varying minimum counts of n-grams in the corpus. N-grams seen fewer times than the threshold are ignored when building the document vectors.

In every case we define

$$\mathrm{tf}(w_1^n, d) = 0 \text{ if } w_1^n \notin d$$

and give the other case below:

$$\mathrm{tf}_1(w_1^n, d) = 1 \tag{1}$$

$$\mathrm{tf}_2(w_1^n, d) = \mathrm{freq}(w_1^n, d) \tag{2}$$

$$\mathrm{tf}_3(w_1^n, d) = 1 + log\left(\mathrm{freq}(w_1^n, d)\right) \tag{3}$$

$$\mathrm{tf}_4(w_1^n, d) = .4 + .6\frac{\mathrm{freq}(w_1^n, d)}{max_{w_1'^n} \mathrm{freq}(w_1'^n, d)} \tag{4}$$

$$\mathrm{tf}_5(w_1^n, d) = \frac{\mathrm{freq}(w_1^n, d)}{max_{(\bar{w}_1^n, \bar{d})} \mathrm{freq}(\bar{w}_1^n, \bar{d})} \tag{5}$$

$$\mathrm{tf}_6(w_1^n, d) = \sqrt{\mathrm{freq}(w_1^n, d)} \tag{6}$$

In the same way we implement weighting schemes for inverse document frequency $\mathrm{idf}(w_1^n, D_s, D_t) = idf(\cdot)$:

$$\mathrm{idf}_1(\cdot) = 1 \tag{7}$$

$$\mathrm{idf}_2(\cdot) = \frac{|D_s \cup D_t|}{1 + \mathrm{df}(w_1^n, D)} \tag{8}$$

$$\mathrm{idf}_3(\cdot) = log\left(1 + \frac{max_{\bar{w}_1^n} \mathrm{df}(\bar{w}_1^n, D)}{\mathrm{df}(w_1^n, D)}\right) \tag{9}$$

$$\mathrm{idf}_4(\cdot) = \log\left(1 + \frac{|D_s \cup D_t|}{\mathrm{df}(w_1^n, D)}\right) \tag{10}$$

$$\mathrm{idf}_5(\cdot) = \max\left(0, \log\frac{|D_s \cup D_t| - \mathrm{df}(w_1^n, D)}{\mathrm{df}(w_1^n, D)}\right) \tag{11}$$

$$\mathrm{idf}_6(\cdot) = 1 + \log\frac{|D_s \cup D_t|}{1 + \mathrm{df}(w_1^n, D)} \tag{12}$$

where $D = D_s \cup D_t$ and

$$\mathrm{df}(w_1^n, D) = |\{d \in D \mid w_1^n \in d\}|$$

Slight variations of the above definitions can be found in the wild, for example the search engine Apache Lucene[1] uses tf_6 and idf_6 but uses $1 + |D_s \cup D_t|$ in the numerator since version 6.

We evaluate the cross product of weighting schemes using the train and dev splits as described above. Looking at the results in Tables 2 and 3, a number of interesting observations can be made:

1. Performance differs between train and dev data, with results on the training portion of the data being several percents better. This indicates a skew in the data distribution which is surprising given that the web-domains were selected beforehand. We know from the training data that about $\frac{1}{4}$ of the known pairs, 236 of 998, are found in a single webdomain *tsb.gc.ca* which could explain the skew. However, the difference remains if that large webdomain is removed.

   Further investigation reveals that the underlying cause of poor performance on the dev set can be attributed to three webdomains that contain near duplicates, such as the same main content but interface elements in a different language.

2. When choosing the optimal length of scoring n-grams, shorter is better. Good recall can be achieved using 1-grams for the monolingual

[1] `https://lucene.apache.org/core/6_0_1/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html`

case where no machine translated (MT) data is used and 1-grams or 2-grams for the case where all French data is translated to English beforehand.

3. In tf-idf weighting the inverse document frequency acts as an indicator of a term's importance. This is important in the case of information retrieval where query words differ in utility. In the duplicate detection setting idf weights play a less important role and a common choice such as idf_3 defined in Equation 9 can be used throughout.

4. Results produced by using only the untranslated text, a configuration that requires no bilingual resources and little computational resources, are better than we expected: only between 5% (for train) and 8% (for dev) below the recall achieved using machine translated texts. In this case we just ignore that two pages are written in different languages and only rely on untranslated parts such as boilerplate, names, and numbers to provide sufficient cues.

For our submission we used the machine translated text provided by the organizers and chose $n = 2$, tf_4 (Equation 4), and idf_3 (Equation 9).

3.3 Greedy vs. optimal solution

We found that producing the optimal solution for the assignment problem using the Kuhn-Munkres algorithm (Munkres, 1957) was slightly worse in almost all cases. We hypothesize that by maximizing the aggregate score for all selected pairs the low-scoring pairs for which no matching document exists are over-emphasized. To test this hypothesis we compare the scores of the selected pairs for both algorithms: For each webdomain we sort the selected pairs by their score and select, for each algorithm, the n top scoring pairs:

Let $s(d_s, d_t)$ be our scoring function, in this case we use Cosine similarity, and let

$$(d_{s,g_1}, d_{t,g_1}), \dots, (d_{s,g_N}, d_{t,g_N})$$

be the document pairs selected by the greedy algorithm and, likewise,

$$(d_{s,o_1}, d_{t,o_1}), \dots, (d_{s,o_N}, d_{t,o_N})$$

those selected by the optimal algorithm. Let these pairs be sorted by score such that

$$s(d_{s,g_i}, d_{t,g_i}) \geq s(d_{s,g_{i+1}}, d_{t,g_{i+1}}) \; \forall i$$

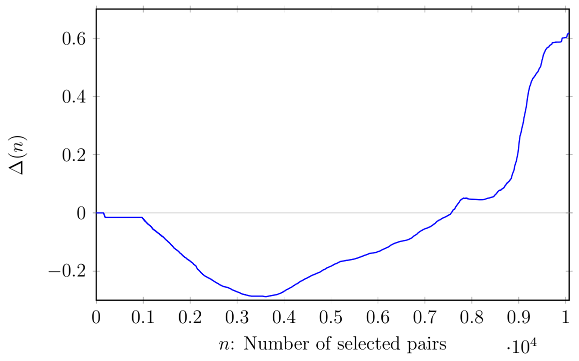

Figure 2: Difference in accumulated cosine distances between greedy and optimal algorithm. For more than the first half of the selected pairs, the greedy algorithm overall outperforms the optimal one, indicated by a negative $\Delta(n)$.

and

$$s(d_{s,o_i}, d_{t,o_i}) \geq s(d_{s,o_{i+1}}, d_{t,o_{i+1}}) \; \forall i$$

Let $\Delta(n)$ be the accumulated difference of scores for the first n pairs:

$$\Delta(n) = \sum_{i=1}^{n} s(d_{s,o_i}, d_{t,o_i}) - s(d_{s,g_i}, d_{t,g_i}) \quad (13)$$

Since the greedy algorithm is not necessarily optimal we know that $\Delta(N) \geq 0$. However, as can be seen from Figure 2, the greedy selection of the best scoring pairs outperforms the Kuhn-Munkres algorithm for the top-scoring half, confirming our assumption that lower scoring pairs are selected in order to find better scoring matches for the documents without a counterpart.

We note that even after selecting 10 000 pairs, the accumulated difference is comparatively small, hinting that very similar sets have been selected. Figure 3 shows the Jaccard Similarity between the top n pairs

$$P_g(n) = \{(d_{s,g_i}, d_{t,g_i}) | 1 \leq i \leq n\} \quad (14)$$
$$P_o(n) = \{(d_{s,o_i}, d_{t,o_i}) | 1 \leq i \leq n\} \quad (15)$$

for both algorithms. The Figure confirms that either approach selects virtually the same set of pairs for low numbers of n.

Thus, the globally optimal solution is not only expensive to compute but also very similar to the greedy selection and it outperforms the greedy algorithm mostly for pairs in the tail that are likely misaligned anyways, because no translated page exists. Hence, all our reported results use the greedy selection introduced in Section 2.1.

n		idf_1	idf_2	idf_3	idf_4	idf_5	idf_6	idf_1	idf_2	idf_3	idf_4	idf_5	idf_6
1	tf_1	90.4	93.5	93.9	93.9	93.9	93.4	86.1	86.1	86.3	86.1	86.3	85.9
	tf_2	65.5	83.7	80.6	81.7	81.2	83.5	69.5	80.8	80.7	80.8	81.0	80.7
	tf_3	86.8	91.3	92.0	92.1	91.8	91.4	83.4	85.0	84.8	84.8	84.3	85.1
	tf_4	88.7	**92.9**	**93.5**	**93.4**	**93.6**	**92.9**	86.6	**87.2**	**87.2**	**87.1**	**86.7**	**87.1**
	tf_5	65.5	83.7	80.6	81.7	81.2	83.5	69.3	80.8	80.7	80.8	81.0	80.7
	tf_6	65.6	83.7	80.5	81.6	81.2	83.5	68.8	80.2	80.4	81.0	80.8	80.2
2	tf_1	76.6	84.9	84.0	84.7	85.5	84.9	78.1	81.5	81.5	81.3	81.5	81.0
	tf_2	70.3	81.1	80.5	80.7	81.3	80.4	70.8	76.0	75.6	75.7	76.5	75.2
	tf_3	76.0	83.2	82.6	83.2	83.6	83.0	75.1	79.9	79.4	79.7	80.2	79.6
	tf_4	76.1	84.3	83.7	84.0	85.1	84.2	77.8	80.5	80.5	80.7	80.7	80.0
	tf_5	70.3	81.1	80.5	80.7	81.3	80.4	70.8	76.0	75.6	75.7	76.5	75.2
	tf_6	70.3	81.1	80.5	80.7	81.3	80.4	70.8	76.0	75.6	75.7	76.5	75.2
3	tf_1	65.7	72.2	71.7	72.3	72.6	72.0	64.9	65.3	65.3	65.8	65.2	65.2
	tf_2	62.7	70.5	70.0	70.3	70.6	70.1	61.2	63.6	63.3	63.7	63.4	63.6
	tf_3	65.3	71.8	71.4	71.6	72.0	71.2	62.9	64.5	64.9	64.9	64.2	65.0
	tf_4	65.1	71.9	71.7	71.9	72.5	71.7	63.4	65.8	66.0	66.0	65.7	66.0
	tf_5	62.7	70.5	70.0	70.3	70.6	70.1	61.2	63.6	63.3	63.7	63.4	63.6
	tf_6	62.7	70.5	70.0	70.3	70.6	70.1	61.3	63.6	63.3	63.7	63.4	63.6
4	tf_1	59.0	63.2	63.0	63.4	63.7	63.4	56.5	57.5	57.5	57.7	57.2	57.3
	tf_2	58.1	63.6	62.9	63.6	63.7	63.3	55.1	56.5	56.2	56.7	56.5	56.4
	tf_3	58.5	63.1	62.9	63.6	63.7	63.3	56.2	57.7	58.0	58.3	57.7	57.7
	tf_4	58.6	63.3	63.0	63.3	63.6	63.1	56.7	58.1	58.1	58.3	57.7	57.7
	tf_5	58.1	63.6	62.9	63.6	63.7	63.3	55.1	56.5	56.2	56.7	56.5	56.4
	tf_6	58.1	63.6	62.9	63.6	63.7	63.3	55.1	56.7	56.2	56.7	56.7	56.4

Table 2: Recall on train (left) and dev (right) split of the training data using different tf/idf weighting schemes and only *untranslated* text.

n		idf_1	idf_2	idf_3	idf_4	idf_5	idf_6	idf_1	idf_2	idf_3	idf_4	idf_5	idf_6
1	tf_1	97.7	97.9	97.9	97.9	97.9	97.9	89.8	89.8	89.8	86.7	89.8	89.8
	tf_2	93.1	94.8	94.8	94.8	91.0	94.8	88.2	89.9	89.9	89.9	87.5	89.9
	tf_3	97.9	97.7	97.7	97.8	97.7	97.8	89.8	90.4	90.4	90.4	90.6	90.4
	tf_4	97.9	**98.2**	**98.2**	**98.2**	**98.2**	98.2	89.6	89.1	89.1	89.1	89.1	89.1
	tf_5	93.2	94.9	94.9	94.9	91.1	94.9	88.2	89.9	89.8	89.9	87.5	89.9
	tf_6	93.4	94.9	95.1	94.9	91.1	95.1	88.5	89.9	89.5	89.5	86.9	90.3
2	tf_1	97.9	98.2	98.2	98.2	98.2	98.2	94.2	94.7	94.7	94.9	94.7	94.6
	tf_2	95.5	96.8	96.8	96.8	96.8	96.8	92.8	93.9	93.9	94.1	93.9	93.6
	tf_3	97.7	98.1	**98.2**	**98.2**	98.1	**98.3**	94.1	**95.4**	**95.4**	**95.4**	**95.4**	95.2
	tf_4	97.8	**98.2**	**98.2**	**98.2**	**98.2**	**98.3**	94.1	95.0	95.0	95.0	95.0	94.9
	tf_5	95.5	96.8	96.8	96.8	96.8	96.8	92.8	93.9	93.9	94.1	93.9	93.6
	tf_6	95.4	96.8	96.8	96.8	96.8	96.8	93.0	94.1	94.1	94.2	93.9	93.6
3	tf_1	96.6	96.9	96.9	96.8	96.9	96.9	94.1	93.5	93.5	93.5	93.5	93.5
	tf_2	95.3	96.1	96.1	96.1	96.1	96.1	92.3	93.5	93.5	93.3	93.5	93.5
	tf_3	96.2	96.5	96.5	96.5	96.5	96.5	93.6	93.6	93.6	93.6	93.6	93.6
	tf_4	96.5	96.7	96.7	96.7	96.7	96.7	93.8	93.5	93.5	93.5	93.5	93.5
	tf_5	95.4	96.1	96.1	96.1	96.1	96.1	92.3	93.5	93.5	93.3	93.5	93.5
	tf_6	95.2	96.1	96.1	96.1	96.1	96.1	92.7	93.5	93.5	93.5	93.5	93.6
4	tf_1	95.0	96.1	96.0	96.0	96.0	96.1	93.3	93.6	93.6	93.6	93.6	93.6
	tf_2	94.5	96.3	96.3	96.3	96.3	96.3	91.7	92.5	92.5	92.5	92.5	92.5
	tf_3	95.1	96.2	96.1	96.1	96.1	96.1	93.0	93.1	93.1	92.7	93.0	93.3
	tf_4	95.0	96.0	95.9	95.9	95.9	96.0	93.5	93.5	93.5	93.5	93.5	93.6
	tf_5	94.5	96.3	96.3	96.3	96.3	96.3	91.7	92.5	92.5	92.5	92.5	92.5
	tf_6	94.5	96.3	96.3	96.3	96.3	96.3	91.5	92.8	92.5	92.7	92.8	92.5

Table 3: Recall on train (left) and dev (right) split of the training data using different tf/idf weighting schemes to compare English and machine translated French text.

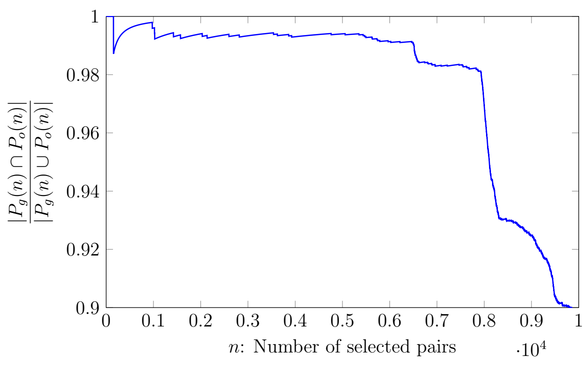

Figure 3: Jaccard Similarity between the top-n pairs selected by greedy and Kuhn-Munkres algorithm.

4 Results

The test data for the shared task consists of 203 crawled websites that are all distinct from the training set. No additional known pairs are provided for these webdomains, but the organizers offer translations of French text into English, as for the training data. As above, performance in evaluated via recall under the condition that every document can only be part of a single pair. The number of pages per domain varies wildly between 9 and almost 100k. In the latter case, 50k pairs need to be picked from roughly 2.5B possibilities. After some preprocessing such as tokenization we produce 368 260 pairs using greedy selection and cosine distance as explained above. For all webdomains this takes less than 4h on a single machine.

In total, 13 research teams contributed 21 submissions to the shared task. The official results can be found in Table 4. Our submission ranks on 3^{rd} place. We would like to point out that, apart from selecting the best performing tf/idf weighting method, the training data is not used at all. Thus, besides a baseline machine translation system no additional resources are needed, which makes our approach widely applicable.

A baseline system based on matching URL patterns such as `site.com/home-fr/` and `site.com/home/en/` as used in previous work (Resnik and Smith, 2003; Smith et al., 2013) is provided by the organizers. We combine our approach and the Baseline by simply selecting all 148 537 baseline pairs first. While not on official submission, Table 4 shows that this combination outperforms all other systems.

Name	Recall %	Found
NovaLincs-url-coverage	94.96	2 281
YODA	93.92	2 256
UEdin1_cosine	**89.09**	**2 140**
NovaLincs-coverage	88.63	2 129
DOCAL	88.59	2 128
UEdin2_LSI-v2	87.64	2 105
UEdin2_LSI	85.85	2 062
NovaLincs-coverage-url	85.76	2 060
ILSP-ARC-pv42	84.93	2 040
UFAL-4	84.22	2 023
YSDA	84.14	2 021
UA_PROMPSIT_bitextor_5.0	83.31	2 001
UFAL-1	81.31	1 953
UFAL-3	80.68	1 938
Meved	79.39	1 907
Jakubina-Langlais	79.31	1 905
UFAL-2	79.14	1 901
UA_PROMPSIT_bitextor_4.1	31.14	748
ADAPT	27.10	651
ADAPT-v2	26.81	644
JIS	2.00	48
Baseline	59.78	1 436
Baseline + UEdin1_cosine†	**96.21**	**2 311**

Table 4: Official results on the shared task test data. Results described in this work are fat. Across all webdomains a total of 2402 known pairs were to be found. (†) indicates a non-official result that was produced post-submission.

5 Conclusion

We present a comparison of tf/idf weighting schemes for comparison of original and translated documents via cosine distance. We find that the right choice of term-frequence (tf) weighting is crucial in this setting, along with the inclusion of low-frequency words.

We compare a greedy selection algorithm to a computationally more expensive solution which yields a slightly better global solution. We can show that the former often outperforms the latter in practical settings where a tail of un-pairable document exits.

Our best results are based on machine translated documents. However, even when ignoring the fact that two documents are written, at least partially, in different languages, we are still able to discover a substantial number of parallel pages.

Results of the shared task show that our approach, which only uses the website's text, yields competitive results. Results improve further when our predictions are combined with pairs found via URL matching.

References

[Manning et al.2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York.

[Munkres1957] James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

[Resnik and Smith2003] Philip Resnik and Noah A Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

[Resnik1999] Philip Resnik. 1999. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 527–534, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ruopp and Xia2008] Achim Ruopp and Fei Xia. 2008. Finding parallel texts on the web using cross-language information retrieval. In *IJCNLP*, pages 18–25.

[Shi et al.2006] Lei Shi, Cheng Niu, Ming Zhou, and Jianfeng Gao. 2006. A dom tree alignment model for mining parallel data from the web. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 489–496. Association for Computational Linguistics.

[Smith et al.2013] Jason Smith, Hervé Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of ACL*. Association for Computational Linguistics, August.

[Ture et al.2011] Ferhan Ture, Tamer Elsayed, and Jimmy Lin. 2011. No free lunch: Brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 943–952. ACM.

[Uszkoreit et al.2010] 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*, pages 1101–1109. Association for Computational Linguistics.