$B_1A_3D_2$ LUC@WMT 2016: a Bilingual$_1$ Document$_2$ Alignment$_3$ Platform Based on Lucene

Laurent Jakubina
RALI - DIRO
Université de Montréal
jakubinl@iro.umontreal.ca

Philippe Langlais
RALI - DIRO
Université de Montréal
felipe@iro.umontreal.ca

Abstract

We participated in the Bilingual Document Alignment shared task of WMT 2016 with the intent of testing plain cross-lingual information retrieval platform built on top of the Apache `Lucene` framework. We devised a number of interesting variants, including one that only considers the URLs of the pages, and that offers — without any heuristic — surprisingly high performances. We finally submitted the output of a system that combines two informations (`text` and `url`) from documents and a post-treatment for an accuracy that reaches 92% on the development dataset distributed for the shared task.

1 Introduction

While many recent efforts within the machine translation community are geared toward exploiting bilingual comparable corpora — see (Munteanu and Marcu, 2005) for a pioneering work and (Sharoff et al., 2013) for an extensive review — there is comparatively much less work devoted to identifying parallel documents in a (potentially huge) collection. See (Uszkoreit et al., 2010; Smith et al., 2013) for two notable exceptions. This is due in large part to conventional wisdom that holds that comparable corpora can be found more easily and in larger quantity than parallel data. Still, we believe that parallel data should not be neglected and should even be preferred when available.

The Bilingual Document Alignment shared task of WMT 2016 is designed for precisely identifying parallel data in a (huge) collection of bilingual documents mined over the Web. The collection has been processed by the organizers in such a way that this is easy to test systems: the language of the documents is already detected, and we have access to the content of the Web pages. Although the organizers encouraged participants to test their own way of pre-processing data, we decided (for the sake of simplicity) to use the data as prepared.

We describe the overall architecture of the BADLUC framework as well as its components in Section 2. We explain in Section 3 the experiments we conducted and provide some analysis in Section 4. We conclude in Section 5.

2 BADLUC

We built variants of a Cross-Language Information Retrieval (CLIR) platform making use of the popular Apache framework `Lucene`.[1] We describe here the different components embedded in this platform.

We participated in this task by relying entirely on the pre-processing carried out by the organizers, that is, we used the text of the pages as it was extracted. Figure 1 shows an excerpt of the text extracted from a given URL. Sometimes, the conversion to text is noisy and deserves further work. While we could have used the machine translation provided as well, we decided to resort to a bilingual dictionary, mainly for the sake of simplicity: the resulting system is very light and can be deployed without retraining any component.

After some exploration with the platform, we settled for a configuration — named RALI — that we used for treating the official dataset of the shared task. RALI is a combination of variants that delivers good performance both in terms of processing time and accuracy. This system achieves 92.1% TOP@1 on the development dataset, a performance we consider satisfactory considering the simplicity of the approach.

[1] `https://lucene.apache.org/core/`

2.1 Indexes

We built two main indexes. One from the source and one from the target documents of the collection provided. This last was organized into web-domains (49 in the development set) but, to ease implementation, we built the indexes from all, and enforced a posteriori that only target documents of a given web-domain are returned. In each index, documents are indexed (and tokenized) into three `Lucene` fields, one based on the text itself (`text`), one based on its `url` and one with the size of the text content (in number of tokens).

`Lucene` provides a number of tokenizers, but we felt the need to develop our own in order to properly handle the cases where punctuations is glued to words, and other typical cases one finds in real data. One point worth mentioning is that our tokenizer splits `urls` into several tokens.[2] This way of handling `urls` leads us to a simple but efficient `url`-based baseline. See Figure 1 for an illustration of a few bag-of-word queries considered in BADLUC.

2.2 Query Instantiation

Each field of each (source) document can be treated as a bag-of-word query. We used the *MoreLikeThis* query generator available in `Lucene`[3] to implement this. The generator uses a variant of *tf.idf* and allows for the adjustment of a number of meta-parameters mainly for finding an application-specific compromise between the retrieval speed and its accuracy. We investigated the following ones[4]:

- minimum frequency of a term in a document (tf) for it to be considered in a query,
- minimum ($mindf$) and maximum ($maxdf$) number of documents in the collection that should contain a candidate query term,
- minimum ($minwl$) and maximum ($maxwl$) word length of a term in a query,
- maximum number of terms in a query ($size$),
- only words absent from a specified stop-list are legitimate query terms ($stop$).

These meta-parameters allow to easily create specific-purpose queries on the fly. For instance, by setting $mindf$ and $maxdf$ to 1, we built a collection-wide hapax query, while setting $minwl$ and $maxwl$ to 1 allows to build queries containing only punctuations marks.

2.2.1 Mono and Bilingual Queries

We tested two main families of queries: monolingual (`mono`) and bilingual (`bili`). The former is a way of easily capturing the tendency of a document and its translation to share a number of specific entities such as named-entities, numbers, or `urls`, for which no translation is required. Obviously, we were not expecting a high accuracy with monolingual queries, but we thought it would provide us with a very simple baseline. Actually the performance of such an engine on a given collection might be a valid metric to report, as a measure of the *difficulty* of the collection.

Bilingual queries involve a translation procedure. We simply translate the terms of the query based on a bilingual lexicon. We could have used the machine translated text provided by the organizers, but we decided early on in our experiments to resort to a simple bilingual lexicon approach, to simplify deployment. As a matter of fact, in a previous work on identifying parallel material in Wikipedia (Rebout and Langlais, 2014), we observed the inadequacy of the features computed from a generic SMT engine. Arguably our lexicon might not be very good either to deal with the nature of data collected over the Web, but we felt that a *general* bilingual lexicon might be more robust in this situation.

There are two meta-parameters that control our translation procedure:

- $keep$ when set to true (which we note K), will leave untranslated terms (that is, terms unknown from our lexicon) in the query.[5] Hopefully this will leave in named- and numerical-entities that are useful for distinguishing parallel documents (Patry and Langlais, 2011).
- $nbTrans$ controls the number of translations to keep when there is more than one available for a given source term. We consider two possible values: `all` puts all available

[2] We split `urls` according to a list of 33 separators, among which: @,?,/,<,>,(,),+,!,%,∼

[3] `https://lucene.apache.org/core/4_4_0/queries/index.html`

[4] The *MoreLikeThis* mechanism also allows to settle a boost factor per query term, but we did not play with it.

[5] At least terms that meet the *MoreLikeThis* meta-parameters.

http://creationwiki.org/Earth Earth - CreationWiki, the encyclopedia of creation science [...] 23.439281 0.409rad 26.044grad Physical characteristics Mass 5.9736 * 1024 kg [...] taking 23 hours, 56 minutes, and 4.091 seconds to line up relative to the stars (Sidereal day), and 24 hours plus or minus 20 seconds to line up relative to the sun [...] is closer to the sun at some times of the year than others; the Earth moves faster [...] Kepler's laws of planetary motion [...] Saturnine - Uranian - Neptunian [...]		
text	mono	*texttok:* hours neptunian 1024 tennessee 2008 closer 397 ...
	bili	*texttok:* hours neptunian 1024 penchant théorie visible intensité fois tennessee prononcée prénommé 2008 métrique note closer équateur ...
url	mono	*urltok:* earth creationwiki / org http . : ...
	bili	*urltok:* déblai masse tanière terre earth creationwiki / org http . : ...
both	bili	*texttok:* hours neptunian 1024 penchant théorie tennessee absorbant prononcée prénommé 2008 métrique 397 équateur ... *urltok:* déblai masse tanière terre earth creationwiki / org http . : ...

Figure 1: Excerpt of bag-of-word queries for `http://creationwiki.org/Earth`.

translations in the query, while `first` picks the first one listed in the bilingual lexicon.[6]

2.2.2 Queries on Both Fields

`Lucene` allows to combine queries made on different fields. We use this functionality in order to produce queries with terms to be searched in both fields (`text` and `url`) in a single pass. An explicit example of this query is illustrated at the bottom of Figure 1.

2.2.3 Length-based Filter

`Lucene` allows to write queries as filters. It is basically a query that is executed before the main one and that returns an initial list of target documents on which the main query is applied. We implemented one such filter (`size`), using the third indexed field, based on the observation that pairs of parallel documents should have similar lengths (counted in tokens). We assumed the size ratio of source/target documents follows a normal distribution whose variance defines a confidence interval in which the target document size should fall. Unfortunately we estimated the parameters of the normal distribution on all reference pairs of documents provided by the shared task. This could explain our unsatisfactory scores on the official test set of the shared task[7].

[6]There is no specific order in the multiple translations listed in our lexicon for a given term, but some lexicons might list more general translations first.

[7]We noticed this bias after the submission period.

2.3 Post Processors

Query execution produces for each source document a ranked list of target documents. Since each query is carried out independently over the collection, we run the risk of having a given target document associated with more than one source document. As a solution, we tested a few post-processors that select exactly one candidate per source document:

`hungarian` the Hungarian Algorithm (Kuhn, 1955) is a well-known combinatorial optimization algorithm[8] that solves the assignment problem in polynomial time.

`b-greedy` a *batch* greedy solution which picks the best ranked candidate among all the source documents paired, removes the selected pairs and loops until all source documents get paired with exactly one document.

`o-greedy` an *online* version of the greedy procedure just described, where we select for each source document the top ranked candidate that has not been paired with a previous source document yet. Once selected, the target document is removed from the potential list of candidates for subsequent source documents.

On a task of identifying parallel documents in Wikipedia, (Rebout and Langlais, 2014) shows that both the `hungarian` and the `b-greedy` algorithms deliver good performance overall.

[8]Implementation available here: `https://github.com/KevinStern/software-and-algorithms`

	Strategies		TOP (%)		
	Query	[MLT] + [Trans]	@1	@5	@100
		`text` variants			
default	`mono`	$[2, 5, \infty, 0, 25, \texttt{F}]$	6.4	15.8	49.5
default+tok		$[2, 5, \infty, 0, 25, \texttt{F}]$	35.4	57.0	83.9
		$[1, 1, \infty, 1, 200, \texttt{F}]$	48.3	78.2	94.7
		$[1, 1, \infty, 1, \infty, \texttt{T}]$	57.2	86.2	96.2
		$[1, 1, \infty, 1, 200, \texttt{T}]$	64.9	87.2	96.8
	`+size`	$[1, 1, \infty, 1, 200, \texttt{T}]$	76.2	92.1	97.3
	`+size`	$[1, 1, \infty, 1, \infty, \texttt{T}]$	76.6	92.6	97.2
stop words	`+size`	$[1, 1, \infty, 1, \infty, \texttt{F}]$	69.2	89.7	96.4
wl = 3	`+size`	$[1, 1, \infty, 3, \infty, \texttt{T}]$	75.1	92.0	97.1
hapax	`+size`	$[1, 1, 1, 1, \infty, \texttt{T}]$	49.5	49.8	49.8
	`bili`	$[1, 1, \infty, 1, \infty, \texttt{T}]$ + [K,`first`]	74.4	93.5	98.7
		$[1, 1, \infty, 1, \infty, \texttt{F}]$ + [K,`first`]	71.9	92.8	**98.8**
		$[1, 1, \infty, 1, \infty, \texttt{F}]$ + [K,`all`]	34.5	53.2	88.4
		$[1, 1, \infty, 1, \infty, \texttt{T}]$ + [K,`all`]	44.1	64.5	95.0
	`+size`	$[1, 1, \infty, 1, \infty, \texttt{F}]$ + [K,`all`]	81.2	**97.1**	98.3
	`+size`	$[1, 1, \infty, 1, \infty, \texttt{T}]$ + [$\neg$K,`all`]	81.0	94.8	98.2
best-`text`	`+size`	$[1, 1, \infty, 1, \infty, \texttt{T}]$ + [K,`first`]	**83.3**	96.2	98.2
		`url` variants			
WMT 2016			67.9		
	`mono`	$[1, 1, \infty, 1, \infty, \texttt{F}]$	75.4	84.4	92.9
	`+size`	$[1, 1, \infty, 1, \infty, \texttt{F}]$	78.4	87.5	95.3
	`bili`	$[1, 1, \infty, 1, \infty, \texttt{F}]$ + [K,`all`]	77.0	86.6	93.5
	`+size`	$[1, 1, \infty, 1, \infty, \texttt{F}]$ + [K,`first`]	78.8	88.0	91.3
best-`url`	`+size`	$[1, 1, \infty, 1, \infty, \texttt{F}]$ + [K,`all`]	**80.1**	**88.6**	**95.6**
RALI	`bili-size`	best-`text`+best-`url`	**88.6**	**97.6**	98.3

Table 1: Performances of some selected variants we tested. The MLT meta-parameters are [tf, $mindf$, $maxdf$, $minwl$, $maxwl$, $size$, $stop$], while those specific to the translation process are [$keep$, $nbTrans$]. See Section 2 for more.

3 Experiments

3.1 Protocol

We conducted these experiments on the `lett.train` webcrawl available on the WMT2016 shared task webpage.[9] This crawl consists of 49 webdomains of various sizes, and the language of each document has been identified.

The test set made available for participants to calibrate their systems contains 1624 English `url`s for which the target (French) parallel counterpart is known. It is noteworthy that the task does not evaluate the ability of a system to detect whether a given source `url` has its parallel counterpart in the collection, which would require to train a classifier. [10] Because of this, we always propose a target `url` for a source one; and we measure performance with accuracy at rank 1, 5 and 100. Accuracy at rank i (TOP@i) is computed as the percentage of source `url`s for which the reference `url` is identified in the top-i candidates.

On top of our tokenizer which is clearly bi-

[9] `http://www.statmt.org/wmt16/bilingual-task.html`

[10] We have conducted the training of such a classifier in past experiments (Rebout and Langlais, 2014), with results we evaluated to be around 85%.

ased toward space-oriented language scripts, we use two language specific resources: a stop-word list for English which comprises 572 entries,[11] as well as an in-house English-French bilingual lexicon of 107 799 entries. Very roughly, our lexicon could help the translation of only half of the query terms, which is an issue we should look at in the future.

3.2 Results

We tested over a thousand configurations, varying the meta-parameters of the *MoreLikeThis* (MLT) query generator, as well as the components described in the previous section. Table 1 shows a selection of some of the variants we tested. A line in this table indicates the best MLT meta-parameters we found for the configuration specified.

First of all, and without much surprise, we are able to outperform the `url` baseline (line WMT 2016) proposed by the organizers and which relies on some rules for matching `url`s in both language. Our best variant (line best-`url`) relying only on `url`s significantly outperforms this baseline by 12 absolute points in TOP@1. This variant tokenizes the `url`, then translates its words.[12]

Focusing on variants that exploit the text of the documents, we achieve a decent result without involving translation at all: the best monolingual variant we tested performs at 76.6 TOP@1, which also outperforms the WMT baseline. It should be noted that the default `Lucene` configuration (line `default`) does not perform well at all. Clearly, some tuning is necessary. In particular, using our tokenizer instead of the default one (which separates words at spaces) drastically increases performance (line `default+tok`). See Figure 1 for the kind of noisy input a tokenizer needs to handle. Unquestionably, using translation increases performance. The best variant we tested (line best-`text`) picks only one translation per source word, and leaves in the query the terms without translation.

Another interesting fact is the positive impact of the length-based filter presented in Section 2.2.3. Not only does this filter improve performances (a gain of 2 to 40 absolute points in TOP@1 is observed depending on the configuration tested), it also gives an appreciable speed up (2 to 10, depending on the variants).

Incidentally, we reproduced a proxy to systems that would only consider hapax words, somehow similarly to (Enright and Kondrak, 2007; Patry and Langlais, 2011). The best variant we obtained lagged far behind other variants exploiting all the available text. One reason for this bad result might simply be that only collection-wide hapax terms are considered here.

The impact of the post-processor can be observed in Table 2. With the exception of the `url` variants, applying a post-processor improves TOP@1, a finding that corroborates the observations made in (Rebout and Langlais, 2014). We do not observe a huge performance difference between the algorithms. For the final submission, we applied the `o-greedy` algorithm because the others could not handle the size of the data set[13].

	`url`	`text`	`both`
w/o	80.1	83.3	88.6
`o-greedy`	79.7	87.6	91.6
`b-greedy`	80.7	87.9	92.1
`hungarian`	80.4	87.9	**92.1**

Table 2: TOP@1 of the post-processors we tested.

4 Analysis

4.1 Sensitivity to Source Document Size

We explored how our variants behave as a function of (source) document length. Figure 2 reports the cumulative accuracy of selected variants as a function of document size. We observe (red curve) the tendency for the RALI variant (the one we submitted) to globally improve as source documents get larger. Comparing the two dotted green curves, we also see that the benefit of embedding translation increases with document size. It is not entirely clear why we observe an increase in performance of the `url` variants as document size increases, since only the `url`s are considered. There are not many documents with a very short size, therefore the very first point of each curve is likely not significant.

[11] We downloaded it from: `http://www.perseus.tufts.edu/hopper/stopwords`

[12] Keeping all translations is in this case preferable to keeping only one translation.

[13] Without deep modifications of the algorithms.

	Almost no text inside
src	http://rehazenter.lu/en/medical/explorations_fonctionnelles/explorations_posture/laboratoire_de_biomecanique
trg	http://rehazenter.lu/fr/medical/explorations_fonctionnelles/explorations_posture/laboratoire_de_biomecanique
src	http://www.dakar.com/2009/DAK/RIDERS/us/equipage/57.html
trg	http://www.dakar.com/2009/DAK/RIDERS/fr/equipage/57.html
	Reference problem
src	http://www.nauticnews.com/en/2009/06/23/burger-boat-company-launches-151-03-fantail-motor-yacht-sycara-iv
trg	http://www.nauticnews.com/2009/07/13/ishares-cup-2009-a-bord-dholmatro

Table 3: Examples of problematic pairs of urls found in the development set.

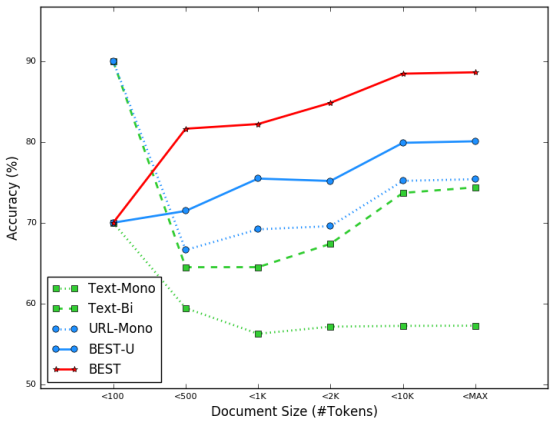

Figure 2: Accuracy (TOP@1) as a function of document size (counted in tokens).

4.2 Error Analysis

We conducted a small-scale analysis of the errors made by the RALI configuration. First of all, we observed frequent cases where a French page contains a fair amount of English material (which might explain part of the performance of monolingual variants). We also noticed that a given document has often several associated urls. In such a situation, our system will almost invariably pick the largest url (more tokens do match), which is not necessarily the case of the reference.

In the 1.7% cases of RALI could not identify the expected target document in the top-100 positions, we observed that many documents contained almost no text. Typical examples are provided in Table 3. In such cases, the url-based approach should be more efficient. This means that learning which variant to trust given a source document could be fruitful. We also observed inevitable reference errors (see the bottom line of Table 3 for an example). Last, we noticed that some documents are rather specific, and our lexicon does not help much the translation process. This is the case for the document shown in Figure 1.

5 Conclusion

Our participation in the shared task has been carried out thanks to the Lucene framework. We devised a number of configurations by varying the parameters of the *MoreLikeThis* query mechanism, as well as by exploiting other built-in features. We notably found a simple yet efficient way of matching documents thanks to their urls, which outperforms the baseline provided by the organizers. We also observe that querying the target collection with queries built without translation already achieves a decent performance and that involving a translation mechanism as simple as using a bilingual lexicon gives a nice boost in performance. We also propose to filter target documents based on the length of the source document. This not only improves results, but also speeds up retrieval. Last, we measured that applying a post-processor (such as the Hungarian algorithm) further improves performance.

The best system we identified on the development set combines (in a single query) terms translated from the source document as well as terms from its url. A length-based filter is applied, as well as a post-processor (Hungarian algorithm). This system achieves a TOP@1 of 92.1, and a TOP@100 of 98.6, a respectable performance for such a simple system.

We are currently investigating whether better performance can be obtained by using machine translation instead of the lexicon-based translation approach used here.

Acknowledgments

This work has been founded by the *Fonds de Recherche du Québec en Nature et Technologie* (FRQNT).

References

Jessica Enright and Grzegorz Kondrak. 2007. A fast method for parallel document identification. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 29–32.

H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Comput. Linguist.*, 31(4):477–504, December.

Alexandre Patry and Philippe Langlais. 2011. Identifying parallel documents from a large bilingual collection of texts: Application to parallel article extraction in wikipedia. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 87–95.

Lise Rebout and Philippe Langlais. 2014. An iterative approach for mining parallel sentences in a comparable corpus. In *LREC*, pages 648–655, Rekjavik, Iceland.

Serge Sharoff, Reinhard Rapp, and Pierre Zweigenbaum. 2013. Overviewing Important Aspects of the Last Twenty Years of Research in Comparable Corpora. In Serge Sharoff, Reinhard Rapp, Pierre Zweigenbaum, and Pascale Fung, editors, *Building and Using Comparable Corpora*, pages 1–17. Springer Berlin Heidelberg, January.

Jason R. Smith, Herve 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 the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1374–1383.

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.