Innovation Action (IA)

ICT-14-2016-2017

H2020-ICT-2017-1

Enabling procurement data value chains for economic development, demand management, competitive markets and vendor intelligence

THEY BUY FOR YOU

Deliverable D3.1

Cross-lingual Document Similarity Service v1

Date	15/06/2018
Author(s)	Matej Kovačič, Janez Brank (JSI)
Dissemination level	Confidential
Work package	3
Version	3

Document metadata

Quality assurers and contributors

Quality assurer(s)	Ian Makgill, Till Christopher Lech, Ahmet Soylu
Contributor(s)	Matej Kovačič, Janez Brank

Version history

Date	Version	Description
12/06/2018	1	Initial report submitted for review
18/06/2018	2	Report rewritten and reformatted to reflect feedback and discussion from review
29/06/2018	3	Report revised in light of further review

Executive summary

This report describes deliverable D3.1, which goal is to develop a methodology and tools for automatic comparison of public orders and spending documents across different languages. It provides a brief description of document similarity computation service APIs and example Python application to call this service.

Table of contents

1 Objectives

The main goal of work package WP3 is to develop methodology and tools for automatic real-time monitoring and analysis of public spending information published in different languages. Governments on local and European level produce a constant stream of documents, e.g. procurement notices, prior information notices, public orders etc. These documents are commonly published in the official language of the respective country. Some of these documents, for instance those published in Tenders Electronic Daily (TED), are multilingual, but usually the documents in the language of the issuing country are typically longer and a lot more detailed than their translations in other languages.

Our goal is to develop automatic means for analysing this documents, i.e. providing the means to compare and link documents across languages. Using the developed cross-lingual tools and machine learning methodology we are developing a real-time monitoring and analytics framework for public spending and procurement intelligence.

The main goal of task T3.1 is to develop a methodology and tools for automatic comparison of public orders and spending documents across different languages.

In natural language processing a typical approach to document analysis is to represent each document as a vector of semantic concepts and terms from the document resulting in different vector space for each language. In order to compare vectors from different spaces we use an approach based on canonical correlation analysis to find a mapping to a common semantic vector space which preserves document similarities within individual languages while enabling their comparison across languages. A similar approach was already successful used in the past for comparison of news articles. The main challenge in this work package is to adapt this approach to the domain of public spending documentation.

For deliverable D3.1 we have developed a cross-lingual document similarity computation service. Service is deployed as a web-service used through a RESTful API.

2 Background

Since input are text documents in different languages, we must compute the similarity with removed effect of the language. There are several methods to compute similarity from multi lingual documents: statistical cross-lingual computation, semantic cross-lingual computation and similarity computation through machine translation. We have implemented all three methods in our document similarity computation service.

The document comparison service builds on top of several existing services to provide cross-lingual comparison of documents based on several different representations:

(1) **Wikipedia concepts.** We call the JSI Wikifier service ･Wikipedias are mapped into their English-language equivalents using the Wikipedia’s cross-language links. Various measures of similarity are then computed between the resulting sets of annotations for each input document:

- Intersection, i.e. the number of shared concepts (concepts that appear as annotations for both input documents)

- Jaccard measure: the size of the intersection of the two sets of concepts divided by their union.
- Cosine measure: we can think of the annotations of a given document as forming a vector in which each component corresponds to one concept and its value is the numeric relevance score returned by the Wikifier (i.e. the relevance of this document to this document). If a certain concept was not proposed as an annotation of that document at all, its corresponding component in the vector is set to 0. Two vectors (resulting from two input documents) can then be compared using the cosine measure, i.e. by calculating the cosine of the angle between those two vectors (using the dot product).

Both the cosine and the Jaccard measure return values in the range from 0 to 1, with higher values indicating greater similarity. Two identical documents would have the similarity measure 1, while two documents that have no annotations in common would have the similarity measure 0.

The JSI Wikifier currently supports more than 100 languages, though the quality of annotations depends on the size of the Wikipedia in that language.[1]

Note: the approach used by the JSI Wikifier to generate annotations works best for documents of moderate length, e.g. a typical news story (from a few hundred to a few thousand characters). Passing an extremely short document (e.g. less than one paragraph) is less likely to lead to good annotations. Documents that are longer than 25000 characters are rejected by the Wikifier web service to conserve CPU time and memory. If necessary, a future version of our document comparison service can work around this constraint by splitting longer documents into shorter chunks, making separate calls to the Wikifier, and combining the resulting annotations.

(2) **CCA projections.** Traditionally, in text mining, documents are represented by high-dimensional vectors using the "bag of words" model, where each dimension corresponds to one word of the input language. Thus, documents in different languages are represented by vectors in different high-dimensional spaces and cannot be compared directly. CCA (canonical correlation analysis) is a statistical technique that computes projections (mappings) from these spaces into a new, shared (language-independent) space Eucliean distance. Our service reports the cosine measure as it is independent of the length of the documents. Note that since the CCA projections are not limited to the positive orthant (i.e. they may contain negative values, even though the input vector didn't), the cosine measure between two vectors of projections is not necessarily in the range from 0 to 1 but may be anywhere from –1 to +1. It is still the case, however, that higher values indicate greater similarity.

CCA is currently available for the following languages: Catalan (`ca`), German (`de`), English (`en`), Spanish (`es`), French (`fr`), Italian (`it`), Russian (`ru`), Slovene (`sl`).

(3) **Translations.** We call a machine-translation service to translate both input documents into English. Now that they are in the same language, they can be represented by vectors using the usual bag-of-words model and similarity between them can be computed using the cosine measure. Currently the Microsoft Azure translator is used and calculating the translation-based similarity measures requires the caller to provide an Azure subscription key or authorization token. Currently the Azure service has a

1 For a list of supported languages, see http://www.wikifier.org/languages.html.

limit of 5000 characters per 1 translation call,[2] but our document comparison service works around this by automatically splitting the document into shorter chunks and making multiple calls as needed. When splitting the document, breaks are only made at sentence boundaries (as defined by the Unicode sentence breaking rules ▪), to maintain as much of the integrity of the text as possible.

To minimize the usage of the monthly quota associated with the given Azure subscription, our service caches the responses so that if the same document appears in multiple calls, it will only be translated once. If cached translations of both input documents are available, our document comparison service will report the translation-based similarity measures even if no Azure key or token was provided in the request.

As of this writing, Azure supports translation for 62 languages.[3]

3 Document comparison API

Document comparison API is described in https://github.com/TBFY/crosslinguality.

To call the document comparison web service, the user should send an HTTP request to: `http://wikifier.org/compare-docs`. Both GET and POST requests are supported, but POST is preferred due to the server limits on the maximum length of the URL.

The request should contain the following arguments in URL-encoded form:

- `doc1` = the text of the first document;
- `lang1` = two-letter language code of the language of the first document in ISO 639-1 format (eg. `en` = English, `sl` = Slovene, `de` = German);
- `doc2`, `lang2` = the same for the second document.

To represent non-ASCII data in the input documents, encode the text using UTF-8 and then apply %-encoding (e.g. *Beyoncé* → `Beyonc%C3%A9`).

If you want the service to calculate similarities based on translations into English, you should also pass one of the following two arguments:

- `azureKey` = a Microsoft Azure subscription key ;
- `azureToken` = a temporary Microsoft Azure authentication token.[4]

The response will be a JSON object containing the results of the comparison. This object contains the following attributes:

- `wikiIntersection` = number of Wikipedia concepts that were common to both documents.

- `wikiJaccard` = Jaccard measure between the sets of Wikipedia concepts for the two

2 https://docs.microsoft.com/en-us/azure/cognitive-services/translator/reference/v3-0-translate
3 The list of features supported for each languge is available through Azure's Languages API call: https://docs.microsoft.com/en-us/azure/cognitive-services/translator/reference/v3-0-languages
4 This option is preferable as it is more secure (the token is only valid for 10 minutes). For details on the Azure authentication options, see Microsoft's documentation: https://docs.microsoft.com/en-us/azure/cognitive-services/translator/reference/v3-0-reference#authentication

documents.

- `wikiCosine` = cosine similarity between the vectors of Wikipedia annotations for the two documents. Unlike the intersection and Jaccard measures, this measure takes into account the fact that each annotation has a numeric score (a higher score indicates that the annotation is believed to be more relevant). Thus the set of annotations with their weights can be interpreted as a vector, and the cosine measure can be computed between two such vectors.

- `ccaCosine` = cosine similarity between the vectors representing the two documents in a shared semantic space, obtained by projecting the TF-IDF based representations from language-specific input spaces into a common space using CCA (canonical correlation analysis).

- `translationCosineBinNoSw`, `translationCosineBinSw`, `translationCosineTfNoSw`, `translationCosineTfSw` = cosine similarity between the translations of both input documents into English. The translated documents are represented by vectors using the bag-of-words model. Several slightly different representations (and hence slightly different values of the cosine measure) can be obtained under this model depending on whether the vectors are binary (`Bin`; each component indicates whether a word is present in the document or not) or based on term frequency (`Tf`; each component indicates the number of occurrences of a word in the document), and whether stopwords are removed (`NoSw`) or kept (`Sw`).

- `doc1` = details about the first document: this is an object that includes the text of the document, a list of Wikipedia annotations and their relevance scores (extracted from the output returned by the JSI Wikifier), the projection of the document into a shared semantic space obtained using CCA (a list of floating-point values representing the components of a 500-dimensional vector), and the English version of the document obtained using machine translation (this is given as a list of all the requests made to Azure's translation service, with the input strings for each request and the corresponding response from Azure).

- `doc2` = like `doc1`, but for the second document.

4 Example of a Python3 application to call the document comparison web service

For an illustration we provide an example of a Python 3 application to call the document comparison web service. Python application is also available at Github repository at: https://github.com/TBFY/crosslinguality. Application takes 4 input arguments, first document (in text format, UTF-8 encoded), language of the first document, second document and language of second document.

Python 3 application:

```
import urllib.parse, urllib.request, json

def CompareDocs(text1, lang1, text2, lang2):
    # Prepare the request.
    data = urllib.parse.urlencode([
        ("doc1", text1), ("lang1", lang1),
        ("doc2", text2), ("lang2", lang2),
        # ("azureKey", "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"),
        ])
    url = "http://www.wikifier.org/compare-docs"
    # Call the server and read the response.
    req = urllib.request.Request(url, data=data.encode("utf8"), method="POST")
    with urllib.request.urlopen(req, timeout = 60) as f:
        response = f.read()
        #g = open("response.txt", "wb"); g.write(response); g.close()
        response = json.loads(response.decode("utf8"))
    # Output the results.
    print("Similarity based on Wikifier annotations:")
    print(" - Cosine measure: %g" % response["wikiCosine"])
    print(" - Intersection: %d" % response["wikiIntersection"])
    print(" - Jaccard measure: %g" % response["wikiJaccard"])
    print("Similarity based on CCA projections:")
    print(" - Cosine measure: %g" % response["ccaCosine"])
    if "translationCosineBinSw" in response:
        print("Similarity based on translations into English:")
        print(" - Cosine measure over binary vectors, stopwords removed: %g" %
response["translationCosineBinNoSw"])
        print(" - Cosine measure over binary vectors, stopwords kept: %g" %
response["translationCosineBinSw"])
        print(" - Cosine measure over TF vectors, stopwords removed: %g" %
response["translationCosineTfNoSw"])
        print(" - Cosine measure over TF vectors, stopwords kept: %g" %
response["translationCosineTfSw"])

import sys
if len(sys.argv) != 5: print("Usage: sampleCall.py fileName1.txt lang1
fileName2.txt lang2"); sys.exit(0)
with open(sys.argv[1], "rt", encoding = "utf8") as f: doc1 = f.read()
with open(sys.argv[3], "rt", encoding = "utf8") as f: doc2 = f.read()
CompareDocs(doc1, sys.argv[2], doc2, sys.argv[4])
```

After running the application we get the following result:

```
python3 sampleCall2b.py doc1-utf8.txt de doc2-utf8.txt en

Similarity based on Wikifier annotations:
 - Cosine measure: 0.0549141
 - Intersection: 1
 - Jaccard measure: 0.0204082
Similarity based on CCA projections:
 - Cosine measure: 0.45718
Similarity based on translations into English:
 - Cosine measure over binary vectors, stopwords removed: 0.0718782
 - Cosine measure over binary vectors, stopwords kept: 0.162828
 - Cosine measure over TF vectors, stopwords removed: 0.0373881
 - Cosine measure over TF vectors, stopwords kept: 0.712192
```

5 Conclusions and future work

We have presented a cross-lingual document similarity service that computes several similarity measures based on different representations (independent of the input language) of the input documents. In the future, this approach could be extended with additional representations and/or similarity measures. One of the challenges of measuring document similarity, whether monolingual or cross-lingual, is that the very notion of how similar two documents are is only vaguely defined and, in practice, may depend on the application needs (i.e. what the user actually needs the similarity scores for). Thus, one possible extension of this work in the future would be, in case some training data becomes available, to use machine-learning techniques to train a model (e.g. a weighted sum that combines the similarity measures described so far into a new measure tailored to the application requirements as embodied in the training data.

6 References

- J. Brank, G. Leban, M. Grobelnik. Semantic annotation of documents based on wikipedia concepts. *Informatica*, 42(1):23-32 (2018).
- J. Rupnik, A. Muhič, G. Leban, P. Škraba, B. Fortuna, M. Grobelnik. News across languages – cross-lingual document similarity and event tracking. *Journal of Artificial Intelligence Research*, 55:283-316 (2016).
- J. Brank, G. Leban, M. Grobelnik. A high-performance multithreaded approach for clustering a stream of documents. Proc. of the Slovenian Conference on Data Mining and Data Warehouses (SiKDD 2014), Ljubljana, Slovenia, 6 October 2014.
- Mark Davis (ed.). *Unicode Text Segmentation*. Unicode Standard Annex #29, version 11.0.0, 22 May 2018. http://unicode.org/reports/tr29/ (accessed 12 June 2018).