POLITECNICO DI TORINO
Master Degree in Computer Engineering

Master Degree Thesis

Improving Document Summarization Using Crosslingual Word Embeddings

Supervisors
Luca Cagliero
Paolo Garza

Candidate
Catia Blengino

Anno accademico 2019-2020

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last few years, thanks to the spread of social networks and the rise of news web pages, we have had a huge growth in data traffic over the Internet. Every day a large amount of information, regarding the most disparate topics, is published on different sites. It is impossible for a user to manually examine all the available material of his interest. That's why text summarization has become an important and useful tool. The idea behind is to group documents by topic and extract a summary including the most salient aspects. Another potential problem is the text is written in various languages, often not understandable by end users. Cross-lingual summarization (CLS) aims at solving this problem. It allows to extract a summary in a different language from the one of the source documents, generally using translations tools to translate input texts into the target language.

In this thesis we address a variant of the cross-lingual summarization task, in which we suppose to have a mix of documents in the target and other languages. Similar to CLS, the summary is expected to be in the target language and consists of a selection of sentences picked from the input documents in the target languages. Unlike traditional summarization tasks, the summary should reflect also the content of the input documents written in the other languages, i.e., sentence selection in the target language is influenced by the presence of related content in the other languages.

The main contributions of this thesis can be summarized as follows:

- it presents a variant of the official CLS task
- it proposes a graph-based and a distance-based summarization approaches

that used multilingual word embedding model to tackling the newly proposed task

- it investigates the influence of using different metrics in measuring the similarity between sentences in the embedding space

The typical use case of this system is to generate a summary in a particularly complex target language (e.g. Hindi) for which, generally, few documents are available and their word embedding models contain representations for a limited number of words. To solve the problem, domain knowledge is added in another more common language (e.g. English) to improve the quality of the summaries in Hindi.

In practice, a centroid-based summarizer and several alternative versions of TextRank are proposed. The latter differ in the way sentence embeddings are obtained and in the measure of similarity used. The different types of sentence vector are derived by combining word embedding in various ways. In particular, Word2Vec and FastText are the embedding models used.

The dataset the methods are tested on is MultiLing Pilot 2011 Dataset. It provides articles in several languages (Arabic, Czech, English, French and Hindi) and from 2 to 5 human summaries for each topic and each language to allow to evaluate the generated summaries. Several tests were performed using as input English documents plus those in one of the other languages mentioned above and by considering 5 or 10 articles for each of the source languages and also only 10 articles in the target language. Furthermore all the algorithms proposed have been compared to their alternative version which uses Google Translator and to extractive state-of-the-art summarizers, like TextRank, LexRank and CoReRank. The quality of the generated summaries was assessed quantitatively by measuring the similarity with summaries generated by experts.

1.1 Structure of the Thesis

This thesis is organized as follows:

- Chapter 2 makes an overview of the main existing text summarization techniques. Also an overview of the most relevant works in the field of extractive text summarization and cross-lingual summarization is then provided.
- Chapter 3 first shows which are the main techniques of vector representation of the text, dealing with both word embeddings and sentence embeddings methods. In addition, two ways to align word embedding models are described. In the final part, the similarity measures used in this thesis are illustrated.
- Chapter 4 presents the proposed summarization methods and the various stages that make them up: preprocessing, model alignment, sentence representation and sentence selection.
- Chapter 5 describes the tests carried out and the relative results.
- Chapter 6 contains the conclusions about this thesis work and what could be the future works.

Chapter 2

Text Summarization: State-of-the-Art

2.1 Text Summarization Techniques

Text Summarization[11] is a relevant NLP task. It turns a source text into a shorter version, trying to preserve the overall meaning. The main advantage of having a summary is that it allows you not to have to read the whole source text, thus saving reading time. A good summarization system should be able to maintain the different topics covered in the document, aiming to keep redundancy to a minimum.

The following part provides an overview of the main summarization techniques, although in this thesis only extractive methods for cross-lingual text summarization are proposed.

2.1.1 Abstractive and Extractive Summarization

There are two main approaches for the automatic generation of summaries: abstractive and extractive summarization.

- **Abstractive summarization:** it consists in interpreting and reworking the original text through the use of advanced language analysis techniques to generate a summary that contains the most critical and important information. It is based on the selection of words according to their semantics, but also on those words not present in the source documents. It's a more advanced technique and is comparable to the method

used by humans to read a text and summarize it with their own words. Researchers are very interested in this approach because of its potential.

- **Extractive summarization:** the summary is generated by concatenating only some selected sections of the input text, such as sentences or even paragraphs. Therefore the document is first divided into subsections and then the most important are selected. The importance is based on statistical or linguistic features of sentences.

2.1.2 Single-Document and Multi-Document Summarization

Based on the number of documents provided as input, the text summarization techniques are classified into:

- **Single-document summarization:** the summary obtained is representative of a single input document.
- **Multi-document summarization:** it aims to extract a meaningful summary of a group of texts. The input documents must all concern the same topic, although in each one it will be described from different perspectives and putting emphasis on different aspects. It allows individual users to quickly become familiar with the information contained in a large group of documents. Between the two approaches, it's the one that best suits the information redundancy on the internet.

2.1.3 Multi-Lingual and Cross-Lingual Summarization

A further subdivision is made based on source and target languages.

- **Multilingual Summarization:** it consist in a summarization algorithm who allows to generate a summary of a document set where all the documents share the same language. The language of the output summary is the same of input documents. Generally it is tested on a variety of languages.
- **Cross-Lingual Summarization**[23][22]**:** its goal is to obtain a summary in a different language from the one of the source documents.

  First studies in cross-language text summarization analyzed the information in only one language.

The typical schemes of CLTS are the early and late translations (see Figure 2.1). The first scheme consists in summarizing the input documents translated into the target language, using only the information of the translated sentences. Instead, the other scheme does exactly the reverse. Firstly it gets the summary, then translates it into the target language.

Some approaches use a translation quality score to improve the quality of cross-language summarization. Generally a human translation is provided as a reference to make a comparison between it and the system translated text because the aim of machine translation evaluation is to assess the correctness and quality of the translation.

More recent methods generate summaries by taking into account informations from both languages. However the quality of the summary could be lower as the performances may vary by languages.

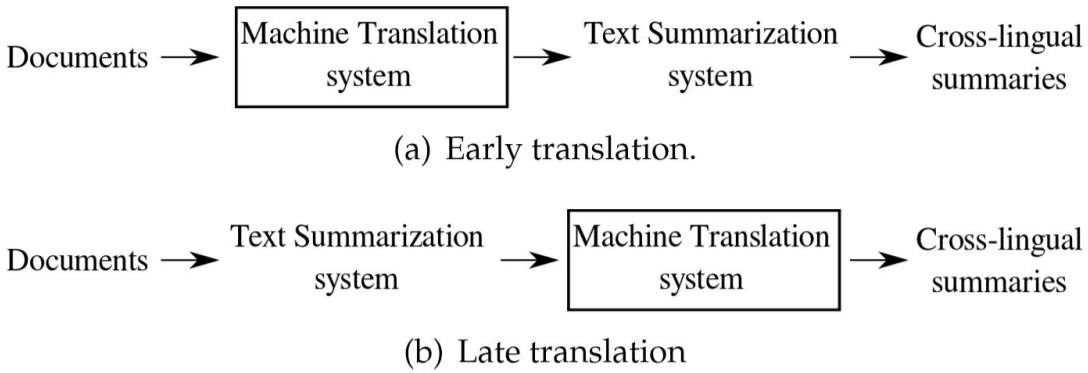

Figure 2.1. Early and late translation schemes for cross-lingual summarization. In the first case, documents are translated before applying the summarization algorithm. In the second one, after extracting the summary, it is translated into the target language.
(image from [22])

2.2 Related Works

This section is organized as follows: in the first are shown part extractive summarization methods generally used as baselines, in the second part some multilingual documents are described, instead the third part will focus on the main existing cross-lingual summarization methods.

2.2.1 Extractive Summarization

Many approaches that are used as baselines for extractive summarization are graph-based. The two most famous are Lexrank[7] and TextRank[24] and both of them are inspired by Google PageRank algorithm[4].

PageRank is used for online searches and allows to classify the resulting web pages. To each page it associates a rank based on the contained links pointing to another page. The score corresponds to the probability that a user, clicking on one of the links, will end to visit a particular site. The probabilities of a user to go from a page to another one are collected in a square matrix. Then, the values in it are iteratively updated to get the final page rankings. The modified version of this algorithm is used in TextRank and LexRank. In these cases, instead of web pages they consider the sentences. The probability of a user visiting one sites from another is replaced by the similarity between two sentences. As before, these scores are then stored in a square matrix.

LexRank [7]

It is an unsupervised method for text summarization. Since it is an extractive technique, it is based on the identification of the most relevant phrases in a document or set of documents. To do this, it represents the sentences in a graph and, based on the concept of eigenvector centrality, derives how much important each sentence actually is. The idea is that if a sentence is similar to many others, then it is probably a phrase of high importance.

LexRank uses as similarity measure among sentences the IDF-modified cosine of TF-IDF vectors. The graph is created by associating to each vertex a sentence and by linking two sentences with an edge every time the similarity between them is above a threshold. At this point, PageRank is applied to rank sentences. Among the outputted ranked sentences, the top-n are

selected to become part of the summary. LexRank also adds a passage in which it processes the top-n sentences so that those choices are not too similar to each other.

TextRank [24]

As a graph-based ranking algorithm, it consists in identifying the vertices that are most important inside a graph, basing the decision on all the information contained within the graph.

After the subdivision of the input text into phrases, it generates the matrix by calculating the similarity among each pair of sentences based on the number of words two sentences have in common, which is then divided by the length of the sentence. From it, a graph is obtained and then is applied the PageRank ranking algorithm to derive the scores of each sentence. In the graph each vertex corresponds to a phrase and the edges are the scores between sentences. The last step is to select the top-n ranked sentences to extract the final summary.

This method is the one mainly adopted in this thesis. In particular, it has been enriched with the use of word embeddings and has been tested considering various ways of obtaining sentence embeddings.

Rossiello et al. (2017) [29] proposed a centroid-based method. Unlike the techniques mentioned so far, this one exploits word embeddings. What it does is use TF-IDF to find the words to include in the centroid and select the phrases that are most similar to the centroid vector. Affinity is computed with cosine similarity.

This latter method has also been implemented in this thesis and adapted to become a cross-lingual summarizer instead of a mono-lingual one.

Kobayashi et al. (2015) developed two methods that use embeddings to compute the similarity between documents: DocEmb and EmbDist ,that differs for the objective functions used [18].

2.2.2 Multilingual Summarization

The characteristic of these types of summarizers is to be completely independent from the input language. An example is the JRC summarizer[33] proposed by Steinberg et al. (2011). It makes use of the Latent Semantic

Analysis and applies the Singular Value Decomposition (SVD) to a term-by-sentence matrix, using the output to select the most important sentences. It is the summarizer that reached top results in the MultiLing Pilot of the TAC 2011 contest.

Another multilingual summarizer is the one developed by Cagliero et al. (2019) . It's based on LSA and on itemset extraction. The summary is obtained by selecting sentences in which the largest number of frequent itemsets are present, but with the aim of minimizing redundancy [5].

Radev et al. (2000) have implemented MEAD [28], a summarizer that uses TFIDF to represents phrases and creates clusters of sentences. Based on the characteristics of the sentences and the calculation of cosine similarity, some sentences from each cluster are picked to become part of the summary.

2.2.3 Cross-Lingual Summarization

First studies in this field consider only the information of one language. More recent methods have instead tried to improve performance by considering a translation quality score and information both in the source language and in the target language.

Wan et al. (2010) [38] proposed a method for English to Chinese cross-language summarization which that takes into account the translation quality of each sentence. For all the English sentences, this score is first predicted. They used a Support Vector Machine (SVM) regression method for this calculation. Then they used it for the summarization task in combination with the informativeness scores. After the selection of the most relevant English sentences, they are translated in Chinese.

Wan (2011) in [37] developed two ways to obtain a summary in a target language starting from documents in a different source language. He proposed to use informations in both languages. In particular he considered only English and Chinese. The methods proposed are Simfusion and CoRank, both graph-based algorithms.

SimFusion [37][14]

It uses English-side information and Chinese-side information together to produce a summary. Starting from a set of documents in English, each sentence is translated into Chinese. Later, for each pair of sentences, the similarity is calculated, but by merging the similarity scores obtained in both languages. Finally it uses an algorithm similar to PageRank to extract the most important sentences which is applied to a graph built by considering each node as a sentence and the similarity scores as edges. The phrases selected are only the Chinese one and in case of multidocument summarization, a greedy algorithm reduces importance of highly scored sentences.

This method is almost similar to the one proposed in this thesis work. The main differences consist in the additional use of word embedding models and in the non-use of automatic translation tools. A further difference is the calculation of the scores. It is based on the cosine similarity which is computed between each pair of sentences (also between a sentence in the source language and a sentence in the target language).

CoRank [37][14]

Similar to SimFusion, it is graph-based too. It ranks both English and Chinese-translated sentences at the same time. The main idea is that the importance of a sentence depends on those to which it is connected, whether they are of the same language or of the second language. In particular this method relies on three kinds of sentence relationships: English-English, Chinese-Chinese, English-Chinese (see Figure 2.2). Based on this, three affinity matrices are calculated:

- M_{ij}^{cn}: describes the cosine similarity between Chinese sentences i and j
- M_{ij}^{en}: describes the cosine similarity between English sentences i and j
- M_{ij}^{cn}: represents the affinity between English sentence i and the Chinese sentence j

Wan et al.(2018) [39] proposed a new framework. They first extract many candidate summaries and analyzed them to improve quality. These summaries are obtained in different ways and with different translation tools. Then, they proposed a new method to classify this candidates which is an ensemble ranking that exploits biligual informations.

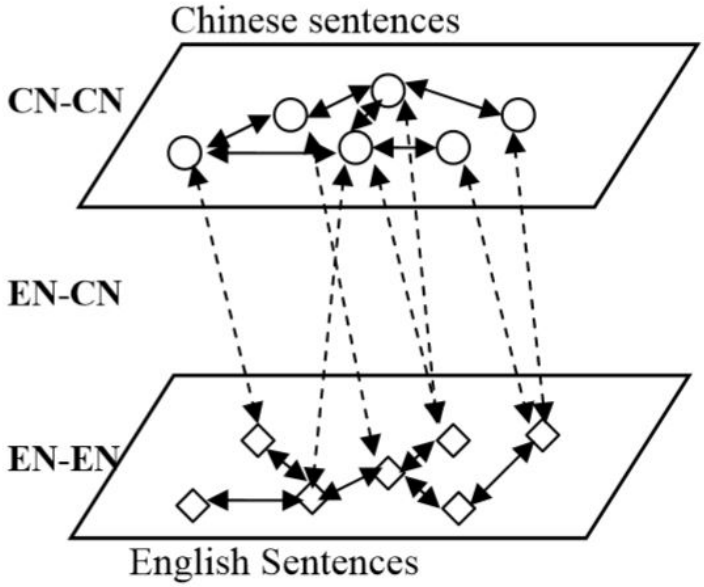

Figure 2.2. Relationship between English sentences (en-en), Chinese sentences (cn-cn) and between an English and a Chinese sentence (en-cn).
(image from [37])

Chapter 3

Vector Representations of Text

In Natural Language Processing there's the need to transform text into something understandable by the machine. There are several text representation techniques but not all of them are equal, consequently the choice of the method can influence the performances [31].

3.1 Word Representation

The following section describes word embeddings that is the representation used in this thesis.

3.1.1 Word Embeddings

It's a technique capable of providing a numerical representation of the words of a vocabulary. It associates to each single word a vector of real numbers in a predefined vector space and is able to capture the relationship with other words in such a way that similar terms have similar representations.

The dimensionality of the vector is low compared to the number of words in the vocabulary. In order to achieve good results, during the learning phase, it needs very large texts that include as many words as possible.

Word2Vec [26]

It is an efficient method used for word embeddings, developed in 2013 at Google by a team lead by Tomas Mikolov. These are neural networks with an input layer, a projection layer and an output layer, trained to reconstruct the linguistic contexts of words.
It provides a vocabulary in which each term has associated a vector and is able to capture the relationships between words. In fact it has been noticed that word vectors manage to capture many linguistic regularities and generates useful properties such as linear relationship (see Figure 3.1).

A largely used example is the analogy *'king is to queen as man is to woman'*. In vector space the difference between king and queen is similar to the one between man and woman. This implies that a vector operation such as:

$$vector('king') - vector('man') + vector('woman')$$

results in a vector very close to *vector('queen')*.

There are two methodologies for obtaining a Word2Vec model: SkipGram and Common Bag Of Words (CBOW).

- **Skip-Gram model:** given an input word, it gives the probability for each word in the vocabulary to be the closest word. In this case it's a simple neural network, trained to perform a task, that will not be the one for which it will actually be used. The real goal is in fact to learn the hidden layer weights that represent the word vectors it is trying to learn.
- **CBOW model:** it tries to predict a term using context as input. For example, given the words *w(t-1)*, *w(t-2)*, *w(t + 1)*, *w(t + 2)*, the model will output *w(t)*. The context can be a single word or a set of words. The number of words in each context is determined by a parameter called *window size* which indicates how many words before and after a given term will be included in the context.

Given a window size equal to 2, if the Skip-Gram model is used, the input word will be used to predict its two previous and two subsequent words. If the CBOW model is used, given the two previous words and the next two words of the target one, the target word will be predicted.
The table 3.1 shows some examples using the phrase "*the quick brown fox*

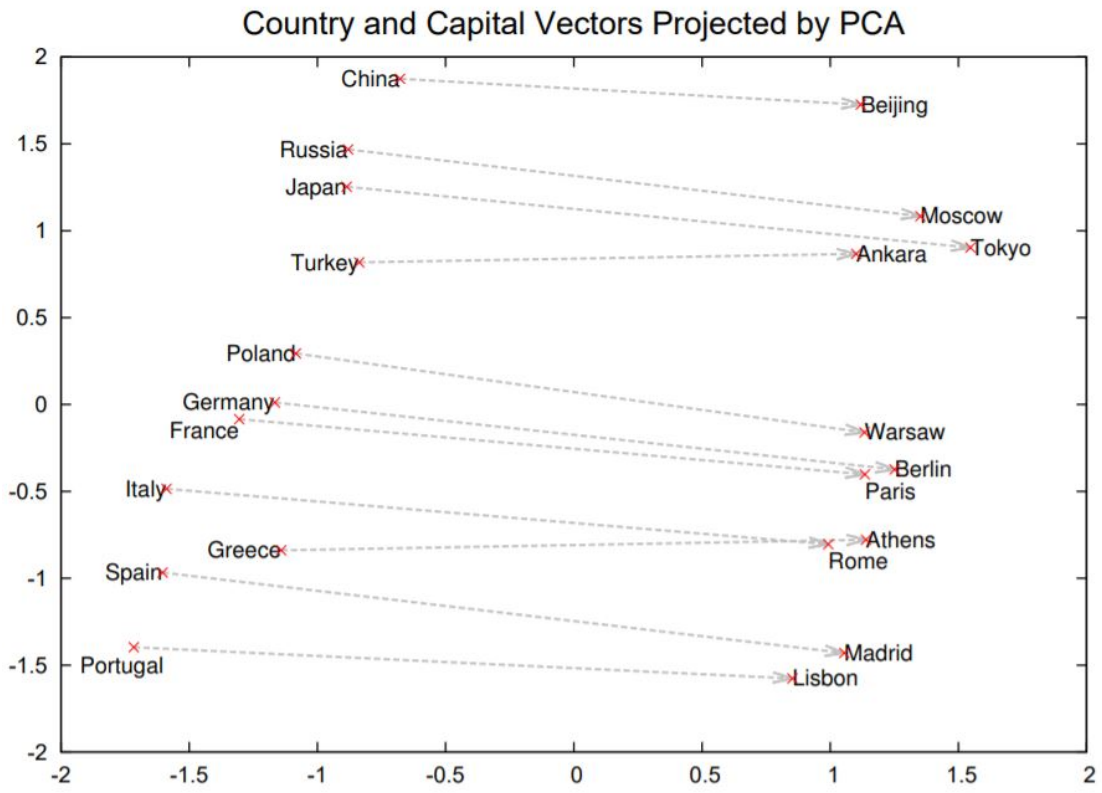

Figure 3.1. Example of two-dimensional PCA projection of the vectors of countries and their capital cities. It shows the model's ability to organize concepts and learn the relation between them. (image from Mikolov et al., 2013 [25])

jumps over the lazy dog":

According to Mikolov, CBOW represents well the most frequent words and is faster. At the same time, Skipgram better represents rare words and works best with a small amount of data.

A feature of the Word2Vec model is that it can be queried to detect relationships between words. This can be achieved with the nearest neighbor functionality.

For example, if the model is interrogated to know the 10 nearest neighbors of the word *person*, the output will be:

	Input	Target
Skip-Gram	the quick brown ...	quick,brown the,brown,fox the,quick,fox,jumps ...
CBOW	quick,brown the,brown,fox the,quick,fox,jumps ...	the quick brown ...

Table 3.1. Example of input and target of Skip-Gram and CBOW starting from the sentence: *"the quick brownfox jumps over the lazy dog"*

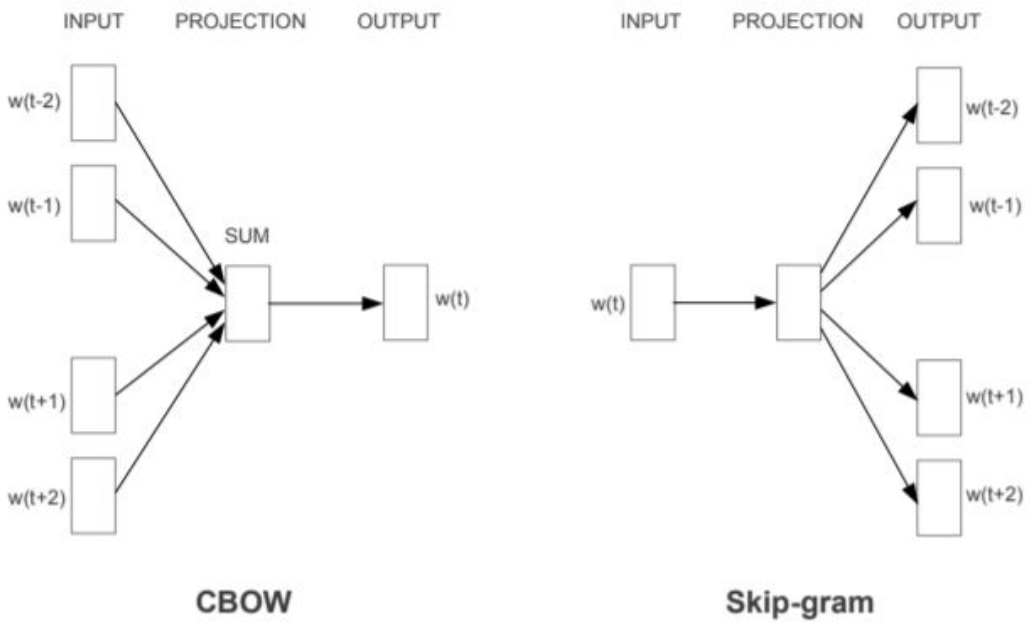

Figure 3.2. CBOW architecture uses the context to predict the current word; Skip-gram does the opposite: it predicts surrounding words based on the current word. (image from Mikolov et al., 2013 [26])

```
[('woman', 0.633482813835144),
 ('persons', 0.6320774555206299),
```

```
('someone', 0.6248975992202759),
('patient', 0.5782430171966553),
('individuals', 0.5622785687446594),
('anyone', 0.5621103048324585),
('man', 0.55032145977020 26),
('victim', 0.5433194041252136),
('defendant', 0.5410549640655518),
('child', 0.5287619233131409)]
```

They are sorted by the cosine similarity with the input word. Cosine similarity is a metric used to measure the cosine of the angle between two vectors. (see Section ...)

Word2Vec is not the only existing word embedding method. Other popular models in use today are GloVe and FastText.

GloVe [27]

GloVe, abbreviation of Global Vectors, is an unsupervised learning algorithm for obtaining vector representations for words published in 2014 by Stanford researchers. GloVe exploits global co-occurence statistics to obtain word vectors. In this it differs from Word2Vec, who rely on local context information of words ignoring whether some context words appear more often than others. The main idea underlying the model is that if words co-occur many times, it means they have some linguistic or semantic similarity.

Firstly it constructs a co-occurrence matrix and then computes conditional probability for each word. For example, consider the co-occurrence probabilities for target words *ice* and *steam* with various words from the vocabulary. As can be seen from the Figure 3.3, the probability *P(k|ice)* is higher if k is

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 3.3. Co-occurrence probabilities for words *ice* and *steam* (image from [27])

solid than when k is *gas*. Considering instead the word *steam*, it co-occurs

more frequently with *gas* than it does with *solid*.
After performing all statistical computations, the large matrix is formed, whose dimensionality is reduced afterwards.

FastText [3][15][2]

This model is a development by Facebook released in 2016. The idea is similar to Word2Vec, but it uses not just words to get word embeddings. In fact, it takes into account sub-word and character-level information. The word embeddings obtained look like the ones by Word2Vec, with the difference they are a combination of lower-level embeddings. Each word is represented by the word itself and as bag of characters n-grams.

For example, taking n = 3 and considering the word *person*, FastText representation for the character n-grams is

$$< pe, per, ers, rso, son, on >$$

The word itself ('*person*') need to be added to this group.
This approach offers two main advantages: needs less training data because much more information can be extracted from each piece of text and allows generalization as long as new words have the same characters as known ones.

3.2 Sentence Representation

Since the final objective of this thesis is to create a summary by selecting from the input texts the sentences considered most relevant, it is necessary to obtain a vectorial representation of the phrases themselves. Starting from the words representation, it is possible to get the vector of a sentence.

Mikolov et al. in [25] show the unweighted average of word vectors are a good way to meaningfully combine words of a sentence. This is the simplest method and one of those used in this thesis.

An improvement of it would be to average them proportionally to their TF-IDF or, as is done in the thesis [12], only to their IDF. In the last case, before summing all the word vectors, each of them is divided by the number of sentences in which it appears.

Another weighting function is smooth inverse frequency (SIF), proposed in [1] as a new baseline for sentence embeddings.
The following part explains TF-IDF and SIF.

3.2.1 TF-IDF

TF-IDF[35] is the abbreviation for Term Frequency - Inverse Document Frequency. It is a weighting function used in information retrieval. It allows to measure the importance of a word in a text. In fact its goal is to give more importance to less recurring words and, at the same time, reduce the one of the most frequent words, believing that those terms brought less specific information to the text. Each word is associated with a weight which grows proportionally to the number of times the word occurs in the document and inversely proportional to its occurrences in the collection.

Term Frequency

TF is the number of times the term i occurs in document d. Mathematically:

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{|d_j|}$$

where $n_{i,j}$ is the number of occurences of terms i in the document j, while the denominator $|d_j|$ is the dimension of the document j, which corresponds to the number of words in it.

Inverse Document Frequency

IDF indicates the general importance of the term and is obtained with:

$$\mathrm{idf_i} = \log \frac{|D|}{|\{d : i \in d\}|}$$

where $|D|$ is the number of documents in the cluster and the denominator represents the number of documents in which the terms i appears.
Thus IDF value is calculated based on the entire corpus.

TF-IDF formula

It is obtained from the two formulas shown above. Indeed it is calculated as:

$$\mathrm{(tf\text{-}idf)_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

Sentence Embedding with TF-IDF

The vector of a word is multiplied with the word's tf-idf score for the sentence and this is done to all the words that belong to the phrase. Then they are all added up and divided by the number of words in the sentence.
The sentence embedding is the obtained vector.

3.2.2 SIF

Aurora et al. proposed SIF [1], acronym for Smooth Inverse Frequency, an unsupervised approach for sentence embedding.
It makes the weighted average of the vectors of the words in the sentence. The weight function is $\frac{a}{a+p(w)}$, where a is a parameter tipically set to 0.001 and $p(w)$ is the estimated frequency of the word. Then, for the set of sentences, the principal component of their embeddings is computed. Finally it substracts from the sentence embeddings their projection on their first principal component. All these operations are schematized in Algorithm 1.

What SIF tries to do is to reduce the importance of words (e.g. "the", "but","and", etc.) that are quite irrelevant from a semantic point of view and, according to the authors, the component removal allows semantic information to be more dominant in the vector's direction, decreasing the amount of syntactic information contained in sentence embeddings.

Algorithm 1 SIF Embedding

Input: Word embeddings $\{v_w : w \in V\}$, a set of sentences S, parameter a and estimated probabilities $\{p(w) : w \in V\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in S\}$

1: **for all** sentence s in S **do**
2: $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in S\}$, and let u be its first singular vector
5: **for all** sentence s in S **do**
6: $v_s \leftarrow v_s$ - $uu^T v_s$
7: **end for**

All the methods described above have been used in this work, but, to give a wider view, two further methods are explained (paragraph vector and

Doc2Vec), although not used in this field of extractive summarization.

3.2.3 Paragraph Vector

More popularly known as Doc2Vec[20], is an unsupervised method to generate embedding for input sequences of variable length like sentences, paragraphs or documents. It's an adaptation of Word2Vec indeed it mappes the texts in the vector space using the Word2Vec model, but adding another feature, unique for each document: the paragraph ID.

In the paper are proposed two ways to obtain the vectors: the Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag of Words (DBOW).

PV-DM [20]

The idea is similar to the CBOW model of Word2Vec. Given a paragraph, it samples consecutive words randomly and predicts a central word from randomly sampled set of words. This is done taking context words and a paragraph ID as input (see Figure 3.4).

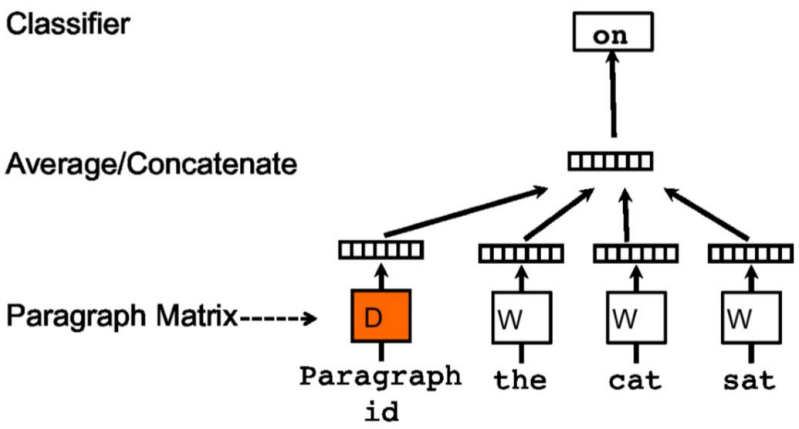

Figure 3.4. Distributed Memory Model of Paragraph Vectors (PV-DM). It uses context words and paragraph ID as input to predict a central word. (image from [20])

Every column of paragraph matrix (D) is the vector of a paragraph, while columns in matrix W represents word vectors. Paragraph and word embedding are then averaged or concatenated to predict the next word in the

context.

PV-DBOW [20]

Distributed Bag of Words differs slightly from PV-DM. It is a method which consists in forcing the model to predict the words randomly sampled from the paragraph in the output, not taking into account the context words in the input. This is shown in Figure 3.5. Since this technique doesn't need to

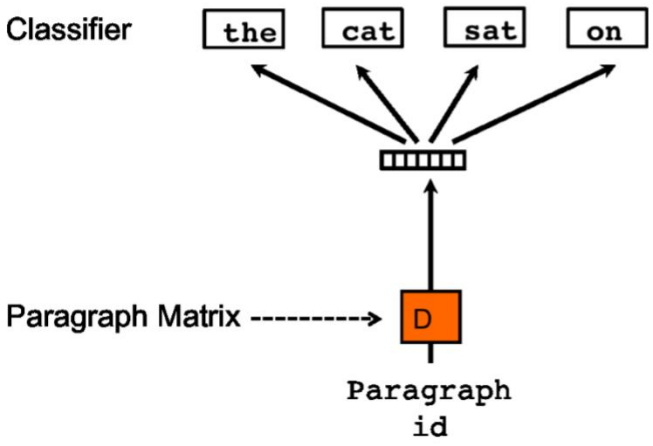

Figure 3.5. Distributed Bag of Words of Paragraph Vector. Paragraph vector is trained to predict the words in a small window. (image from [20])

save word vectors, it is faster and consumes less memory.

Skip-Thought Vectors

Skip-thought vectors, presented in the article [17] is a relatively simple method which has achieved good results in tasks like sentiment analysis and paraphrase detection. It can be viewed as the Word2Vec version for sentences. Instead of predicting context words from a specific word, it predicts the surrounding sentences starting from a certain phrase (see Figure 3.6). This is done by training an encoder-decoder model. First it encodes a sentence in a vector and then decodes that representation into the surrounding sentences.

For example, given a tuple $(s_i1, s_i, s_i + 1)$ of contiguous sentences, with s_i the i-th sentence of a text, the sentence s_i is encoded and tries to reconstruct the previous sentence s_i1 and next sentence $s_i + 1$.

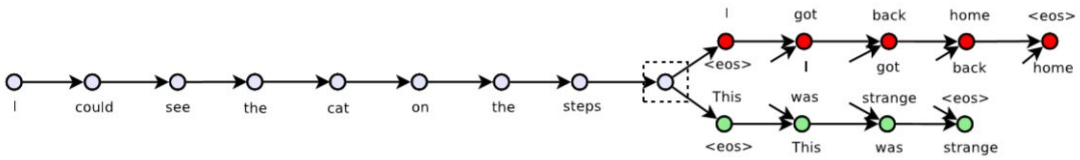

Figure 3.6. Example of the skip-thoughts model. In this case the input is the sentence triplet *"I got back home.", "I could see the cat on the steps.", "This was strange"*. (image from [17])

3.2.4 FastSent

FastSent [13] takes up the idea of skip-thought vectors, trying to improve its training time. It proposes a much simpler model, where the sum of the word embeddings is used to compose a sentence vector. The word embeddings are trained for the purpose of maximize the inner product between sentence embedding and the word embeddings of surrounding sentences.

The methods described so far who allows to get a sentence embedding from word embeddings obtained by word2vec are those used in this thesis, thus Paragraph Vector, Skip-thought vectors and FastSent are not used.

3.3 Model Alignment

The alignment of two word embedding models is a central problem in machine learning with several applications in natural language processing, including sentence translation. For the final purpose of this thesis it is necessary that words, and consequently sentences, in different languages, can be comparable in a common space. This is why alignment is needed. It generally consists in learning a rotation matrix by using a bilingual lexicon and applying the transformation to one of the model. It also makes possible to expand the lexicon because the transformation generalizes well the transformation is able to generalize well to words which were not considered during training [16].

Consider for example the English word *'earth'* and the corresponding French word *'terre'*. When the two embedding models of the two languages are trained, the English one will associate a vector to the word *'earth'* which will most likely be different from the one that for the French model corresponds to the word *'terre'*. Thanks to the alignment, the French model is

transformed in such a way that the word *'terre'* is mapped to a vector almost similar to the one of the corresponding English word. This concept is best illustrated in the figure 3.7.

Joulin et al. (2018) developed a supervised alignment [16]. It is the method used to align the FastText models used in this work. The implementation of it is available on GitHub[1].

In the next part two type of alignment are described: one based on Procrustes and an unsupervised alignment of embeddings with Wasserstein Procrustes [10].

3.3.1 Procrustes

It's a method aimed to find the orthogonal matrix that maps a given set of points to another given set of points. It needs to known a priori the one-to-one corrispondence of points between the two sets.
Given two matrices X and Y, the goal is to find an orthogonal matrix R which most closely maps X to Y. Mathematically, the orthogonal Procrustes problem is:

$$R = \arg\min_{\Omega} \|\Omega X - Y\|_F \quad \text{subject to} \quad \Omega^T\Omega = I$$

Peter Schönemann solved it in 1964 [32], showing that the solution is equal to:

$$R = UV^T.$$

where $USV^T$is the singular value decomposition of YX^T.

3.3.2 Wasserstein Procrustes

It's an approach to transform one set of data to represent another set of data as closely as possible. To align two sets of points in high dimension, it uses the joint estimation of an orthogonal matrix and a permutation matrix. In this case the one-to-one correspondences are not known [10].

Like in the Procrustes problem, the goal is to learn the orthogonal matrix such that the points of A are close to the one of B, but also to deduce one-to-one matches. It uses the Wasserstein distance to represent the distance between the two set of points.

[1]https://github.com/facebookresearch/fastText/tree/master/alignment

Combination of Wasserstein with Procrustes lead to:

$$\min_{\Omega} W_2^2(X\Omega, Y) = \min_{\Omega} \min_{P} \|X\Omega - PY\|_2^2$$

Its a non-convex and computationally expensive problem and in the paper [10] they proposed a stochastic algorithm to solve it.

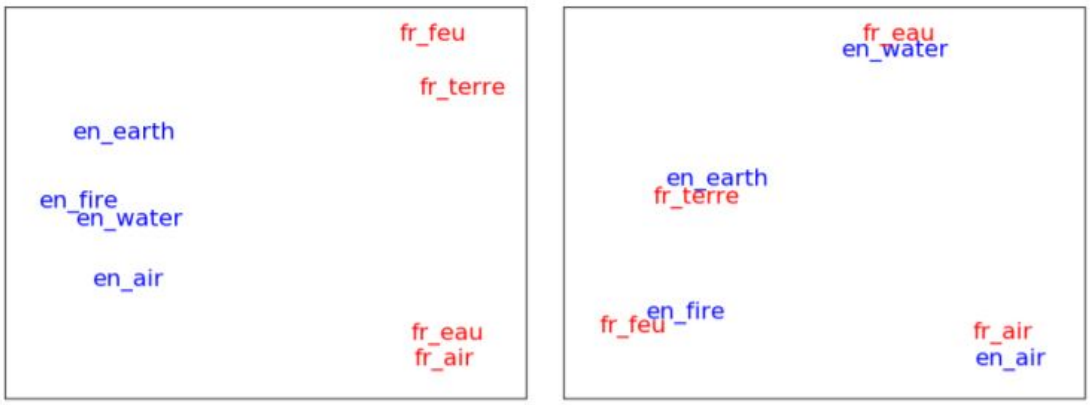

Figure 3.7. Unsupervised alignment problem for word vectors [10]. Aim: estimate the transformation to map the vectors and the correlation between the words. Left: PCA on the non-aligned word embeddings. Right: PCA on the aligned word embeddings.

3.4 Similarity Measures

3.4.1 Cosine similarity

It measures the similarity between two vectors by computing the cosine of the angle between them. It is obtained by taking the dot product of the two vectors, divided by the magnitude value. Considering a vector A and a vector B, their cosine similarity is defined as:

$$similarity = cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

This implies that if two vectors have the same orientation, their similarity will be equal to 1; a value of -1 means they have an opposite orientation; if one vector is oriented 90° with respect to the other, the cosine similarity is 0.

3.4.2 WMD

Word Mover's Distance[19] is a distance function who measures the dissimilarity between two documents even when they have no words in common. It is inspired by the Earth Mover's Distance, a well known transportation problem, but adapted to the space of documents. Indeed the dissimilarity between two documents corresponds to the minimum cumulative distance that all words in a document need to travel to exactly match another document. A visualisation of the idea is in Figure 3.8.

To calculate the distance, WMD makes use of word embeddings and normalized Bag-of-Words. The advantage of this metric is that it has no hyperparameters, but the main problem is the cost of computing the EMD.

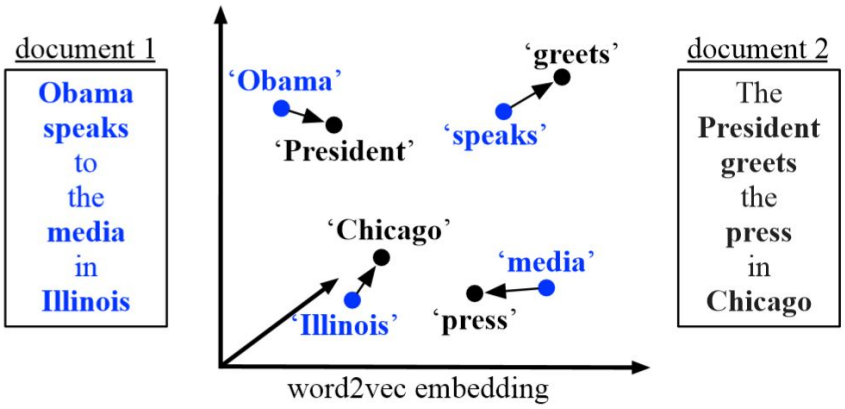

Figure 3.8. Example of the word mover's distance. Bold words are all non-stop words and are embedded into a word2vec space. The distance between document 1 and document 2 is the minimum amount of distance that the embedded words of document 1 need to "travel" to reach the embedded words document 2. (image from [19])

In the upper part of the Figure 3.9 is shown the WMD metric from sentences D_1 and D_2 to D_0. Each arrow represents towards which term the word will "travel" to, while the label corresponds to the distance. Instead the bottom part shows how WMD behaves when sentences, in this case D_0 and $D3$, differs in the number of words they contain.

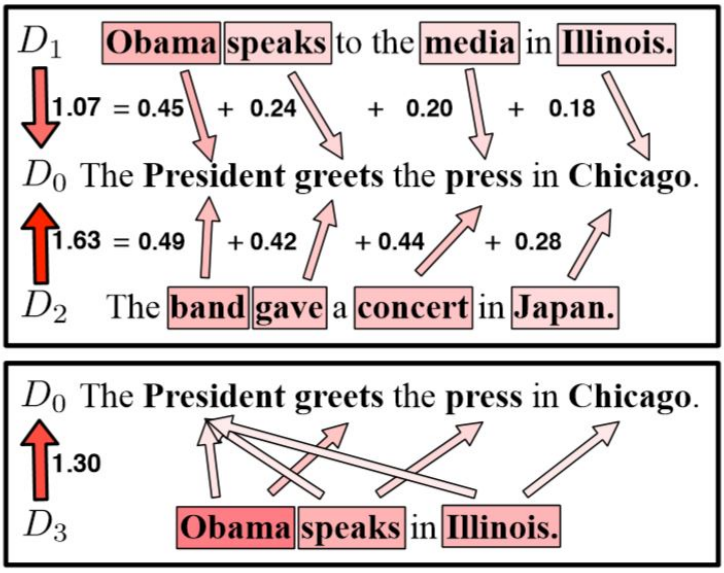

Figure 3.9. Example of how Word Mover's Distance works in two different scenarios (image from [19])

Chapter 4

Proposed Methodology

This thesis aims to obtain a cross-lingual summary trying to exploit the semantic relationships between cross-lingual content by using word embedding models. In particular, documents in two different languages (e.g. English and French) are taken as input to obtain a summary in one of them (e.g. English).
The collection of documents in non-target language can be:

- a collection of other documents
- the translated target collection (translated by human or through Machine Translation)

The typical use case of this system is to generate a summary in a particularly complex target language (e.g. Hindi) for which generally few documents are available and their word embedding models contain representations only for a limited number of words. To solve the problem, domain knowledge is added in another more common language (e.g. English) to improve the quality of the summaries in Hindi.

Only extractive methods are proposed. The main steps that constitute the extractive summarization algorithm are: preprocessing, model alignment + sentence representation, selection and evaluation (see Figure 4.1), although the latter is not strictly necessary for the summarization task.

- Preprocessing: it allows to remove irrelevant and noisy information present in the text.

- Model Alignment and Representation: consists in aligning the word embedding models and in finding a vector representation for each sentence in the source text.

- Selection: the most relevant sentences are taken and concatenated to create the summary.

- Evaluation: it is used to check the quality of the summary.

The following sections explore each phase.

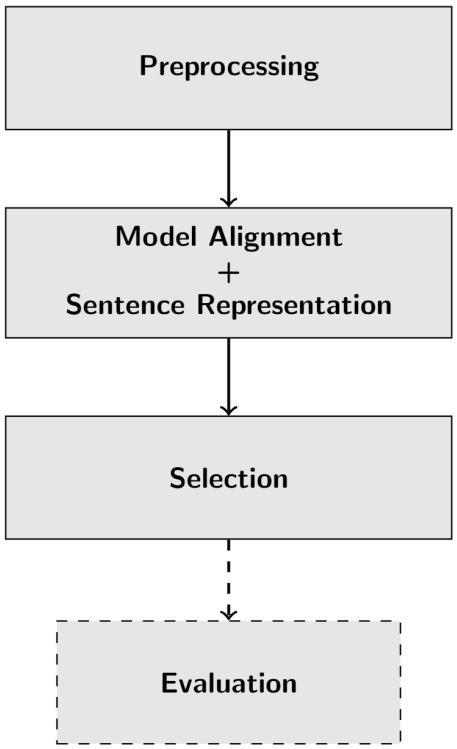

Figure 4.1. Phases of the proposed extractive summarization methods

4.1 Preprocessing

In this case it is necessary to obtain a better representation of the sentences. There are different text preprocessing techniques.

4.1.1 Lowercasing

One of the simplest way of preprocessing, useful in most of NLP problems. It consists in lowercasing all the text data. It can help when the dataset is small.

4.1.2 Tokenization

There are two types of tokenization:

- Word tokenization: is the task of splitting a document into pieces called *tokens*. The main approach is to use single space character to separate words.
- Sentence tokenization: is usually done based on punctuations such as ".", "!", "?" as they generally represent sentence boundaries.

4.1.3 Stemming

It is the process of reducing inflected words to a root form. The *root* may not correspond to the morphological root of the word, it is just a smaller or equal form of the word. In general, it is enough that related words are mapped into the same stem. What it does is to cut off the ends of the words according to a crude heuristic process. The most common algorithm for stemming English is Porter's algorithm, which is rule based.

Considering the English language and the Porter algorithm, in Tab 4.1.3 there are two examples of how stemming works. As said before, the stem may not be a real word, as can be seen in the second example in the tab.

original words	**stemmed words**
connect, connects, connection, connected	connect
argue, argues, argued, arguing	argu

Table 4.1. Example of stemmed words

4.1.4 Lemmatization

It maps words into the actual root without just chopping off the ends of words: for example it transforms *is* and *are* in *be*. Therefore is similar to

stemming, but more complex. It needs to know the part of speech of the word to map to its lemma and require more knowledge about the structure of a language.

Considering the English language and the WordNet Lemmatizer, the table 4.2 shows some examples.

original words	**lemmatized words**
better	good
is, are, were	be

Table 4.2. Example of lemmatized words.

4.1.5 Stop-Word Removal

It's a technique that allows to focus on the most important words of a text since it consists in removing low information words. In fact, stop words are the most common terms used in a language. Examples of English stop words are *"the", "are", "a",* etc.

4.1.6 Noise Removal

It is about cleaning up the text from characters which can represent a problem for the text analysis. This includes numbers removal, html formatting removal, punctuation removal, special characters removal, etc. How the text is cleaned depends on the domain you are working in and what is considered "noise" for that specific task.

In all the extractive summarization methods proposed in this work, preprocessing is perfomed. In the source documents there are probably words like "the", "at", "of", etc., that are stop-words hat don't bring useful information, so there is the need to remove them to allow to focus on the most important terms of the text. Other words need to be removed and are those for which the model does not have the corresponding embedding. The text is lowercased and is also cleaned from special character which can represent a problem. Whenever possible, stemming was done. In particular for English, French and Arabic texts, the stemmer used is the Snowball Stemmer that is

available in nltk[1] package. It doesn't support Czech and Greek and no valid stemmers for these two languages has been found.

4.2 Model Alignment and Sentence Representation

Word embedding models have to be aligned before sentence representation. Alignment allows words of various languages to be comparable. Taking models of two different languages as input, a word in the language-1 and the corresponding word in language-2 need to be mapped to similar vectors. The method applied here is Procrustes and is explained in section 3.3.

Section 3.2 describes different techniques for sentence representation. It consists into combining embedding of words contained in a phrase to obtain the sentence vector.
Among them, the methods used in this work are:

- average of word embeddings
- weighted average with TF-IDF
- weighted average with IDF
- SIF embedding

4.2.1 Average of Word Embeddings

The sentence vector is obtained just averaging the embedding of the words belonging to the phrase itself, as shown in Algorithm 2.

[1]https://www.nltk.org/

Algorithm 2

Variables:
- v_w is the word embedding of word w
- S is a set of sentences
- n is the number of words in the sentence
- v_s is the resulting sentence embeddings

1: **for all** sentence s in S **do**
2: $v_s \leftarrow \frac{1}{n}\sum_{w \in s} v_w$
3: **end for**

4.2.2 Weighted Average with TF-IDF

First for all the words the TF-IDF value is calculated, then the vector of each word is multiplied by the corresponding TF-IDF score. Mathematically this score is the multiplication of the following two values:

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{|d_j|}$$

with $n_{i,j}$ the number of occurences of terms i in sentence j and with $|d_j|$ the number of words in it,

$$\mathrm{idf_i} = \log \frac{|D|}{|\{d : i \in d\}|}$$

where $|D|$ is the total number of articles and the denominator is the number of articles in which the terms i appears.

Thanks to this score, words have not all the same weight: more importance is given to less recurring words, and most frequent words become less relevant. Then they are all added up and divided by the number of words in the sentence, as shown in Algorithm 3.

Algorithm 3 TF-IDF Embedding

Variables

- v_w is the word embedding of word w
- S is a set of sentences
- n is the number of words in the sentence
- $tfidf_w$ is the tfidf score of word w
- v_s is the resulting sentence embeddings

1: **for all** sentence s in S **do**
2: $v_s \leftarrow \frac{1}{n}\sum_{w \in s} tfidf_w \times v_w$
3: **end for**

4.2.3 Weighted Average with IDF

Similar to TF-IDF embedding, but here the weighting function is the IDF. Algorithm 4 illustrates it.

Algorithm 4 IDF Embedding

Variables

- v_w is the word embedding of word w
- S is a set of sentences
- n is the number of words in the sentence
- idf_w is the idf score of word w
. v_s is the resulting sentence embeddings

1: **for all** sentence s in S **do**
2: $v_s \leftarrow \frac{1}{n}\sum_{w \in s} idf_w \times v_w$
3: **end for**

4.2.4 SIF embedding

It also makes a weighted average of word embeddings, but in addition to it, it subtracts from the sentence embeddings their projection on the first principal component (see Algorithm 5).

Algorithm 5 SIF Embedding

Variables
- v_w is the word embedding of word w
- S is a set of sentences
- a is a variable equals to 0.001
- $p(w)$ is the estimated frequency of w
- v_s is the resulting sentence embeddings

1: **for all** sentence s in S **do**
2: $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in S\}$, and let u be its first singular vector
5: **for all** sentence s in S **do**
6: $v_s \leftarrow v_s$ - $uu^T v_s$
7: **end for**

4.3 Sentence Selection

In this phase the sentences which results to be the most representative of the input documents are selected. The methods used are:

- PageRank [4]
- the centroid method proposed by Rossiello et al. in the paper "Centroid-based Text Summarization through Compositionality of Word Embeddings"[29]

4.3.1 PageRank Method

Once the sentence embeddings are obtained, it is necessary to calculate the similarity among phrases. This is done in two ways: with the cosine similarity and with the word mover's distance. The results are then stored in a matrix which is passed to PageRank, one of the algorithm Google uses to rank search results. In this case it is used to convert the similarity matrix into a graph: each vertices represents a sentence, each similarity score an edge. The output is a ranked list of sentences, from which the ones that will compose the summary are taken.

Since input documents are not monolingual, the top-n phrases of the ranked list obtained by PageRank can be composed of sentences in different languages and for this reason they must be filtered. The applied strategies are:

- A: select the top-n sentences in the target language, ignoring the ones in the other language

- B: from the top-n sentences, the ones in the target language are directly picked, while each one of the others is used to find what sentence in the target language is most similar to it. The similarity is derived using the cosine similarity or the word's mover distance, depending on which one was used in the previous step.

4.3.2 Centroid Method

It is an approach presented by Rossiello et al. [29]. After getting the TF-IDF score for all the words of the input texts, it selects the ones with a score above a threshold. Those words represents the centroid and their embeddings are combined with TF-IDF to get the centroid vector. The same embedding is done for all the sentences in the source documents. To select the sentences for the final summary, it calculates the cosine similarity between each of them and the centroid. The top-n sentences most similar to the centroid are the one choosen for the summary. Also in this case the selection was made in two ways, that is with method A and method B already described in Section 4.3.1.

The procedure is reported in Algorithm 6.

Algorithm 6 Centroid algorithm

Input: $S, Scores, st, limit$
Output: $Summary$

```
S ← SortDesc(S, Scores)
k ← 1
for i ← 1 to m do
    length ← Len(Summary)
    if length > limit then return Summary
    SV ←SumVectors(S[i])   include ← True
    for j ← 1tok do
        SV2 ← SumVectors(Summary[j])
        sim ← Similarity(SV, SV2)
        if sim > st then
            include ← False
    if include then
        Summary[k] ← S[i]
        k ← k + 1
```

Chapter 5

Experiments

5.1 Experimental Design

5.1.1 Dataset

The dataset used to test the proposed methods is MultiLing Pilot 2011 Dataset[1]. It contains 700 files, 100 for each of these languages: Arabic, Czech, English, French, Greek, Hebrew, Hindi. The 100 articles per language are further divided into 10 topics, each consisting of 10 texts. This dataset was created taking English texts from WikiNews[2]. Subsequently, native speakers translated them in order to create articles in the other 6 languages. To allow the evaluation of the generated summaries, are also provided from 2 to 5 human summaries for every topic and every language. For each topic, the gold summaries have been made by native speakers, after they read all the articles concerning that topic.

5.1.2 Word Embedding Models

All the methods are performed once with Word2Vec models and once with FastText[3] models. In both cases the vectors have been trained on Wikipedia and their dimension is 300. As regards Word2Vec, the alignment is performed through the Procrustes method [32], while FastText models have been aligned using the method proposed in Joulin et al. (2018)[16].

[1] http://www.nist.gov/tac/2011/Summarization/

[2] http://www.wikinews.org/

5.1.3 Experiments

Among the documents available in MultiLing Pilot 2011 Dataset, the ones in English, French, Arabic, Czech and Hindi are taken into account. For all the methods mentioned in Chapter 4, the following tests have been done:

- English + French to English
- English + French to French
- English + Arabic to English
- English + Arabic to Arabic
- English + Czech to English
- English + Czech to Czech
- English + Hindi to English
- English + Hindi to Hindi

Several tests have been done by considering as input different numbers of articles for each language. These are the combination tested:

- 10 articles for each of the source languages. It is used to evaluate the impact of the presence of the same articles in another language.
- 5 articles for each of the source languages about the same topic. This is the case that most represents a real situation. In fact, thinking about news that can be found in online newspapers which relate to the same fact or event, they can be written in different languages, but, usually, they are not a literal translation of the other.
- 10 articles in the source language (equals to the target one) plus those obtained by translating them with Google Translate into the other source language. In another case, starting from the 10 articles in the source language (equals to the target one), 5 of them are used in the original language, the other 5 are translated into the other source language with Google Translate. They are both used to evaluate the impact of using machine translation to generate documents in other languages for summary extraction.

- 10 or 5 articles only in the target language. They are useful to understand if the previous cases actually allow to obtain better results, considering not only articles in the target language.

Every summarization method selects a number of sentences necessary to reach the 250 words limit.

The performances of the proposed techniques are evaluated and compared with the following baselines:

- TextRank [24]
- LexRank [7]
- CoReRank [6]

TextRank and LexRank summaries were obtained thanks to the implementation of these methods provided in the library *sumy 0.8.1* [34]. Instead, for CoReRank, the code used is the one provided by Chetan et al. on GitHub [6].

5.1.4 Algorithm execution time

The execution time, in addition to depending on the machine used for the experiments, mainly depends on the distance measurement computed and the number of words present in the embedding models used. For the first reason, using word mover's distance instead of cosine similarity leads to a longer run time (it can also take up to 20 minutes, 10 in the other case). Instead, for the second reason, the less rich the embedding model, the less the number of computations that the algorithm must perform (for example if one of the two languages is Hindi, the algorithm is faster). Experiments have been done on a PC with Intel Core i7-6700HQ 2.60GHz CPU and 16GB memory.

5.2 Evaluation

Summaries evaluations are obtained through the ROUGE 2.0 toolkit[9]. It allows to measure the quality of a summary by comparing it with the golden summary. Since in all the cases the reference summaries provided for each topic in the dataset used are more than one, the tool compares the generated summary with each of the reference ones, and the result it provides is given

by the average of the values obtained from the individual comparisons. All system summaries have been truncated to 250 words before carrying out the evaluation.

5.2.1 ROUGE

ROUGE[21][30] stands for Recall-Oriented Understudy for Gisting Evaluation. It is a software for the evaluation of automatic summarization. It compares the generated summary with one or more reference summaries, which generally are human produced, to determine how good it is. At the beginning it was only recall oriented. Recall allows you to obtain how much the system summary is "capturing" of the reference summary. In case the single words are considered, recall is obtained by:

$$\frac{n_{overlapping}}{N_{reference}}$$

where $n_{overlapping}$ is the number of overlapping words and $N_{reference}$ is the total number of words in the reference summary. The problem with this measure is that a summary, if very long, could capture all the words in the reference summary, obtaining a recall value of 1, but resulting verbose. To compensate for this problem, precision has been introduced. It can be computed as:

$$\frac{n_{overlapping}}{N_{system}}$$

where $n_{overlapping}$ is the number of overlapping words and N_{system} is the total number of words in the system summary. It allows to measure how much of the system summary is actually relevant or necessary.

In general is better to compute both the precision and recall and obtain the F-Score from them. It is the harmonic mean of precision and recall:

$$Fscore = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

In certain situations, precision may be of little relevance and therefore may be omitted, whereas only recall may be considered. Generally these are the cases in which summaries are forced to be concise by imposing certain constraints.

The ROUGE metrics available are:

- ROUGE N: it is based on n-grams overlap. ROUGE-1 refers to the overlap of unigrams between the system summary and the reference one. Instead, ROUGE-2 is based on the overlap of bigrams.

- ROUGE L (Longest Common Subsequence): measures the longest matching sequence of words using LCS.
- ROUGE S: measures the skip-bigram co-occurrence. A skip-bigram is a pair of words present within a sentence reported in their original order, regardless of the terms between them.
- ROUGE SU: is an extension of ROUGE S. It is based on skip-bigram and unigram-based co-occurrence.

5.3 Qualitative Analysis

Figure 5.1 and 5.2 respectively show an example of an English article and the corresponding French article, while in Figure 5.3 there is an example of one of the French human summary provided for the same topic of the articles mentioned above. All of them are taken from the Multiling Pilot 2011 Dataset.

In the figures 5.4, 5.5 and 5.6 there are three examples of system French summaries. Specifically, they have been obtained through the WMD method, using as input respectively:

- 10 French articles from MultiLing Pilot 2011 Dataset of which 5 used in the original language and 5 translated into English with Google Translator
- 5 French and 5 English articles from MultiLing Pilot 2011 Dataset
- 5 French documents from MultiLing Pilot 2011 Dataset

2005/01/08 Tsunami aid donations in 2005 deductible for 2004 in the U.S.

Saturday, January 8, 2005

U.S. citizens donating in 2005 to help tsunami victims may write off their donations on their 2004 tax returns, thanks to a bill quickly passed in the U.S. House of Representatives and the U.S. Senate on a voice vote, and signed into law by president George W. Bush.

Without the new law, contributors would have waited until 2006 and their 2005 tax returns to be able to write off their charitable donations. The law is intended to promote donating towards the tsunami relief effort.

CBS News reports Indiana University's Center on Philanthropy is estimating approximately 322 million U.S. dollars in goods and cash have been donated by private U.S. citizens and corporations, in addition to the 350 million that was promised by the government.

An AP/ISOS poll has found three in ten U.S. citizens have donated to Tsunami Aid organizations.

Figure 5.1. Example of an English article from MultiLing Pilot 2011 Dataset.

08/01/2005 Les dons pour le tsunami en 2005 déductibles des impôts 2004 aux USA

Samedi 8 janvier 2005

Les citoyens américains qui ont fait des dons en 2005 pour aider les victimes du tsunami peuvent reporter leurs dons sur leur déclaration d'impôts 2004, grâce à une loi passée rapidement à la Chambre des représentants et au Sénat américains par vote par acclamation, puis approuvée officiellement par le président George W. Bush.

Sans la nouvelle loi, les contributeurs auraient dû attendre 2006 et leur déclaration d'impôts 2005 pour pouvoir y reporter leur geste charitable. La loi a pour but de promouvoir les dons en faveur de l'aide aux victimes du tsunami.

CBS News rapporte que le centre sur la philanthropie de l'université d'Indiana estime qu'approximativement 322 millions de dollars en biens et argent ont été donnés par des citoyens privés et des entreprises, en plus des 350 millions qui avaient été promis par le gouvernement.

Un sondage AP/ISOS a montré que trois Américains sur dix ont effectué des dons à des organisations d'aide aux victimes du tsunami.

Figure 5.2. Example of a French article from MultiLing Pilot 2011 Dataset.

As can be seen from Figure 5.2 and 5.1 the articles in the various languages are translated from the English one in such a way as to maintain the general structure and so that each sentence and the corresponding one in another language express the same concept.

In general, the phrases of the summaries are not necessarily connected

Le 26 décembre 2004 s'est produit à Sumatra un séisme très important de magnitude 9,1-9,3 sur l'échelle de Richter. Ce tremblent de terre, le plus long jamais enregistré, a provoqué un tsunami qui a frappé de nombreux pays d'Asie et d'Afrique de l'Est. Près du tsunami, la province d'Aceh en Indonésie pourrait à elle seule compter au moins 80 000 morts. Des villages entiers ont disparu, ne laissant aucune trace de leur existence. On estime que le tsunami a fait plus de 120 000 morts, dont 28 500 au Sri Lanka et plus de 7 700 en Inde. Plus de 10 500 étrangers, des touristes pour la plupart, sont portés disparus dans la région. Jusqu'à quatre fois plus de femmes que d'hommes sont morts dans le tsunami. Dans certains villages, il apparaît maintenant que jusqu'à 80% des personnes tuées étaient des femmes. Les décès résultant directement du séisme et des tsunamis ne sont qu'une fraction de l'effet total de la catastrophe. L'Organisation Mondiale de la Santé (OMS) estime que 5 millions de personnes manquent de nourriture, d'eau ou d'assainissement de base. Après des débuts timides, l'argent récolté auprès des nations les plus riches s'élèvent en date du 1er janvier à environ 2 milliards de dollars américains. Outre les aides gouvernementales, la population et des entreprises ont donné massivement des dons. Malgré les promesses encourageantes, un responsable des Nations Unies a averti que la logistique pour obtenir les fonds, acheter les vivres et les acheminer aux régions touchées prendrait du temps.

Figure 5.3. Example of a French human summary from MultiLing Pilot 2011 Dataset.

Bilans des victimes locaux. D'autres pays continuent néanmoins de communiquer au mieux le nombre de morts : Sri Lanka - environ 28 500, Inde - plus de 7 700. Plus de 300 personnes ont été tuées en Malaisie, au Myanmar, au Bangladesh, aux Maldives, en Somalie, en Tanzanie et au Kenya. Outre le fait que les rapports des travailleurs sur le terrain ont été « de plus en plus conscients qu'un pourcentage disproportionné des décès étaient des femmes », l'étude examine en détail un certain nombre d'endroits en Indonésie, au Sri Lanka et en Inde, les trois pays avec la plupart des morts, et trouve toujours que les femmes tuées sont toujours plus nombreuses que les hommes. Jusqu'à quatre fois plus de femmes que d'hommes sont morts dans le tsunami de l'océan Indien du 26 décembre, des chiffres publiés aujourd'hui par Oxfam International. Environ un million de personnes sont sans abri et les organisations humanitaires évaluent à 5 millions le nombre de personnes ayant besoin de secours. Près d'une semaine après le tsunami qui a frappé les côtes de plusieurs pays d'Asie du Sud, le bilan officiel s'élève à plus de 120 000 morts. La Chine a promis 60,5 millions, le don national le plus important après le Japon, les États-Unis, le Royaume-Uni et la Suède.

Figure 5.4. Example of a French summary obtained with the WMD method taking 10 French documents from MultiLing Pilot 2011 Dataset: 5 are used in original language, the other 5 are translated in English with Google Translate.

Des douzaines d'organisations humanitaires internationales ont accepté les dons pour aider les victimes du tremblement de terre et du tsunami. Grâce à un téléthon, les Roumains ont recueilli 15 milliards de lei, soit l'équivalent de 395 000 euros, pour les victimes du tsunami dévastateur dans l'océan Indien, du lendemain de Noël 2004, qui a fait plus de 175 000 victimes. Lors d'un volte-face inattendu, les nations les plus riches du monde ont commencé à arroser d'argent les régions touchées par le tremblement de terre et le tsunami. La Chine a promis 60,5 millions, le don national le plus important après le Japon, les États-Unis, le Royaume-Uni et la Suède. Et cette puissance a duré plus longtemps que tous les tremblements de terre enregistrés jusqu'à maintenant. Les 395 000 euros collectés par le téléthon complètent les 150 000 euros d'aide promis par le gouvernement roumain sous la forme de tentes, d'eau et de médicaments dans les zones frappées par le tsunami d'Asie du Sud. Les scientifiques ont revu à la hausse la puissance du tremblement de terre, de 9,0 à 9,1-9,3 sur l'échelle de Richter, ce qui représente un tremblement de terre bien plus important. Les fonds promis ont doublé en 24 heures, et s'élèvent maintenant à environ 2 milliards de dollars américains (USD).

Figure 5.5. Example of a French summary obtained with the WMD method taking as input 5 French documents and 5 English documents from MultiLing Pilot 2011 Dataset.

Des douzaines d'organisations humanitaires internationales ont accepté les dons pour aider les victimes du tremblement de terre et du tsunami. Lors d'un volte-face inattendu, les nations les plus riches du monde ont commencé à arroser d'argent les régions touchées par le tremblement de terre et le tsunami. Et cette puissance a duré plus longtemps que tous les tremblements de terre enregistrés jusqu'à maintenant. La Chine a promis 60,5 millions, le don national le plus important après le Japon, les États-Unis, le Royaume-Uni et la Suède. Les scientifiques ont revu à la hausse la puissance du tremblement de terre, de 9,0 à 9,1-9,3 sur l'échelle de Richter, ce qui représente un tremblement de terre bien plus important. Grâce à un téléthon, les Roumains ont recueilli 15 milliards de lei, soit l'équivalent de 395 000 euros, pour les victimes du tsunami dévastateur dans l'océan Indien, du lendemain de Noël 2004, qui a fait plus de 175 000 victimes. Les promesses d'aide augmentent; le Japon promet 500 millions de dollars. Les fonds promis ont doublé en 24 heures, et s'élèvent maintenant à environ 2 milliards de dollars américains (USD). Il y aura d'autres tremblements de terre de ce type, et avec plus de vies exposées au danger, les conséquences en seront d'autant plus dévastatrices. Les 395 000 euros collectés par le téléthon complètent les 150 000 euros d'aide promis par le gouvernement roumain sous la forme de tentes, d'eau et de médicaments dans les zones frappées par le tsunami d'Asie du Sud.

Figure 5.6. Example of a French summary obtained with the WMD method taking as input only 5 French documents from MultiLing Pilot 2011 Dataset.

to each other. In fact, from the linguistic quality point of view, the system summaries are not comparable to that obtained by a human, given that no checks of this type are carried out and the summarization technique used is extractive. This can be seen in the summary in the Figure 5.4, where the initial sentence does not make much sense. Each method allows to obtain different summaries. They may have sentences in common, but are usually positioned differently. For example, the sentence "La Chine a promis 60,5 millions, le don national..." is selected from both 5+5* summary and 5+0 summary, but while in the first summary the phrase is in last position, in the other summary it has been found to be more important and is therefore in 4th position. This implies that adding English information to French ones, influences the importance that is associated to each sentence.
All the summaries are less than 250 words long to be evaluated.

5.4 Quantitative Evaluation

The following part shows all the ROUGE-2 Recall scores obtained for the various tests performed. In order to understand the tables, here is the list of the abbreviations used:

- A: as well as B, it refers to the sentence selection method used. In this case it selects the top-n sentences in the target language, ignoring the ones in the other language

- B: from the top-n sentences, the ones in the target language are directly picked, while each one of the others is used to find what sentence in the target language is the closest to it. The similarity is derived using the cosine similarity or the word's mover distance, depending on which one was used in the previous step

- 10+10: 10 articles for each of the source languages are taken as input

- 10+10*: the inputs are 10 articles in the source language (equals to the target one) plus those obtained by translating them with Google Translate into the other source language

- 10+0: it takes as input only the 10 articles of the target language

- 5+5: 5 articles for each of the source languages are taken as input

- 5+5*: starting from the 10 articles in the source language (equals to the target one), 5 of them are used in the original language, the other 5 are translated into the other source language with Google Translate
- 5+0: it takes as input only the 5 articles of the target language

Moreover the title of each table, e.g. "EN + FR to EN", refers to the languages use as input and as output. For example in the case "EN + FR to EN" it means that an EN summary is obtained by taking EN and FR documents as inputs.

In some tables, in addition to the results obtained by the various methods proposed, there are also the scores getted by the baselines: CoReRank is available only for the English language, while TextRank and LexRank could not be calculated for Arabic and Hindi.

Before the detailed results tables, the ones with the Borda Count [36] reranking results are shown. It is a consensus function that allows to determine the overall summarizer ranking from a set of results on each individual language. The one with the best score wins. Thanks to these results it was possible to draw conclusions on the method in general rather than on a specific language.
The first part of the summarizer name indicates the method used, the second part the number of articles for language, the last part the selection method.

Word2Vec

Summarizer	**Borda Count Ranking**
WMD_5+5*_C	350
WMD_5+5_F	323
WMD_5+5_C	310
WMD_5+5*_F	286
WMD_10+0_C	282
WMD_5+5_F	282
MEDIA_5+5_F	280
WMD_10+0_F	277
TFIDF_5+5_F	276
MEDIA_5+5*_F	265

Table 5.1. Borda Count Rankings achieved by the methods over all the tested languages. The table shows the top-10 summarizers ranked by ROUGE-2 Recall.

FastText

Summarizer	Borda Count Ranking
WMD_10+0_F	330
WMD_10+0_C	327
TFIDF_5+5*_C	287
WMD_5+5_C	283
WMD_5+5*_F	280
WMD_5+5_F	273
TFIDF_5+5_F	266
WMD_5+5*_C	264
IDF_5+5_C	262
TFIDF_5+5_C	256

Table 5.2. Borda Count Rankings achieved by the methods over all the tested languages. The table shows the top-10 summarizers ranked by ROUGE-2 Recall.

The method that is most able to take advantage of aligned embedding is the TextRank with distance measurement given by the WMD (in that case the best configuration seems to be 5+5*). Furthermore, from the two tables above result that exploiting the FastText embedding models leads to lower results than those of Word2Vec.

The results are detailed in the following part. Firstly all the results obtained with Word2Vec are shown, followed by those of FastText.

Considering the summaries obtained in English in the case *10+10*, if the articles in French or Arabic are added to the English input ones, the final results are better than a summary obtained through the use of Google Translator or that obtained with baselines. Even the articles in Czech lead to better summaries than those obtained considering only the English articles, although in this case the method that uses the translation tool prevails.

For the *5+5* case, English with Arabic and English with Czech have better performances than *5+5** or *5+0* cases.

As regards the summaries extracted in the French language, there is an increase in performance thanks to the informations brought by the 10 additional English articles. The same thing can be observed from the test carried out with 5 articles per language.

For the Arabic summaries, on the other hand, the scores do not allow to

clearly understand if the use of two languages is effective: in the *5+5* test it would seem so, but in the *10+10* one, a better summary is obtained by using only the Arabic articles.

In extracting summaries in the Hindi language, it was demonstrated that using documents in multiple languages allows for better summaries.

However, this does not apply to Czech summaries where LexRank performs much better. It is the only case where adding information in English results in significantly worse results.

Comparing the methods that exploit the cosine similarity and pagerank, but each with different techniques of sentence embeddings, it is clear that the worst is the one with SIF. This type of embedding is mainly designed for textual similarity tasks, and this is perhaps the reason why it does not perform well here. Instead, the methods with IDF and TFIDF are better, even if they do not beat the one in which a simple average of the embeddings of the words in a sentence is used.

However, in most cases, the method that performs better is the one with the word mover's distance instead of the cosine similarity. Even the centroid method, sometimes, leads to the best results. A particular case is the one which deals with summaries in Hindi, where the method that uses pagerank and gets sentence embeddings by averaging the embeddings of the words in it, outperforms the others.

As regards the method of selecting the sentences, on the other hand, method A seems to perform better in several cases, but the distinction is not clear-cut, therefore it is probably better to use the A selection mode, but also mode B is not to be excluded a priori.

Unfortunately with FastText models the results do not correspond to those obtained using Word2Vec embeddings. Here integrating the articles in one language with those of another language does not allow to give an added value for the extraction of a better summary. From these scores, however, is confirmed the greater efficiency of the method that exploits the word mover's distance compared to the others.

EN + FR to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,116513	0,118463	0,098611	0,123188	0,120768
idf+pagerank	0,096041	0,104019	0,09891	0,120989	0,092883
tfidf+pagerank	0,100519	0,10953	0,09026	0,099546	0,112969
sif+pagerank	0,08664	0,091593	0,068851	0,091163	0,089056
wmd+pagerank	**0,155752**	0,151951	0,12851	0,142367	0,148278
centroid	0,123309	0,122952	0,124941	0,121169	0,125486
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	0,14719

Table 5.3. ROUGE-2 Recall scores obtained by extracting an English summary getting English and French documents as input. Word embedding models used: Word2Vec.

EN + FR to EN

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,130172	0,120018	0,120832	0,106209	0,131927
idf+pagerank	0,119807	0,109688	0,112724	0,105272	0,117253
tfidf+pagerank	0,126405	0,108307	0,108837	0,111867	0,120823
sif+pagerank	0,093917	0,086835	0,111173	0,084456	0,091694
wmd+pagerank	0,128165	0,132268	0,132923	**0,138499**	0,132236
centroid	0,118531	0,115168	0,129424	0,117308	0,116011
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.4. ROUGE-2 Recall scores obtained by extracting an English summary using English and French documents as input. Word embedding models used: Word2Vec.

Tables 5.3 and 5.4 show that adding articles in French and using the corresponding word embedding model, can lead to better results than simply considering English articles. However, if 5 articles per language are considered, the method that uses google translator performs better, in particular the version with WMD.

EN + FR to FR

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,095498	0,097488	0,099705	0,110028	0,092849
idf+pagerank	0,073528	0,084989	0,081208	0,096475	0,070444
tfidf+pagerank	0,077041	0,087501	0,081506	0,100309	0,07245
sif+pagerank	0,065161	0,074148	0,063443	0,088402	0,068523
wmd+pagerank	0,141332	0,13101	**0,147918**	0,12492	0,138382
centroid	0,122943	0,114663	0,12387	0,11814	0,126791
textrank	-	-	-	-	0,09588
lexrank	-	-	-	-	0,13875

Table 5.5. ROUGE-2 Recall scores obtained by extracting a French summary using English and French documents as input. Word embedding models used: Word2Vec.

EN + FR to FR

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,093921	0,091683	0,093829	0,089834	0,093937
idf+pagerank	0,083171	0,084381	0,082709	0,099164	0,08538
tfidf+pagerank	0,081331	0,084407	0,085705	0,092522	0,083833
sif+pagerank	0,055688	0,074112	0,055925	0,072805	0,051733
wmd+pagerank	0,124705	0,126047	0,130118	**0,144296**	0,131468
centroid	0,110428	0,109414	0,114994	0,108785	0,102347
textrank	-	-	-	-	0,09794
lexrank	-	-	-	-	0,12775

Table 5.6. ROUGE-2 Recall scores obtained by extracting a French summary using English and French documents as input. Word embedding models used: Word2Vec.

Table 5.5 and 5.6 show that English articles help to get a better Frech summary. In particular, in the case 10+10, the method proposed that exploits wmd measure increases a lot the quality of summaries.

EN + AR to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,092902	0,115227	0,114683	0,091919	0,120768
idf+pagerank	0,083555	0,106853	0,099724	0,095711	0,092883
tfidf+pagerank	0,079633	0,092404	0,112092	0,080413	0,112969
sif+pagerank	0,073001	0,082652	0,090212	0,074136	0,089056
wmd+pagerank	**0,147401**	0,122687	0,121217	0,145217	0,14600
centroid	0,142926	0,117442	0,137224	0,133093	0,125486
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	0,14719

Table 5.7. ROUGE-2 Recall scores obtained by extracting an English summary using English and Arabic documents as input. Word embedding models used: Word2Vec.

EN + AR to EN

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,111905	0,091547	0,097698	0,107238	0,131927
idf+pagerank	0,107797	0,085181	0,107918	0,097252	0,117253
tfidf+pagerank	0,098495	0,089127	0,09767	0,106086	0,120823
sif+pagerank	0,091991	0,08155	0,102482	0,088464	0,091694
wmd+pagerank	0,133062	0,130569	0,120848	0,122709	0,132236
centroid	**0,140918**	0,133311	0,135556	0,123922	0,116011
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.8. ROUGE-2 Recall scores obtained by extracting an English summary using English and Arabic documents as input. Word embedding models used: Word2Vec.

From tables 5.7 and 5.8 is clear that better results can be reached by considering Arabic news in addition to the English ones. While in the 10+10 case wmd is confirmed as the best method, in 5+5 the centroid wins.

EN + AR to AR

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,110184	0,111302	0,11104	0,108466	0,100758
idf+pagerank	0,102479	0,096197	0,09741	0,092563	0,080633
tfidf+pagerank	0,095962	0,094975	0,098261	0,091154	0,078281
sif+pagerank	0,058988	0,063524	0,052724	0,056991	0,061673
wmd+pagerank	0,158362	0,152908	0,151712	0,158587	**0,179391**
centroid	0,154169	0,138189	0,123103	0,108849	0,122067

Table 5.9. ROUGE-2 Recall scores obtained by extracting an Arabic summary using English and Arabic documents as input. Word embedding models used: Word2Vec.

EN + AR to AR

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,111611	0,105476	0,107368	0,099358	0,110573
idf+pagerank	0,101068	0,086747	0,109585	0,08975	0,098402
tfidf+pagerank	0,104699	0,102286	0,110572	0,093208	0,09845
sif+pagerank	0,066785	0,058576	0,065259	0,05899	0,075178
wmd+pagerank	0,136018	**0,142624**	0,138013	0,132028	0,132323
centroid	0,128536	0,104724	0,121322	0,115061	0,122875

Table 5.10. ROUGE-2 Recall scores obtained by extracting an Arabic summary using English and Arabic documents as input. Word embedding models used: Word2Vec.

In the case of 10+10 of EN+AR to AR, adding information in English greatly worsens the results. However, it is not confirmed in case of 5 articles for language. In both cases it is the wmd method that performs better.

EN + CS to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,096744	0,10503	0,096149	0,091567	0,120768
idf+pagerank	0,087936	0,092325	0,095179	0,084747	0,092883
tfidf+pagerank	0,087782	0,090128	0,105875	0,094059	0,112969
sif+pagerank	0,092353	0,086495	0,083606	0,092458	0,089056
wmd+pagerank	0,141982	**0,150043**	0,120426	0,132697	0,148278
centroid	0,121616	0,132792	0,137611	0,134016	0,125486
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	0,14719

Table 5.11. ROUGE-2 Recall scores obtained by extracting an English summary using English and Czech documents as input. Word embedding models used: Word2Vec.

EN + CS to EN

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,107934	0,091085	0,112292	0,105231	0,131927
idf+pagerank	0,094849	0,088809	0,1138	0,10497	0,117253
tfidf+pagerank	0,103166	0,0889	0,10307	0,102046	0,120823
sif+pagerank	0,095447	0,103337	0,102842	0,084687	0,091694
wmd+pagerank	**0,139877**	0,139658	0,110038	0,138957	0,132236
centroid	0,111763	0,121388	0,122902	0,115683	0,116011
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.12. ROUGE-2 Recall scores obtained by extracting an English summary using English and Czech documents as input. Word embedding models used: Word2Vec.

In the case EN+CS to EN, thanks to the embedding models, better results are obtained. Moreover, the articles in Czech provide additional information that allow to produce better summaries than those obtained only from English.

EN + CS to CS

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,10909	0,09331	0,090389	0,098767	0,10033
idf+pagerank	0,091726	0,085894	0,088636	0,082414	0,089913
tfidf+pagerank	0,088576	0,079843	0,089503	0,087111	0,08371
sif+pagerank	0,06397	0,077927	0,074062	0,077138	0,064793
wmd+pagerank	0,107977	0,10284	0,101398	0,100399	0,104307
centroid	0,107054	0,090483	**0,114633**	0,08389	0,091564
textrank	-	-	-	-	0,14924
lexrank	-	-	-	-	**0,19208**

Table 5.13. ROUGE-2 Recall scores obtained by extracting a Czech summary using English and Czech documents as input. Word embedding models used: Word2Vec.

EN + CS to CS

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,10333	0,091091	0,100951	0,099279	0,112422
idf+pagerank	0,100583	0,085597	0,104694	0,088311	0,095987
tfidf+pagerank	0,096563	0,088466	0,099238	0,08733	0,101357
sif+pagerank	0,075206	0,055699	0,083754	0,06284	0,08436
wmd+pagerank	0,108138	0,096104	0,110122	0,091634	**0,115223**
centroid	0,094383	0,099053	0,105533	0,091358	0,108918
textrank	-	-	-	-	0,09978
lexrank	-	-	-	-	**0,13514**

Table 5.14. ROUGE-2 Recall scores obtained by extracting a Czech summary using English and Czech documents as input. Word embedding models used: Word2Vec.

Tables 5.29 and 5.14 show that, in the extraction of Czech summaries, baselines produce best summaries, while the other methods obtain significantly lower results.

EN + HI to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,10302	0,110889	0,117444	0,115169	0,120768
idf+pagerank	0,099094	0,102126	0,112838	0,108876	0,092883
tfidf+pagerank	0,09692	0,101481	0,116567	0,102909	0,112969
sif+pagerank	0,074525	0,086604	0,099957	0,084674	0,089056
wmd+pagerank	0,140313	0,147091	0,130765	0,145502	**0,148278**
centroid	0,083876	0,069821	0,12327	0,100043	0,125486
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	0,14719

Table 5.15. ROUGE-2 Recall scores obtained by extracting an English summary using English and Hindi documents as input. Word embedding models used: Word2Vec.

EN + HI to EN

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,110631	0,114269	0,122128	0,113214	0,131927
idf+pagerank	0,10123	0,110083	0,133473	0,108949	0,117253
tfidf+pagerank	0,109007	0,107487	0,121553	0,105089	0,120823
sif+pagerank	0,078595	0,085626	0,105667	0,085577	0,091694
wmd+pagerank	0,129457	**0,141906**	0,126869	0,141906	0,132236
centroid	0,093571	0,08232	0,116904	0,096209	0,116011
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.16. ROUGE-2 Recall scores obtained by extracting an English summary using English and Hindi documents as input. Word embedding models used: Word2Vec.

Also in these cases, the technique with the word mover's distance wins. Embeddings can improve performance, while adding Hindi documents allows good results in the case of 5 articles per language, but when 10 articles are considered, it does not bring benefits.

EN + HI to HI

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,058857	0,046189	0,041928	**0,060396**	0,036147
idf+pagerank	0,044152	0,04762	0,048709	0,052986	0,040336
tfidf+pagerank	0,057053	0,043869	0,040618	0,054594	0,038875
sif+pagerank	0,05523	0,056974	0,05523	0,056894	0,036176
wmd+pagerank	0,056093	0,055293	0,049599	0,049656	0,044968
centroid	0,054629	0,059198	0,054629	0,059198	0,050903

Table 5.17. ROUGE-2 Recall scores obtained by extracting an Hindi summary using English and Hindi documents as input. Word embedding models used: Word2Vec.

EN + HI to HI

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	**0,064256**	0,051553	0,056812	0,050429	0,042876
idf+pagerank	0,057967	0,048659	0,059287	0,043245	0,058544
tfidf+pagerank	0,058024	0,035854	0,0564	0,037948	0,059206
sif+pagerank	0,047925	0,042168	0,044825	0,046637	0,0619
wmd+pagerank	0,04809	0,055159	0,058635	0,057868	0,049572
centroid	0,046797	0,046027	0,046797	0,046027	0,053198

Table 5.18. ROUGE-2 Recall scores obtained by extracting an Hindi summary using English and Hindi documents as input. Word embedding models used: Word2Vec.

Adding English to Hindi helps to improve the summary. The 5+5 case is the only one in which pagerank with average performs better than the other methods.

EN + FR to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,089136	0,08379	0,100966	0,086102	0,092718
idf+pagerank	0,089647	0,088538	0,094643	0,082549	0,091233
tfidf+pagerank	0,0945	0,088827	0,098178	0,092235	0,097686
sif+pagerank	0,089436	0,069997	0,085083	0,080029	0,059421
wmd+pagerank	0,105159	0,112025	0,098022	0,113384	**0,140637**
centroid	0,125859	0,116443	0,136414	0,121179	0,122069
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	**0,14719**

Table 5.19. ROUGE-2 Recall scores obtained by extracting an English summary using English and French documents as input. Word embedding models used: FastText.

EN + FR to FR

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,090507	0,083312	0,103974	0,096107	0,112147
idf+pagerank	0,092158	0,087573	0,095889	0,087997	0,10211
tfidf+pagerank	0,084396	0,076645	0,0865	0,09039	0,113176
sif+pagerank	0,102564	0,081014	0,082877	0,094701	0,091722
wmd+pagerank	0,106972	0,105611	0,096536	0,107237	**0,127931**
centroid	0,12129	0,102653	0,124104	0,095715	0,119833
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.20. ROUGE-2 Recall scores obtained by extracting an English summary using English and French documents as input. Word embedding models used: FastText.

FastText embeddings don't help getting good results with the method proposed and this applies to all the combinations of languages considered. In the cases in which 10 articles for language are taken as input, always win the baselines, while in the case of 5 articles, it is almost always the 5+0 method that wins.

EN + FR to FR

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,097441	0,092569	0,086798	0,104784	0,077561
idf+pagerank	0,083627	0,091398	0,060866	0,068382	0,052658
tfidf+pagerank	0,085643	0,091737	0,075026	0,07563	0,062048
sif+pagerank	0,060407	0,077212	0,056184	0,058662	0,059326
wmd+pagerank	0,103659	0,100348	0,102647	0,095632	**0,137392**
centroid	0,10643	0,104367	0,10643	0,104367	0,11061
textrank	-	-	-	-	0,09588
lexrank	-	-	-	-	**0,13875**

Table 5.21. ROUGE-2 Recall scores obtained by extracting a French summary using English and French documents as input. Word embedding models used: FastText.

EN + FR to FR

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,109956	0,109127	0,118969	0,114967	0,082638
idf+pagerank	0,073953	0,081127	0,06875	0,089044	0,067185
tfidf+pagerank	0,081042	0,085444	0,098046	0,093405	0,081156
sif+pagerank	0,061222	0,080889	0,060644	0,071091	0,052059
wmd+pagerank	0,107238	0,10557	0,10298	0,099869	**0,129083**
centroid	0,103835	0,116311	0,103835	0,116311	0,110853
textrank	-	-	-	-	0,09794
lexrank	-	-	-	-	0,12775

Table 5.22. ROUGE-2 Recall scores obtained by extracting a French summary using English and French documents as input. Word embedding models used: FastText.

EN + AR to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,08741	0,08379	0,089476	0,086102	0,092718
idf+pagerank	0,091709	0,088538	0,085972	0,082549	0,091233
tfidf+pagerank	0,08869	0,088827	0,091466	0,092235	0,097686
sif+pagerank	0,081831	0,069997	0,064465	0,080029	0,059421
wmd+pagerank	0,100912	0,112025	0,101123	0,113384	**0,140637**
centroid	0,122158	0,116443	0,122081	0,121179	0,122069
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	**0,14719**

Table 5.23. ROUGE-2 Recall scores obtained by extracting an English summary using English and Arabic documents as input. Word embedding models used: FastText.

EN + AR to EN

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,082191	0,083312	0,086101	0,096107	0,112147
idf+pagerank	0,088082	0,087573	0,084371	0,087997	0,10211
tfidf+pagerank	0,075976	0,076645	0,083511	0,09039	0,113176
sif+pagerank	0,084291	0,081014	0,08143	0,094701	0,091722
wmd+pagerank	0,102028	0,105611	0,10317	0,107237	**0,127931**
centroid	0,098711	0,102653	0,107276	0,095715	0,119833
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.24. ROUGE-2 Recall scores obtained by extracting an English summary using English and Arabic documents as input. Word embedding models used: FastText.

EN + AR to AR

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,088405	0,085955	0,094151	0,115124	0,078567
idf+pagerank	0,074462	0,070649	0,088088	0,080182	0,06727
tfidf+pagerank	0,088922	0,095736	0,090263	0,077976	0,061239
sif+pagerank	0,057549	0,053324	0,057382	0,049487	0,04106
wmd+pagerank	0,141535	0,141852	0,110613	0,098238	**0,146912**
centroid	0,12041	0,120899	0,110945	0,099676	0,117153

Table 5.25. ROUGE-2 Recall scores obtained by extracting an Arabic summary using English and Arabic documents as input. Word embedding models used: FastText.

EN + AR to AR

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,095139	0,096941	0,088683	0,112057	0,086107
idf+pagerank	0,073022	0,075119	0,084246	0,076509	0,067417
tfidf+pagerank	0,082642	0,085345	0,085689	0,095715	0,071831
sif+pagerank	0,065543	0,065866	0,054222	0,064601	0,052384
wmd+pagerank	0,118942	0,115207	0,119376	0,120114	**0,128778**
centroid	0,099683	0,086928	0,099319	0,081772	0,107175

Table 5.26. ROUGE-2 Recall scores obtained by extracting an Arabic summary using English and Arabic documents as input. Word embedding models used: FastText.

EN + CS to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,092571	0,093029	0,093526	0,080445	0,092718
idf+pagerank	0,083108	0,087171	0,083108	0,077339	0,091233
tfidf+pagerank	0,093139	0,080907	0,093139	0,091172	0,097686
sif+pagerank	0,079093	0,096281	0,079093	0,09856	0,059421
wmd+pagerank	0,111311	0,113276	0,105549	0,11513	**0,140637**
centroid	0,094285	0,12155	0,092572	0,120827	0,122069
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	**0,14719**

Table 5.27. ROUGE-2 Recall scores obtained by extracting an English summary using English and Czech documents as input. Word embedding models used: FastText.

EN + CS to EN

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,084451	0,084689	0,086682	0,086667	0,112147
idf+pagerank	0,088783	0,09476	0,08786	0,082782	0,10211
tfidf+pagerank	0,07812	0,088256	0,084047	0,090885	0,113176
sif+pagerank	0,097326	0,092872	0,094409	0,079144	0,091722
wmd+pagerank	0,102108	0,116491	0,107691	0,117695	**0,127931**
centroid	0,081216	0,106097	0,081706	0,106097	0,119833
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.28. ROUGE-2 Recall scores obtained by extracting an English summary using English and Czech documents as input. Word embedding models used: FastText.

EN + CS to CS

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,060353	0,062577	0,093248	0,087242	0,071851
idf+pagerank	0,059758	0,065253	0,096158	0,091651	0,075721
tfidf+pagerank	0,062365	0,059731	**0,097049**	0,097506	0,071078
sif+pagerank	0,069747	0,066688	0,095923	0,082662	0,075587
wmd+pagerank	0,068536	0,06843	0,073638	0,072888	0,096026
centroid	0,064165	0,062061	0,063013	0,067233	0,087609
textrank	-	-	-	-	0,14924
lexrank	-	-	-	-	**0,19208**

Table 5.29. ROUGE-2 Recall scores obtained by extracting a Czech summary using English and Czech documents as input. Word embedding models used: FastText.

EN + CS to CS

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,067106	0,065904	0,067491	0,064956	0,079702
idf+pagerank	0,065325	0,063	0,085812	0,085177	0,079047
tfidf+pagerank	0,061898	0,063044	0,079956	0,080061	0,075697
sif+pagerank	0,06876	0,074195	0,069433	0,080882	0,073874
wmd+pagerank	0,072488	0,078272	0,077511	0,079555	**0,107802**
centroid	0,081337	0,080564	0,076437	0,087398	0,104431
textrank	-	-	-	-	0,09978
lexrank	-	-	-	-	**0,13514**

Table 5.30. ROUGE-2 Recall scores obtained by extracting a Czech summary using English and Czech documents as input. Word embedding models used: FastText.

EN + HI to EN

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,097522	0,101322	0,104035	0,101322	0,092718
idf+pagerank	0,071867	0,090367	0,083086	0,094135	0,091233
tfidf+pagerank	0,064764	0,080668	0,07836	0,080668	0,097686
sif+pagerank	0,070958	0,090971	0,064868	0,092607	0,059421
wmd+pagerank	0,121605	0,120989	0,125784	0,121219	**0,140637**
centroid	0,082692	0,071389	0,099192	0,089344	0,122069
textrank	-	-	-	-	0,07640
lexrank	-	-	-	-	0,12761
corerank	-	-	-	-	**0,14719**

Table 5.31. ROUGE-2 Recall scores obtained by extracting an English summary using English and Hindi documents as input. Word embedding models used: FastText.

EN + HI to EN

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,085299	0,086924	0,086083	0,086924	0,112147
idf+pagerank	0,082247	0,089749	0,097339	0,099379	0,10211
tfidf+pagerank	0,081942	0,092014	0,084109	0,094205	0,113176
sif+pagerank	0,085489	0,09275	0,087765	0,083861	0,091722
wmd+pagerank	0,113154	0,106717	0,111635	0,106955	**0,127931**
centroid	0,08431	0,092954	0,111615	0,108802	0,119833
textrank	-	-	-	-	0,09773
lexrank	-	-	-	-	0,12549
corerank	-	-	-	-	0,12301

Table 5.32. ROUGE-2 Recall scores obtained by extracting an English summary using English and Hindi documents as input. Word embedding models used: FastText.

EN + HI to HI

	A		B		
	10+10	10+10*	10+10	10+10*	10+0
media+pagerank	0,04173	0,042242	0,0551	0,04231	0,053176
idf+pagerank	0,043103	0,042157	0,050897	0,054038	0,053997
tfidf+pagerank	0,040924	0,040924	0,051986	0,047957	**0,066668**
sif+pagerank	0,047261	0,047239	0,053691	0,050885	0,050933
wmd+pagerank	0,056265	0,056265	0,048165	0,050902	0,058477
centroid	0,048646	0,052205	0,048646	0,052205	0,057212

Table 5.33. ROUGE-2 Recall scores obtained by extracting an Hindi summary using English and Hindi documents as input. Word embedding models used: FastText.

EN + HI to HI

	A		B		
	5+5	5+5*	5+5	5+5*	5+0
media+pagerank	0,040973	0,039364	0,051528	0,042519	0,064609
idf+pagerank	0,033712	0,035712	0,039368	0,034338	0,058413
tfidf+pagerank	0,046041	0,041955	0,041347	0,036394	0,055375
sif+pagerank	0,034765	0,039463	0,034908	0,041982	0,049498
wmd+pagerank	0,064762	0,065222	0,065341	**0,068392**	0,064097
centroid	0,036775	0,035924	0,036775	0,035924	0,053752

Table 5.34. ROUGE-2 Recall scores obtained by extracting an Hindi summary using English and Hindi documents as input. Word embedding models used: FastText.

Considering 5 articles for language, in the EN+HI to HI task, is the only FastText case in which adding articles, not in the target language, brings benefits.

Chapter 6

Conclusions and future works

The aim of this thesis is to propose and evaluate new techniques for extractive summarization. Specifically, it points to extract a summary from a collection of textual documents written in multiple languages, by exploiting the semantic relationships between cross-lingual content. In particular, starting from the implementation of TextRank, it has been modified in a way to enrich it with word embedding models, such as Word2Vec and FastText, and in such a way that it is able to extract a summary from articles in multiple languages. Also another monolingual summarizer, proposed by Rossiello et al. (2017), has been made able to produce a summary getting as input documents in different languages.

In fact, the study aims to verify whether the use of embeddings and of articles, not only in the target language, can lead to better performances. In order to achieve this, all the algorithms proposed have been compared to their alternative version which uses Google Translator and to state-of-the-art methods of extractive summarization.

The results demonstrate that using cross-lingual text correlations improves summarizer performance. In particular, starting from English documents, if French or Arabic or Czech documents are added to them, a better summary is obtained than that extracted only from the articles in English. Also in the extraction of French summaries, the additional information, brought by considering also the articles in English, improves the results. The same applies to Hindi summaries.

Various form of Word2Vec and FastText based sentence embeddings have been incorporated in TextRank by using as edge weights in the graph, instead

of the original normalized word overlap, the cosine similarity and the word mover's distance between the sentence vectors. With the exception of the case in which summaries are obtained in Czech, the best proposed method is always better than the baselines, showing that enriching TextRank with word embeddings allows to obtain more precise summaries. This means that embeddings are able to add information to the model and, at the same time, the latter is able to exploit these informations to improve summary quality. In particular, the version that computes word mover's distance between sentences is the best one, showing that, between the two distance measurements, the most effective in this task is WMD.

All the conclusions presented so far concern only the methods that exploit Word2Vec models. For FastText the results show how the proposed techniques do not allow to obtain valid summaries by exploiting this type of embedding.

It would be useful to compare the results obtained with those of the existing crosslingual summarizer. Specially, it would be interesting the comparison with those that could be getted by SimFusion and Corank[37]. This test could not be done because the code of these methods was not found and moreover, as reported in the corresponding paper, the author only tries to get a summary from articles in English and Chinese, and the latter is a language not considered in this thesis.

Another thing that should be tried is to use other techniques that allow to obtain sentence vectors. In fact, in this thesis, sentence embeddings are getted only by combining the one of the words contained in it. An example of model that can be tested is Doc2Vec [25].

Bibliography

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. "A Simple but Tough-to-Beat Baseline for Sentence Embeddings". In: *ICLR*. 2017.

[2] Timo Böhm. *The General Ideas of Word Embeddings*. URL: https://towardsdatascience.com/the-three-main-branches-of-word-embeddings-7b90fa36dfb9.

[3] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146. ISSN: 2307-387X.

[4] S. Brin and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". In: *Seventh International World-Wide Web Conference (WWW 1998)*. 1998. URL: http://ilpubs.stanford.edu:8090/361/.

[5] Luca Cagliero, Paolo Garza, and Elena Baralis. "ELSA: A Multilingual Document Summarization Algorithm Based on Frequent Itemsets and Latent Semantic Analysis". In: *ACM Trans. Inf. Syst.* 37.2 (Jan. 2019). ISSN: 1046-8188. DOI: 10.1145/3298987. URL: https://doi.org/10.1145/3298987.

[6] Aditya Chetan et al. "CoReRank: Ranking to Detect Users Involved in Blackmarket-Based Collusive Retweeting Activities". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. WSDM '19. Melbourne VIC, Australia: ACM, 2019, pp. 330–338. ISBN: 978-1-4503-5940-5. DOI: 10.1145/3289600.3291010. URL: http://doi.acm.org/10.1145/3289600.3291010.

[7] Günes Erkan and Dragomir R. Radev. "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization". In: *CoRR* abs/1109.2128 (2011). arXiv: 1109.2128. URL: http://arxiv.org/abs/1109.2128.

[8] *FastText*. URL: https://github.com/facebookresearch/fastText.

[9] Kavita Ganesan. "ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks". In: (2015).

[10] Edouard Grave, Armand Joulin, and Quentin Berthet. *Unsupervised Alignment of Embeddings with Wasserstein Procrustes*. 2018. arXiv: 1805.11222 [cs.LG].

[11] Vishal Gupta and Gurpreet Lehal. "A Survey of Text Summarization Extractive Techniques". In: *Journal of Emerging Technologies in Web Intelligence* 2 (Aug. 2010). DOI: 10.4304/jetwi.2.3.258-268.

[12] Lucas de Haas. "Extractive Summarization using sentence embeddings". Theses. Department of Information and Computing Sciences, Faculty of Science, Utrecht University, Nov. 2017.

[13] Felix Hill, Kyunghyun Cho, and Anna Korhonen. "Learning Distributed Representations of Sentences from Unlabelled Data". In: *CoRR* abs/1602.03483 (2016). arXiv: 1602.03483. URL: http://arxiv.org/abs/1602.03483.

[14] Nisarg Jhaveri, Manish Gupta, and Vasudeva Varma. "clstk: The Cross-Lingual Summarization Tool-Kit". In: Jan. 2019, pp. 766–769. ISBN: 978-1-4503-5940-5. DOI: 10.1145/3289600.3290614.

[15] Armand Joulin et al. *Bag of Tricks for Efficient Text Classification*. 2016. arXiv: 1607.01759 [cs.CL].

[16] Armand Joulin et al. "Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.

[17] Ryan Kiros et al. "Skip-Thought Vectors". In: *CoRR* abs/1506.06726 (2015). arXiv: 1506.06726. URL: http://arxiv.org/abs/1506.06726.

[18] Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. "Summarization Based on Embedding Distributions". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1984–1989. DOI: 10.18653/v1/D15-1232. URL: https://www.aclweb.org/anthology/D15-1232.

[19] Matt Kusner et al. "From word embeddings to document distances". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)* (Jan. 2015), pp. 957–966.

[20] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[21] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

[22] Elvys Linhares Pontes. "Compressive Cross-Language Text Summarization". Theses. Université d'Avignon, Nov. 2018. URL: https://hal.archives-ouvertes.fr/tel-02003886.

[23] Elvys Linhares Pontes, Stéphane Huet, and Juan-Manuel Torres-Moreno. "A Multilingual Study of Compressive Cross-Language Text Summarization". In: *17th Mexican International Conference on Artificial Intelligence (MICAI)*. Vol. 11289. Lecture Notes in Artificial Intelligence. Guadalajara, Mexico: Springer, 2018, pp. 109–118. DOI: 10.1007/978-3-030-04497-8_9. URL: https://hal.archives-ouvertes.fr/hal-02021602.

[24] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Text". In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 404–411. URL: https://www.aclweb.org/anthology/W04-3252.

[25] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *arXiv:1310.4546 [cs, stat]* (Oct. 2013). arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

[26] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]* (Sept. 2013). arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[28] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. "Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies". In: *NAACL-ANLP 2000 Workshop: Automatic Summarization*. 2000. URL: https://www.aclweb.org/anthology/W00-0403.

[29] Gaetano Rossiello, Pierpaolo Basile, and Giovanni Semeraro. "Centroid-based Text Summarization through Compositionality of Word Embeddings". In: *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017. DOI: 10.18653/v1/W17-1003. URL: https://www.aclweb.org/anthology/W17-1003.

[30] *ROUGE*. URL: https://rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/#.XmjccXdFwZ4.

[31] Joao Schapke. *Evolution of text representation in NLP*. URL: https://towardsdatascience.com/evolution-of-word-representations-in-nlp-d4483fe23e93.

[32] Peter Schönemann. "A generalized solution of the orthogonal procrustes problem". In: *Psychometrika* 31.1 (1966), pp. 1–10. URL: https://EconPapers.repec.org/RePEc:spr:psycho:v:31:y:1966:i:1:p:1-10.

[33] Josef Steinberger et al. "JRC's Participation at TAC 2011: Guided and MultiLingual Summarization Tasks". In: *TAC*. 2011.

[34] *sumy 0.8.1*. URL: https://pypi.org/project/sumy/.

[35] *Tf-idf*. URL: https://en.wikipedia.org/wiki/Tf%E2%80%93idf.

[36] Merijn van Erp and Lambert Schomaker. "Variants of the Borda count method for combining ranked classifier hypotheses". English. In: *Proceedings 7th International Workshop on frontiers in handwriting recognition (7th IWFHR)*. Ed. by Lambert Schomaker and Louis Vuurpijl. $date_submitted$: $2004Rights$: $UniversityofGroningen$. International Unipen Foundation, 2000, pp. 443–452. ISBN: 90-76942-01-3.

[37] Xiaojun Wan. "Using Bilingual Information for Cross-Language Document Summarization." In: Jan. 2011, pp. 1546–1555.

[38] Xiaojun Wan, Huiying Li, and Jianguo Xiao. "Cross-Language Document Summarization Based on Machine Translation Quality Prediction". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 917–926. URL: https://www.aclweb.org/anthology/P10-1094.

[39] Xiaojun Wan et al. “Cross-language document summarization via extraction and ranking of multiple summaries”. In: *Knowledge and Information Systems* 58 (2018), pp. 481–499.