Eötvös Loránd Tudományegyetem

Informatikai Kar

Adattudományi és Adattechnológiai Tanszék

Content based recommendation in catalogues of multilingual documents

Dr. Horváth Tomáš

Egyetemi adjunktus

Salamon Vilmos Tibor

Programtervező Informatikus MSc

Modellalkotó Informatikus szakirány

Budapest, 2018

Contents

Introduction

In our modern world the most valuable resource is information. Unsurprisingly some of the first electronic machines that humans built was about processing, acquiring and transmitting information. And this hasn't changed ever since.

Humans today generate more information than ever. The amount of data generated daily on community websites, forums, news sites is more than anyone could ever read or process in their lifetimes. Using computers to filter that information for us is essential in our everyday life. Companies use recommendation systems to suggest their products to their customers and every day billions of users attend to search engines to find what they are looking for on the web. It is not a surprise that information retrieval, the task of finding relevant documents is a continuously investigated and researched topic.

Of the millions of textual documents generated daily only a limited amount is available in a given language and this language diversity of the documents poses a great challenge in information retrieval. This presents a problem for users who are in the need of information in multiple languages.

For example a journalist or investigator, governmental institution, market analyst might want to find documents similar to a given one but in different language or companies that are constantly watching what people are writing about them or about their new product. Widespread information retrieval systems are generally monolingual and therefore incapable of satisfying these users information needs.

One straightforward way users try solve this issue is by translating their query to the target language. But apart from the possibility that there could be more than one target language, translation is not free. Even with the recent emerge of machine translation it is not yet a viable way to rely on these systems for those that require these on a daily basis.

Although information retrieval is mainly about monolingual data, one area of it focuses specifically on the task of querying a set of multilingual documents in one language in order to find and retrieve relevant documents in another language. This area is called Cross-Lingual Information Retrieval (CL-IR). In my thesis this field is the central topic.

The aim of my thesis is to review the field of CL-IR and possibly contribute to it with a method that can solve the cross-lingual document recommendation task. To do this I first overview the more general field of information retrieval and learn how the concepts from those are adapted to the cross-lingual scope. Then data from various sources and self made collection is used to experiment with the models that are applied in the literature and that I have came up on my own.

The thesis is divided into 4 chapters. In the first chapter I overview the topic of information retrieval to give a general sense about how text is being processed in a monolingual scope. Then in the second chapter I introduce a few algorithms and methods that are needed to understand the models that were developed. The third chapter describes baseline models found in the literature and the ones proposed by this thesis. In the fourth chapter I detail the conducted experiments on datasets with the baseline models and the proposed models to evaluate their performance.

Chapter 1

Overview of Information Retrieval

Information retrieval (IR) is the process of finding information in an unstructured source of data where unstructured means that it is hard or impossible to translate it for a computer. The data that is mostly the subject of information retrieval are text documents, but other types such as images, videos or audio are also possible targets. The most prominent usage usage of IR consists of finding documents or document fragments relevant to a query. A query is a short (or not so short) piece of text that describes the users information need. The goal of an IR system is to return the most relevant documents to the query from a collection of documents. The system achieves its purpose if the returned documents satisfy the information need of the user. Another application of information retrieval is recommendation based on the content of the documents. In this use-case the document in question constitutes the query and the goal is to find relevant documents to it in the collection [9].

The task of information retrieval can be defined as follows. Given a set of documents $D = \{d_1, d_2, ..., d_n\}$ and a query document q we want to find and retrieve the most relevant documents from D to query q.

We say two documents are relevant to each other if their content is similar. In text if documents are about the same topic, in images if its about the same object etc. So in order to return relevant documents to a query we have to be able to compare them. The challenge in this is the unstructured nature of the data. It is very easy to compare structured objects such as vectors, but there is no easy way for a computer to compare two ordered set of words with meaning that constitutes a document.

1.1 Preprocessing techniques

The road to be able to return a relevant document given a query starts with the preprocessing of those target documents. Preprocessing is the first step in any data mining and machine learning task. It is the process of transforming the data into a proper and understandable form for the algorithms. Assuming that we have the target collection ready as text documents and loaded the following steps can be taken to assure the success of our algorithm.

1.1.1 Tokenization

Raw texts are built up of paragraphs, sentences, words and punctuation. Three of those four are too difficult to comprehend for an algorithm so most information retrieval models use only the words, which is arguably the most important building block of the texts. Therefore the most important thing to do with raw texts is to transform them into list of tokens.

Not every token is valuable to us though: some of them do not have a useful meaning such as web links or numbers. These are usually filtered out along with something called stopwords. Stopwords are the overused words that are the most common in a language such as ”the” or ”a”. Because of this trait they are viewed as noise and filtered out with the other unimportant tokens.

1.1.2 Stemming and Lemmatization

Words are used in different forms in written or spoken language. Write, writes, written, wrote, writing are all forms of the same word. Although humans might be able to draw connection between them algorithms can not. Because there are one or two addition or changes of letters in them, for algorithms these are different words without any relationship. For this task natural language processing researchers developed two kinds of methods to reduce these words to their base form.

The first one is called stemming and it is the crude process of cutting the end of the word by some heuristics. The second one is called lemmatization which is a more sophisticated way of achieving the same goal. It uses rules and dictionaries to find the proper base of a word.

1.2 Bag-of-words model

The most simple and popular way of representing a text document is the bag-of-words (BoW) model. In this method we view every text as a set or multiset of words. Every document is represented by a long vector of real numbers, each component corresponding to a term in the dictionary. There are multiple possibilities of what each component of a documents vector will contain based on what weighting scheme we use. If our corpora is $D = \{d_1, d_2, ..., d_n\}$ and our dictionary is $T = \{t_1, t_2, ..., t_m\}$ then the output of the representation will be an $\mathbb{R}^{m \times n}$ matrix called the term-document matrix.

$$
\begin{array}{c}
 \\
\textbf{term}_1\rightarrow \\
\textbf{term}_2\rightarrow \\
\vdots \\
\textbf{term}_m\rightarrow
\end{array}
\begin{array}{c}
\begin{array}{cccc}
\textbf{document}_1 & \textbf{document}_2 & \cdots & \textbf{document}_n \\
\downarrow & \downarrow & & \downarrow
\end{array} \\
\begin{bmatrix}
w(t_1,d_1) & w(t_1,d_2) & \dots & w(t_1,d_n) \\
w(t_2,d_1) & w(t_2,d_2) & \dots & w(t_2,d_n) \\
\vdots & \vdots & \ddots & \vdots \\
w(t_m,d_1) & w(t_m,d_2) & \dots & w(t_m,d_n)
\end{bmatrix}
\end{array}
$$

Figure 1.1: Term-document matrix

The main advantage of this model is that every document is described as an $\underline{\mathbf{d}} \in \mathbb{R}^m$ vector, so the documents are now in the same vector space. The drawback of this model is that it disregards any information on word order. Therefore the two sentences "This is a cat, not a dog." and "This is a dog, not a cat." although mean exactly the opposite, their representation will be the same. But because of its flexibility and simplicity it is by far the most used vector space model in the field of information retrieval.

1.3 Term-weighting schemes

A term-weighting scheme is a $w : T \times D \rightarrow \mathbb{R}$ function that describes the semantic contribution of the term to the document. The goal of the term-weighting is to give bigger impact to "better" words. The word "better" can mean different things in different applications[6, 18].

1.3.1 Term-frequency

The most simple term-weighting schemes are the simple term-frequency and its variations. The intuition is straightforward; the more times a term occurs in a document, the more important it is for its subject. Let $tf_{d,t}$ be the number of occurrences of term t in document d.

One of the variations of the term-frequency is the binary term-weighting. In this case each term has a 0 or 1 value in the documents vector depending on its presence in the document. This way the magnitude of the contribution of the term to the document is lost.

$$w(t,d) = \begin{cases} 0, & t \notin d \\ 1, & t \in d \end{cases}$$

Another variation is using a sublinear term-frequency. This is a common modification that soothes the effect of very frequent terms in a document by applying some function on the tf.

$$w(t,d) = 1 + \log \mathrm{tf}_{d,t}$$

We can also normalize the term-frequencies in the documents to standardize the data. Many algorithms need this to work correctly.

$$w(t,d) = a + (1-a)\frac{\mathrm{tf}_{d,t}}{\max_t \mathrm{tf}_{d,t}}$$

1.3.2 Inverse document-frequency

The problem of term-frequency weighting is that every term in a document is considered equally important (before the occurrence weighting). But this is not always true; if we have a set of documents about different kinds of apples, the word "apple" will most probably occur in all of them; therefore it is not a good feature for the information retrieval task, yet it is going to have huge term-frequency.

To alleviate the effect of frequently occurring words in a document another weight is used called the inverse document-frequency. The intuition behind it is that the less the word occurs in the corpora, the more its contribution is to the current document.

If we define df_t as the document frequency of term t in our corpora the inverse document-frequency is:

$$\mathrm{idf}_t = \log \frac{n}{\mathrm{df}_t}$$

It follows from the definition that for a high *df* of a term the *idf* will be low, hence the inverse document-frequency punishes the frequently occurring terms in the corpora.

A modification of the idf can be made by using a smoothing parameter α. This can lower the impact of very infrequent words which would otherwise have a huge effect on the weights.

$$\mathrm{idf}_t = \log \frac{\alpha + n}{\alpha + \mathrm{df}_t}$$

1.3.3 Tf-idf weighting-scheme

The most widely used weighting scheme in information retrieval is the combination of a variant of term-frequency and the inverse document-frequency.

$$\text{tf-idf}_{d,t} = \mathrm{tf}_{d,t} \cdot \mathrm{idf}_t$$

Based on the theory previously discussed the tf-idf weighting-scheme decreases the effect of globally frequent and increases the effect of locally frequent terms in the dictionary. It will give high weight to those terms that occur often in the document but at the same time rarely in the whole corpora.

1.4 Similarity measures

With the bag-of-words model and tf-idf weighting-scheme the documents and queries can be represented as an $\mathbb{R}^m$ vector. In order to find which of them are relevant a metric is needed to calculate distance between the vectors. Traditional Euclidean distance is rarely used because it is not normalized for the length of the a documents. Two documents that are on the same topic can be very far apart because of the difference in their length. For this reason the most popular metric used in information retrieval applications is the cosine similarity.

For example consider the following four documents: d_1="apple", d_2="apple apple apple" and d_3="orange apple" and d_4 ="orange". Clearly d_1 and d_2 are about

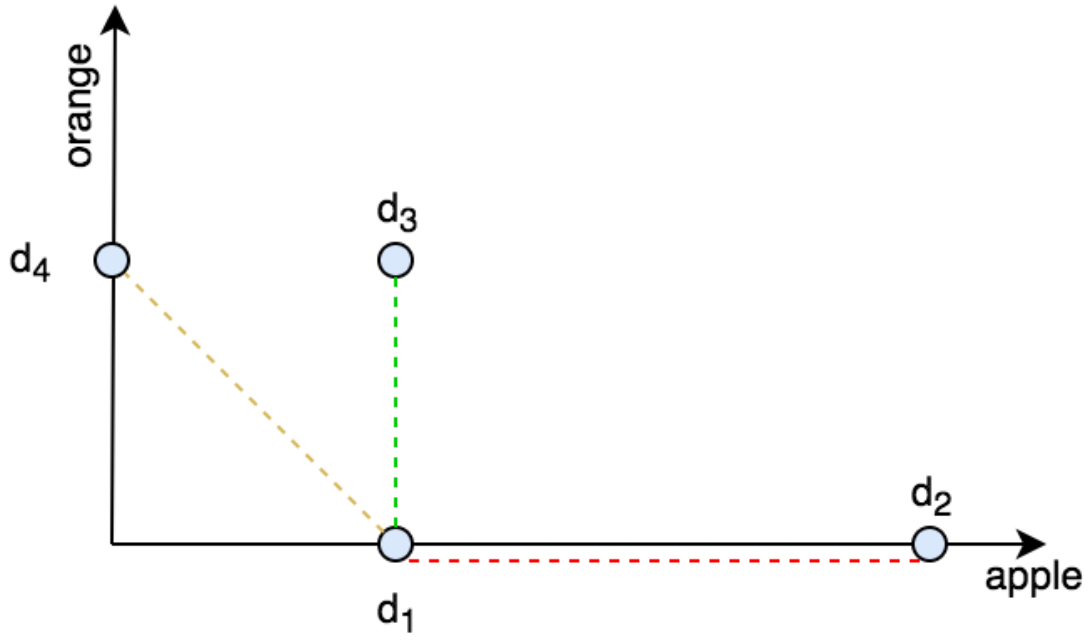

Figure 1.2: Problem with Euclidean distance

the same topic, yet both d_3 and d_4 will be much closer to d_1 than d_2. While with cosine-similarity the order of relevance to d_1 will be d_2, d_3, d_4, as it should be.

Let $a, b \in \mathbb{R}^m$ be the vector representation of two documents. Their cosine similarity is

$$cos(a, b) = \frac{a^T b}{||a||_2||b||_2}$$

This measure is of course exactly the cosine of the angle between the two vectors.

1.5 Challenges of the bag-of-words models

Apart from not understanding natural language some of the greatest challenges in information retrieval are the words with more meaning or words with more form [23].

Synonymous words are those that are although not in the same form can mean the same thing and can be used interchangeably. The problem is clear; if a user types in "windows" to a search engine it is equally possible that he wants to buy an operation system or a house.

Polysemous words are words with more than one meaning that are somehow connected. For example "man" can mean a human male or the species as a whole.

A different concept that can cause even bigger troubles is homonymy where the same term means entirely different concepts; such as "bank" can mean the financial institution or the river bank as a geographical location.

The problem of choosing the right meaning of a word from the concept is called word sense disambiguation and is a whole area of research in the field of natural language processing.

1.6 Cross-Lingual Information Retrieval

The task of Cross-Lingual Information Retrieval can be defined as follows. Given a language set $L = \{l_1, ..., l_p\}$, a set of documents in each language $D_{l_i} = \{d_{l_i,1}, ..., d_{l_i,n_i}\}$ and query document q written in one of the languages; find the most relevant documents to the query from all the $D_{l_i}(i = 1, .., p)$ document sets.

The task of CL-IR takes the challenge of information retrieval to another level. Not only do we have to find relevant documents in the same language, but we also have to deal with the language diversity of the documents. Three main approaches have emerged: bilingual dictionary based, machine translation based and inter-lingual representation based models [17].

1.6.1 Bilingual dictionary based models

The most simple method is to translate either the query or the target documents word-by-word with a bilingual dictionary and then use monolingual information retrieval algorithms on the translated documents. This has been used in many papers ¡sources¿ successfully but it suffers from a multitude problems. One of the difficulties is getting good dictionary for many language pairs. Another problem is that even if the words are translatable from one language to the other this cannot be done without proper lemmatization. Finally, with dictionary based translations there is no word sense disambiguation so the polysemy of words will not be resolved. Therefore this method is very error-prone in addition to depending heavily on the underlying dictionary used.

1.6.2 Statistical machine translation based models

A more sophisticated method involves using some kind of machine translation to transform the query and the target documents into the same language. This method has less problems than the bilingual dictionary based models but this comes with a price. Machine translation systems from a third party (such as Google or Microsoft) are not free to use and although it is possible to make our own system it is a huge computational burden to bear not to say it wont be nearly as accurate.

1.6.3 Inter-lingual representation based models

The most well-researched models are free of any third party resource. As their name suggests inter-lingual representation based methods try to embed vector space model of the languages into a common vector space using aligned concept of document vectors. For this purpose they use many kinds of numerical and statistical algorithms that have been proven useful in monolingual applications.

Because of the previously mentioned challenges of the other methods this thesis focuses on these kinds of methods - as do the majority of the research papers on the field. The models are presented with detail in Chapter 3.

Chapter 2

Theoretical background of the models

This chapter explores the theoretical background of the models that were implemented to solve the problem of cross-lingual document recommendation. Three main methods make up the backbone of the models: K-means clustering, linear least squares and Latent Semantic Indexing.

K-means clustering is the most popular clustering method that groups the observations into k clusters (partitions). The elements in a common cluster should be more similar to each other in some sense, than to those in another cluster.

Linear least squares is the well-known problem of approximating some variable as a linear combination of other variables. It is applied extensively in mathematics, statistics and engineering as one of the most common tasks.

Latent semantic indexing (LSI) is a dimension reduction method of information retrieval that reduces the term-vectors of documents to vectors of latent semantic concepts. It is arguably the most popular dimension reduction tool in text mining and natural language processing.

2.1 K-means clustering

Let $X = \{\underline{\mathbf{x}}_1, ..., \underline{\mathbf{x}}_n\}$ be our dataset where $\underline{\mathbf{x}}_i \in \mathbb{R}^m$ $(i = 1, ..., n)$. The goal of the K-means clustering is to partition this set to $k < n$ pairwise disjoint subsets known as clusters in which the variance of the datapoints is minimal. This is achieved by minimizing the intra-cluster distance of the points to the cluster centroid[8, 9].

A partition of X is $C = \{C_1, ..., C_k\}$, where for all $i : C_i \subset X$, if $i \neq j$: $C_i \cap C_j = \emptyset$ and $\bigcup_{i=1}^{k} C_i = X$. The centroid $\underline{\mu}_i$ of cluster C_i is defined as the mean of the points in the cluster:

$$\underline{\mu}_i := \frac{1}{|C_i|} \sum_{\underline{\mathbf{x}} \in C_i} \underline{\mathbf{x}}$$

The objective function to minimize is the sum of the squared distance of each vector to their representing clusters centroid, also known as the Residual Sum of Squares (RSS).

$$\mathrm{RSS}(C) = \sum_{i=1}^{k} \sum_{\underline{\mathbf{x}} \in C_i} ||\underline{\mathbf{x}} - \underline{\mu}_i||^2$$

The output of the minimization process is the best $\hat{C}$ cluster-set that represent the data.

$$\hat{C} = \arg\min_{C} \mathrm{RSS}(C)$$

The minimization process is as follows:

1. Randomly choose the initial centroids of the clusters from the dataset.

2. Repeat the following two steps until convergence:

   (a) **Assignment step**

   Assign documents to the clusters based on their squared distance from the centroids.

   $$C_i^{(s)} = \{\underline{\mathbf{x}} \in X \quad | \quad \forall j : ||\underline{\mathbf{x}} - \underline{\mu}_i^{(s-1)}||^2 \leq ||\underline{\mathbf{x}} - \underline{\mu}_j^{(s-1)}||^2\}$$

   (b) **Update step**

   Update the centroids of the clusters based on their current members.

   $$\underline{\mu}_i^{(s)} = \frac{1}{|C_i^{(s)}|} \sum_{\underline{\mathbf{x}} \in C_i^{(s)}} \underline{\mathbf{x}}$$

Intuitively the K-means algorithm clusters the data by moving the centroids of the clusters in a way that decreases the RSS. The clustering in process is shown in Figure 2.1.

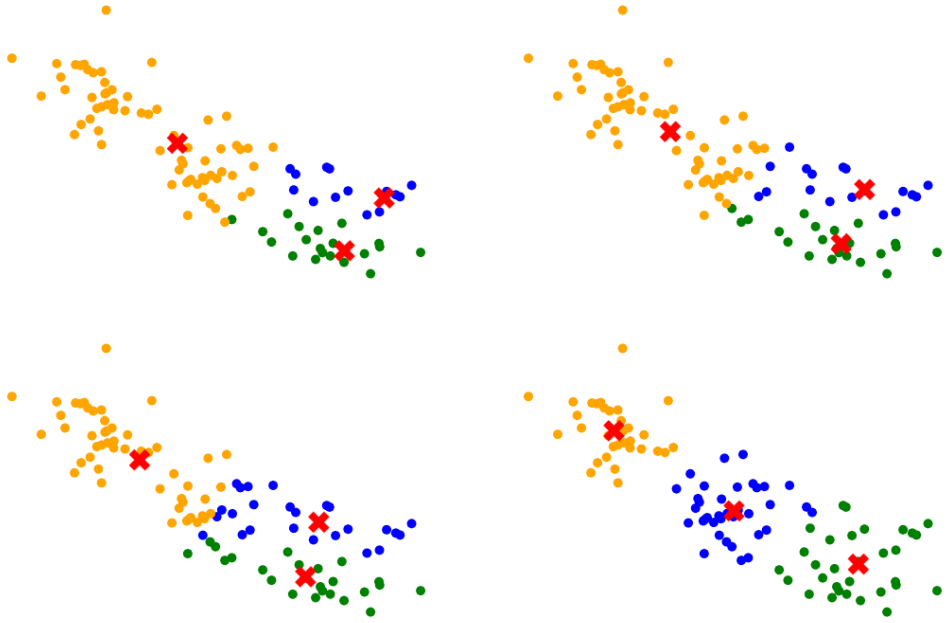

Figure 2.1: K-means clustering process in steps 1,2,4,8. The centroids ("x"-es) start from a random initial position and slowly find their best spot by minimizing the intra-cluster distance of the points to the centroids.

It can be shown that the K-means clustering algorithm converges to a minimum because the assignment and the update steps decrease the RSS in every iteration. Unfortunately there is no guarantee that this is a global minimum. In particular in a dataset with lots of outlier points and an unlucky initialization it can easily get stuck in a local minimum. For this reason sometimes more than one runs are made with different initializations.

2.2 Linear Least Squares

This section describes the method of solving overdetermined linear systems in a least-squares sense and the similar solution to underdetermined systems[24]. Let us consider the overdetermined linear system

$$\mathbf{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\underline{\mathbf{x}} \in \mathbb{R}^n$ and $\underline{\mathbf{b}} \in \mathbb{R}^m$ for $n < m$.

The columns of $\mathbf{A}$ span a subspace in $\mathbb{R}^m$ denoted by $C(\mathbf{A})$. Since $\underline{\mathbf{b}}$ is probably not in this subspace it is highly likely that this overdetermined linear system does not have solution. Instead we want to find the closest point to $\underline{\mathbf{b}}$ in $C(\mathbf{A})$ denoted by $\underline{\mathbf{x}}^*$. To do this we want to minimize the (squared) error of the system:

$$\underline{\mathbf{x}}^* = \arg\min_{\underline{\mathbf{x}}} ||\mathbf{A}\underline{\mathbf{x}} - \underline{\mathbf{b}}||_2^2$$

In terms of matrices the following derivation can be made for the solution:

$$\begin{aligned} ||\mathbf{A}\underline{\mathbf{x}} - \underline{\mathbf{b}}||_2^2 &= (\mathbf{A}\underline{\mathbf{x}} - \underline{\mathbf{b}})^T(\mathbf{A}\underline{\mathbf{x}} - \underline{\mathbf{b}}) \\ &= (\underline{\mathbf{x}}^T\mathbf{A}^T - \underline{\mathbf{b}}^T)(\mathbf{A}\underline{\mathbf{x}} - \underline{\mathbf{b}}) \\ &= \underline{\mathbf{x}}^T\mathbf{A}^T\mathbf{A}\underline{\mathbf{x}} - \underline{\mathbf{x}}^T\mathbf{A}^T\underline{\mathbf{b}} - \underline{\mathbf{b}}^T\mathbf{A}\underline{\mathbf{x}} + \underline{\mathbf{b}}^T\underline{\mathbf{b}} \\ &= \underline{\mathbf{x}}^T\mathbf{A}^T\mathbf{A}\underline{\mathbf{x}} - 2\underline{\mathbf{x}}^T\mathbf{A}^T\underline{\mathbf{b}} + \underline{\mathbf{b}}^T\underline{\mathbf{b}} \end{aligned}$$

.

Where the fact that $\underline{\mathbf{x}}^T\mathbf{A}^T\underline{\mathbf{b}}$ and $\underline{\mathbf{b}}^T\mathbf{A}\underline{\mathbf{x}}$ are both scalars and the transpose of each other thus they are equal are used. After differentiating by $\underline{\mathbf{x}}$ and setting it to 0 the minimization problem takes up the form

$$(\mathbf{A}^T\mathbf{A})\underline{\mathbf{x}} = \mathbf{A}^T b$$

This is the so called normal equation because it projects $\underline{\mathbf{b}}$ onto $C(\mathbf{A})$. It can be proven that $\text{rank}(\mathbf{A}^T\mathbf{A}) = \text{rank}(A)$ and thus $\mathbf{A}^T\mathbf{A}$ is invertible when the columns of $\mathbf{A}$ are linearly independent. In this case the unique solution to the minimization problem is:

$$\underline{\mathbf{x}}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\underline{\mathbf{b}}$$

Where $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}$ is a special case of the Moore-Penrose pseudoinverse (specifically the left inverse) and can be denoted by $\mathbf{A}^+$.

For an underdetermined system where $n > m$ the solution is similar. In that case there are infinite solutions and we want to choose the one with the least norm:

$$\underline{\mathbf{x}}^* = \arg\min_{\underline{\mathbf{x}}}\{||\underline{\mathbf{x}}|| : \mathbf{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}\}$$

If $\mathbf{A}$ has linearly independent rows (and thus $\mathbf{A}\mathbf{A}^T$ is invertible) then it can be shown that the solution is:

$$\underline{\mathbf{x}}^* = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\underline{\mathbf{b}}.$$

Where $\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$ is the right pseudoinverse.

2.2.1 QR Factorization for efficient solution

Inverting $\mathbf{A}^T\mathbf{A}$ in the traditional way and hence solving the normal equation requires approximately $\mathcal{O}(n^3)$ operations. This is a very computationally expensive task that does not scale well when the number of columns is in the tens of thousands.

To speed up this process one can use the QR factorization technique which decomposes the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ into an orthonormal $\mathbf{Q} \in \mathbb{R}^{m\times n}$ and an upper triangular $\mathbb{R}^{n\times n}$ matrix such that $\mathbf{A} = \mathbf{Q}\mathbf{R}$. There are multiple ways to achieve the factorized form of the matrix and once we have that, a simplified form of the normal equation can be derived as follows.

$$\begin{aligned}
\mathbf{A}^T\mathbf{A}\underline{\mathbf{x}} &= \mathbf{A}^T\underline{\mathbf{b}} \iff \\
\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}\underline{\mathbf{x}} &= \mathbf{R}^T\mathbf{Q}^T\underline{\mathbf{b}} \iff \\
\mathbf{R}^T\mathbf{R}\underline{\mathbf{x}} &= \mathbf{R}^T\mathbf{Q}^T\underline{\mathbf{b}} \iff \\
\mathbf{R}\underline{\mathbf{x}} &= \mathbf{Q}^T\underline{\mathbf{b}}
\end{aligned}$$

Using the fact that $\mathbf{Q}^T\mathbf{Q} = I$ and that an upper triangular matrix can be inverted if none of its diagonal elements are zero.

With this method we have to solve an $n \times n$ dimensional upper triangular linear system which can be solved by $\mathcal{O}(n^2)$ operations.

2.3 Dimension reduction with Latent Semantic Indexing

Latent Semantic Indexing (LSI) is one of the most widely used algorithms in text mining. It is popular in all kinds of information retrieval applications from text categorization to sentiment analysis. In this section I present its theoretical background and computation[9, 3].

2.3.1 Singular Value Decomposition

For every $\mathbf{A} \in \mathbb{R}^{m \times n}$ there exists a singular value decomposition (SVD) in the form of

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where the columns of $\mathbf{U} \in \mathbb{R}^{m \times m}$ are the orthonormal eigenvectors of $\mathbf{A}\mathbf{A}^T$ called left singular-vectors, the columns of $\mathbf{V} \in \mathbb{R}^{n \times n}$ are the orthonormal eigenvectors of $\mathbf{A}^T\mathbf{A}$ called right singular-vectors and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix of the square root of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ also known as singular-values denoted by σ_i.

Since the singular value decomposition is unique up to certain column and row permutations the matrices are constructed in such a way that the singular values are in a descending order in $\mathbf{\Sigma}$

$$\sigma_r \leq \sigma_{r-1} \leq \cdots \leq \sigma_1.$$

2.3.2 Low-rank approximation problem

Low-rank approximation is a minimization problem in which we want to approximate an $\mathbf{A} \in \mathbb{R}^{m \times n}$ matrix with rank r by a $\mathbf{A}_k \in \mathbb{R}^{m \times n}$ matrix with rank k where $k < r$ in a least squared sense. This means that we want to minimize the Forbenius norm of the error matrix $\mathbf{A} - \mathbf{A}_k$:

$$||\mathbf{A} - \mathbf{A}_k||_F = \sqrt{\sum_{i,j} (\mathbf{A} - \mathbf{A}_k)_{i,j}^2}$$

One of the most significant results in this field is the Eckart-Young matrix approximation lemma which states that the solution to the low-rank approximation problem can be obtained by the singular value decomposition of the matrix[1],[9].

2.3.3 Eckart-Young theorem

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a real matrix and its singular value decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ where

$$\mathbf{U} = [\underline{\mathbf{u}}_1, \underline{\mathbf{u}}_2, \ldots, \underline{\mathbf{u}}_n]$$
$$\mathbf{V} = [\underline{\mathbf{v}}_1, \underline{\mathbf{v}}_2, \ldots, \underline{\mathbf{v}}_n]$$
$$\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$$

Furthermore let the singular values be in a descending order $\sigma_n \leq \sigma_{n-1} \leq \cdots \leq \sigma_1$. Then the solution to the rank-k matrix approximation problem is

$$\hat{\mathbf{A}_k} = \arg\min_{\mathbf{A}_k} ||\mathbf{A} - \mathbf{A}_k||_F$$

where

$$\hat{\mathbf{A}_k} = \sum_{i=1}^{k} \sigma_i \underline{\mathbf{u}}_i \underline{\mathbf{v}}_i^T.$$

What this means is that any rank-n matrix $\mathbf{A}$ can be optimally approximated as a rank-k matrix $\mathbf{A}_k$ by summarizing the product of the k-largest singular triplets from the singular value decomposition. The following formula outputs the rank-reduced $\mathbf{A}_k$ matrix:

1. Construct the SVD factorization of $\mathbf{A}$, $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.
2. Set all but the largest k singular values in $\mathbf{\Sigma}$ to zero forming $\mathbf{\Sigma}_k$. This is practically the same as removing the last rows/columns of $\mathbf{U}$ and $\mathbf{V}$.
3. The optimal rank-k approximation is then $\mathbf{A}_k = \mathbf{U}\mathbf{\Sigma}_k\mathbf{V}^T$.

This theorem gave mathematical foundation to the most popular dimensional reduction techniques of many fields such as principal component analysis (PCA), canonical correlation analysis (CCA) and latent semantic analysis/indexing (LSA/LSI).

2.3.4 Latent Semantic Indexing

Given an $\mathbf{X} \in \mathbb{R}^{m \times n}$ term-document matrix where m is the number of terms and n is the number of documents, the LSI tries to approximate the original matrix by a lower-dimensional concept-document matrix[3].

A computationally economic way to compute the rank-k approximation for this application is a method called truncated SVD in which only the largest k singular values and the corresponding columns of $\mathbf{U}$ and $\mathbf{V}$ are computed and the rest of the matrices are discarded. In this case the factorization of the term-document matrix is $\mathbf{X}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$ where $\mathbf{U}_k \in \mathbb{R}^{m\times k}$, $\mathbf{\Sigma}_k \in \mathbb{R}^{k\times k}$ and $\mathbf{V} \in \mathbb{R}^{k\times n}$ as shown on Figure 2.2.

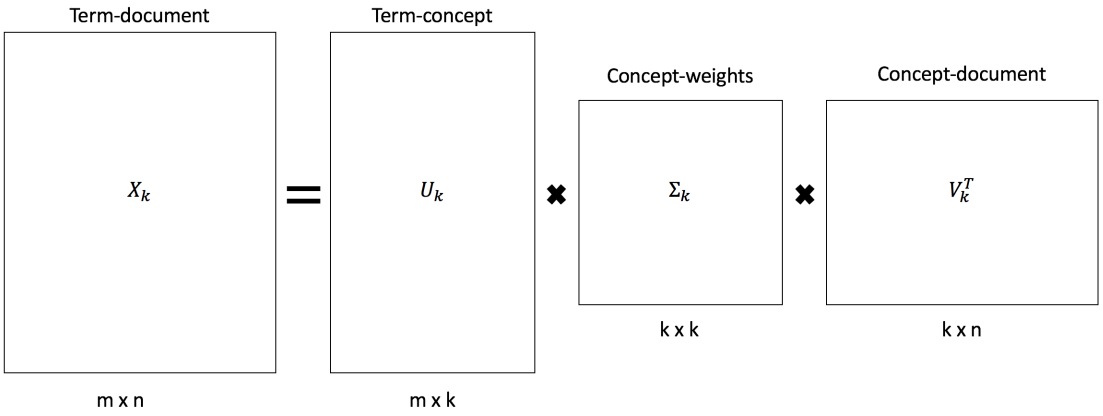

Figure 2.2: Factorization of the term-document matrix to term-concept, concept-weight and concept-document matrices

Note that the $\mathbf{U}_k$ and $\mathbf{V}_k$ matrices contain the eigenvectors of the $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$ products of the term-document matrix. Since $\mathbf{X}$ is the term-document matrix it is easy to see that $\mathbf{X}\mathbf{X}^T$ is the term-term matrix. Its elements are the correlation of each term-pair or their co-occurrence in the documents. Similarly, $\mathbf{X}^T\mathbf{X}$ is the document-document matrix whose elements are the correlations of each document. It has been proven that LSI encapsulates high-order term co-occurrence information[5].

Intuitively it is assumed that the remaining k dimensions that have not been discarded now refer to k latent concepts. $\mathbf{U}_k$ is the term-concept matrix and $\mathbf{V}_k$ is the document-concept matrix with $\mathbf{\Sigma}_k$ giving weights to these concepts. It is possible to get the representation of the documents in this latent semantic space by multiplying the documents concept-vector with their weights. Hence the latent semantic representation of our documents are

$$\hat{\mathbf{X}} = \mathbf{U}_k^T\mathbf{X} \in \mathbb{R}^{k\times n}.$$

And to fold-in a new documents we can do essentially the same with its $\underline{\mathbf{x}} \in \mathbb{R}^m$

term-vector:

$$\hat{\underline{\mathbf{x}}} = \mathbf{U}_k^T \underline{\mathbf{x}} \in \mathbb{R}^k.$$

In this form a new query is the weighted sum of its constitutent terms.

The main advantage of LSI is that by reducing the term-document matrix to a rank-k approximation the terms-with more co-occurrence will be closer in the latent concept space. This solves a few problems that conventional bag-of-words models face[15].

In Chapter 1 I have briefly discussed synonymy and why it is problematic in information retrieval. Words with similar meaning are just two different axises in the coordinate system without any connection. With latent semantic indexing since synonyms are by definition interchangeable, they will co-occur in the same contexts with the same words and thus the rank reduction will bring them closer in the semantic space.

Polysemous words are also in challenge in information retrieval. Most of these words usually have a meaning that is generally used and another not so much used meaning. LSI can filter these less frequently used meaning as if it were noise, by defining the concept weights of the term higher for the more important meaning. This helps in cases where polysemy is a frequently occurring phenomena.

Chapter 3

Models

This chapter describes the models used in the experiments, both the baselines and the ones proposed by this work. As it has been previously mentioned because of the difficulties and drawbacks of other methods this work focuses on models based on some kind of inter-lingual representations of the documents.

To measure the performance of the proposed models two methods were implemented as baselines. These have been used extensively in many scientific papers such as [2, 10, 16] for different tasks in cross-lingual information retrieval and were chosen because of their relative ease of implementation and the abundant number of resources using them. It is safe to say these that are the most popular baseline algorithms for cross-lingual document linking.

All models require training on a set of parallel documents in two languages. After training the models can convert the testing documents to the same vector space in which their comparison is possible. For each method we will show both the training and the conversion methods of the test documents and the way to measure their similarities.

The training set is a $D = \{d_1, ..., d_n\}$ set of documents that are available in both languages. Each training document has a bag-of-words vector representation in each of the languages. The vector representation of document d_i in language l_x is denoted as $\underline{\mathbf{x}}_i \in \mathbb{R}^{m_x}$ where m_x is the number of terms in language l_x. This representation is described in detail in the next chapter. Therefore the training set in language l_x is a term-document matrix $\mathbf{X} \in \mathbb{R}^{m_x \times n}$, whose columns are the bag-of-words vectors of the n training documents.

3.1 Baseline models

3.1.1 Cross-Lingual Latent Semantic Indexing

Cross-lingual Latent Semantic Indexing (or CL-LSI in short) is the extension of the monolingual LSI for the cross-lingual domain[7, 2].

Let our two languages be L_x and L_y and the term-document matrices of the parallel set of training documents $\mathbf{X} \in \mathbb{R}^{m_x \times n}$ and $\mathbf{Y} \in \mathbb{R}^{m_y \times n}$. Training the CL-LSI model begins by stacking the two matrices into a large one by keeping the documents aligned. Denote this matrix by $\mathbf{Z}$.

$$\mathbf{Z} = \begin{bmatrix} \underline{\mathbf{x}}_1 & \underline{\mathbf{x}}_2 & \cdots & \underline{\mathbf{x}}_n \\ \underline{\mathbf{y}}_1 & \underline{\mathbf{y}}_2 & \cdots & \underline{\mathbf{y}}_n \end{bmatrix} \in \mathbb{R}^{(m_x+m_y)\times n}$$

The idea is to train the LSI on this cross-lingual matrix. Using the k-truncated singular value decomposition we get

$$\mathbf{Z}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T.$$

where $\mathbf{U}_k \in \mathbb{R}^{(m_x+m_y)\times k}$ is the multilingual term-concept matrix, $\mathbf{\Sigma}_k \in \mathbb{R}^{k\times k}$ is the diagonal concept-weight matrix and $\mathbf{V}_k \in \mathbb{R}^{n\times k}$ is the document-concept matrix. From this, our new latent semantic representation of the training data is

$$\mathbf{\Sigma}_k \mathbf{V}_k^T \in \mathbb{R}^{k\times n}.$$

To fold-in a test document of language L_x, $\underline{\mathbf{x}} \in \mathbb{R}^{m_x}$, we have to extend it with zeros in the dimensions of the other language.

$$\tilde{\underline{\mathbf{x}}} = \begin{bmatrix} \underline{\mathbf{x}} \\ \underline{\mathbf{0}} \end{bmatrix} \in \mathbb{R}^{m_x+m_y}$$

Then we can fold-in this extended vector to the CL-LSI as it was a monolingual LSI. The test documents representation in the multilingual latent semantic space is then:

$$\hat{\underline{\mathbf{x}}} = \mathbf{U}_k^T \tilde{\underline{\mathbf{x}}} \in \mathbb{R}^k$$

This way all documents from L_x and L_y can be folded into a common vector space in which we can compare them using the cosine similarity.

3.1.2 Improved CL-LSI

The CL-LSI model is a well known and simple method that can be improved by using least-squares approximation in the folding procedure[10, 16]. The training steps are the same but the folding process changes slightly.

To fold-in a document $\underline{\mathbf{x}} \in \mathbb{R}^{m_x}$ of language L_x we can use the fact, that the $\mathbf{U}_k$ term-concept matrix functions as a basis over the concepts. It is used to transform documents into the concept space and fortunately it keeps the alignment of the language-dependent terms. This means that it is divided into blocks for each language:

$$\mathbf{U}_k = \begin{bmatrix} \mathbf{U}_x \\ \mathbf{U}_y \end{bmatrix}$$

where $\mathbf{U}_x \in \mathbb{R}^{m_x \times k}$ and $\mathbf{U}_y \in \mathbb{R}^{m_y \times k}$.

To get the best approximation of a vector in this language dependent basis we have to calculate the pseudoinverse of the matrix.

$$\mathbf{P}_x = (\mathbf{U}_x^T \mathbf{U}_x)^{-1} {\mathbf{U}_x}^T \in \mathbb{R}^{k \times m_x}$$

And use this to compute the best approximation to the vector $\underline{\mathbf{x}}$:

$$\underline{\hat{\mathbf{x}}} = \mathbf{P}_x \underline{\mathbf{x}} \in \mathbb{R}^k.$$

The output is the latent semantic representation of $\underline{\mathbf{x}}$ which we can use to compare documents to each other with cosine similarity.

3.1.3 K-Means centroid model

The K-Means centroid model ([10, 16]) starts with the same idea as CL-LSI. Given the parallel training documents from L_x and L_y we construct the same multilingual matrix $\mathbf{Z}$.

$$\mathbf{Z} = \begin{bmatrix} \underline{\mathbf{x}}_1 & \underline{\mathbf{x}}_2 & \cdots & \underline{\mathbf{x}}_n \\ \underline{\mathbf{y}}_1 & \underline{\mathbf{y}}_2 & \cdots & \underline{\mathbf{y}}_n \end{bmatrix} \in \mathbb{R}^{(m_x+m_y)\times n}$$

We apply the K-means clustering algorithm to these n training documents with multilingual representation. This outputs k centroid vectors each of them is a $\underline{\mathbf{c}}_i \in \mathbb{R}^{m_x+m_y}$ vector. They can be arranged into a centroid matrix $\mathbf{C} \in \mathbb{R}^{(m_x+m_y)\times k}$. Each centroid vector is made up of two component parts one for each language $\underline{\mathbf{c}}_i = [\underline{\mathbf{c}}_{x,i}, \underline{\mathbf{c}}_{y,i}]^T$ where $\underline{\mathbf{c}}_{x,i} \in \mathbb{R}^{m_x}$ and $\underline{\mathbf{c}}_{y,i} \in \mathbb{R}^{m_y}$.

$$\mathbf{C} = \begin{bmatrix} \underline{\mathbf{c}}_{x,1} & \underline{\mathbf{c}}_{x,2} & \cdots & \underline{\mathbf{c}}_{x,k} \\ \underline{\mathbf{c}}_{y,1} & \underline{\mathbf{c}}_{y,2} & \cdots & \underline{\mathbf{c}}_{y,k} \end{bmatrix} \in \mathbb{R}^{(m_x+m_y)\times k}$$

Each of the two parts of the centroid matrix can be viewed as a basis of a subspace of the respective language and because of the K-means we can say that these bases are aligned with each other.

To fold in a test document we use least squares approximation to find their appropriate place in the subspace. If a document $\underline{\mathbf{x}}$ from language L_x has to be folded in one can get its coordinates in the vector space spanned by $\mathbf{C}_x = [\underline{\mathbf{c}}_{x,1}, \cdots, \underline{\mathbf{c}}_{x,k}]$ using the pseudoinverse.

$$\hat{\underline{\mathbf{x}}} = (\mathbf{C}_x^T \mathbf{C}_x)^{-1} \mathbf{C}_x^T \underline{\mathbf{x}} \in \mathbb{R}^k$$

With this formula one gets the coefficients of the linear combination of the basis vectors that outputs the closest vector in the span of $\mathbf{C}_i$ thus this is a language independent representation. To compare two documents one has to compute the cosine similarities of these representations.

3.2 Proposed Models

The proposed models revolve around the same idea as the baselines (and as every other inter-lingual representation based method in the literature); that a training document in one language should contain the same concepts, same topics as their parallel pair in the other language. Therefore it is reasonable to assume that the training documents span a similar vector space in their respective language spaces.

One of the things that distorts this is that translations of texts (especially books) are rarely word-to-word translations. Some words in one language do not have a translation or even a meaning in the other and because of this translators can creatively use their vocabulary to describe a text in the target language. Two sentences with the exact same meaning can be translated without having a single common word by using synonyms.

Latent Semantic Indexing has been proven to be effective in the tackling the challenges presented by synonymic and polysemic words while reducing dimensionality and increasing performance of models. For these reasons our models use LSI to solve the document linking task in an effective and efficient way.

3.2.1 Reference similarity model

The least complex model that has been tested assumes that when two given the training documents the cosine-similarity of two relevant documents' to a training document will also be similar since they have to contain the same topics and concepts.

In mathematical terms let d_1, d_2 be two documents of which we have the translations in two languages with vector representations $\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2 \in \mathbb{R}^{m_x}$ in language L_x and $\underline{\mathbf{y}}_1, \underline{\mathbf{y}}_2 \in \mathbb{R}^{m_y}$ in language L_y. The assumption is that their cosine similarities are approximately equal:

$$\text{cos-sim}(\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2) \approx \text{cos-sim}(\underline{\mathbf{y}}_1, \underline{\mathbf{y}}_2)$$

Extending this assumption to the general case, if $\underline{\mathbf{x}} \in \mathbb{R}^{m_x}$ and $\underline{\mathbf{y}} \in \mathbb{R}^{m_y}$ are the same documents representation in the two languages, then their cosine similarities to each of the training documents should be close. So for $j = 1, \ldots, n$:

$$\text{cos-sim}(\underline{\mathbf{x}}, \underline{\mathbf{x}}_j) \approx \text{cos-sim}(\underline{\mathbf{y}}, \underline{\mathbf{y}}_j)$$

where $\underline{\mathbf{x}}_j$ is the j-th training document-vector of language L_x.

The assumption then is that by forming a vector from the cosine similarities of the test document and each training documents the vectors will be similar. Denote the column-normalized term-document matrix of language L_x by $\mathbf{X} \in \mathbb{R}^{m_x \times n}$ and the normalized test document vector by $\underline{\mathbf{x}} \in \mathbb{R}^{m_x}$ and similarly of L_y by $\mathbf{Y} \in \mathbb{R}^{m_y \times n}$ and $\underline{\mathbf{y}} \in \mathbb{R}^{m_y}$.

$$\mathbf{X}^T \underline{\mathbf{x}} \approx \mathbf{Y}^T \underline{\mathbf{y}}$$

where by definition $\mathbf{X}^T \underline{\mathbf{x}} \in \mathbb{R}^n$ is the cosine-similarity vector of $\underline{\mathbf{x}}$ to the training documents of language L_x.

If this is the case then the monolingual cosine-similarities to the training documents are a good language independent representation of the testing documents. In many cases LSI has been proven to be effective in increasing the cosine-similarity of similar monolingual documents while reducing the dimensionality of the data and the noise, therefore we use it as a preprocessing step for every document[13].

The model can be summarized as follows. Let $\mathbf{X} \in \mathbb{R}^{m_x \times n}$ and $\mathbf{Y} \in \mathbb{R}^{m_y \times n}$ be our training term-document matrices in language L_x and L_y.

1. Use latent semantic indexing to reduce the dimensionality of the space and create the document-concept vectors and matrices.

   $$\mathbf{X}_{k_x} = \mathbf{U}_{k_x} \mathbf{\Sigma}_{k_x} \mathbf{V}_{k_x}^T$$

   where $\mathbf{U}_{k_x} \in \mathbb{R}^{m_x \times k_x}$ is the term-concept matrix, $\mathbf{V}_{k_x} \in \mathbb{R}^{n \times k_x}$ is the document-concept matrix and $\mathbf{\Sigma}_{k_x} \in \mathbb{R}^{k_x \times k_x}$ are the concept weights.

   The training documents in the latent semantic space are now represented as a concept-document matrix

   $$\tilde{\mathbf{X}} = \mathbf{\Sigma}_{k_x} \mathbf{V}_{k_x}^T \in \mathbb{R}^{k_x \times n}.$$

   And a test document $\underline{\mathbf{x}} \in \mathbb{R}^{m_x}$ as a concept vector

   $$\underline{\tilde{\mathbf{x}}} = \mathbf{U}_{k_x}^T \underline{\mathbf{x}} \in \mathbb{R}^{k_x}.$$

2. For every document $\underline{\tilde{\mathbf{x}}}$ calculate its cosine similarities to the training documents and construct its language independent cosine-similarity vector from those.

   $$\underline{\hat{\mathbf{x}}} = \tilde{\mathbf{X}}^T \underline{\tilde{\mathbf{x}}} \in \mathbb{R}^n$$

3. Now it is possible to compare two documents in different languages $d_x \in L_x$ and $d_y \in L_y$ by calculating the cosine-similarity of their language independent representation:

   $$\text{cos-sim}(\underline{\hat{\mathbf{x}}}, \underline{\hat{\mathbf{y}}})$$

Note that in Step 2. the concept-document matrix and vector has to be L_2 normalized before the product.

3.2.2 Coefficient approximation

This model tries to directly embed the test documents into the space spanned by the training documents in the semantic space using linear least squares. The assumption is, that the vector space spanned by the parallel training documents is the same in

their respective language. Therefore the coordinates of the test documents in that span would be a good language independent representation.

Let $\mathbf{X} \in \mathbb{R}^{k_x \times n}$ and $\mathbf{Y} \in \mathbb{R}^{k_y \times n}$ be the LSA reduced training documents. The columns of $\mathbf{X}$ (the documents) span a subspace over the concepts of L_X which means that every test document $\underline{\mathbf{d}} \in \mathbb{R}^{k_x}$ is either the linear combination of the columns (if $\underline{\mathbf{d}}$ by chance lies on the subspace) or can be approximated by it.

We want to find those $\beta_1, \ldots, \beta_n \in \mathbb{R}$ coefficients for which

$$\underline{\mathbf{d}} \approx \beta_1 \underline{\mathbf{x}}_1 + \cdots + \beta_n \underline{\mathbf{x}}_n$$

where $\underline{\mathbf{x_j}}$ is the j-th column (document) of $\mathbf{X}^{(i)}$.

Or in other words we want to minimize the squared error

$$||\mathbf{X}\underline{\beta} - \underline{\mathbf{d}}||_2^2$$

Since from the LSI we know that $k_x < n$, this is an underdetermined linear system. From Chapter 2 we know that the solution of this problem is can be achieved by using the right pseudoinverse:

$$\underline{\beta}^* = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\underline{\mathbf{d}} \in \mathbb{R}^n$$

that is going to be the coordinates of $\underline{\mathbf{d}}^*$.

To sum up the model denote the right pseudoinverse of $\mathbf{X}$ by $\mathbf{P}_X \in \mathbb{R}^{n \times k_X}$ and of $\mathbf{Y}$ by $\mathbf{P}_Y \in \mathbb{R}^{n \times k_Y}$. For documents $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ their language independent representation is:

$$\underline{\hat{\mathbf{x}}} = \mathbf{P}_x \underline{\mathbf{x}} \in \mathbb{R}^n$$

$$\underline{\hat{\mathbf{y}}} = \mathbf{P}_y \underline{\mathbf{y}} \in \mathbb{R}^n$$

The similarity between to documents is again the cosine similarity of the language independent representations.

$$\text{cos-sim}(\underline{\hat{\mathbf{x}}}, \underline{\hat{\mathbf{y}}})$$

3.2.3 Linear concept approximation

The problem with LSI in multilingual environments is that if we reduce the term-document matrices independently the output dimensions of the decomposition are not guaranteed to be the same. Although their will probably be some resemblance because of the language differences the extracted concepts will not be equivalent. And cosine similarity is quite sensitive to this difference.

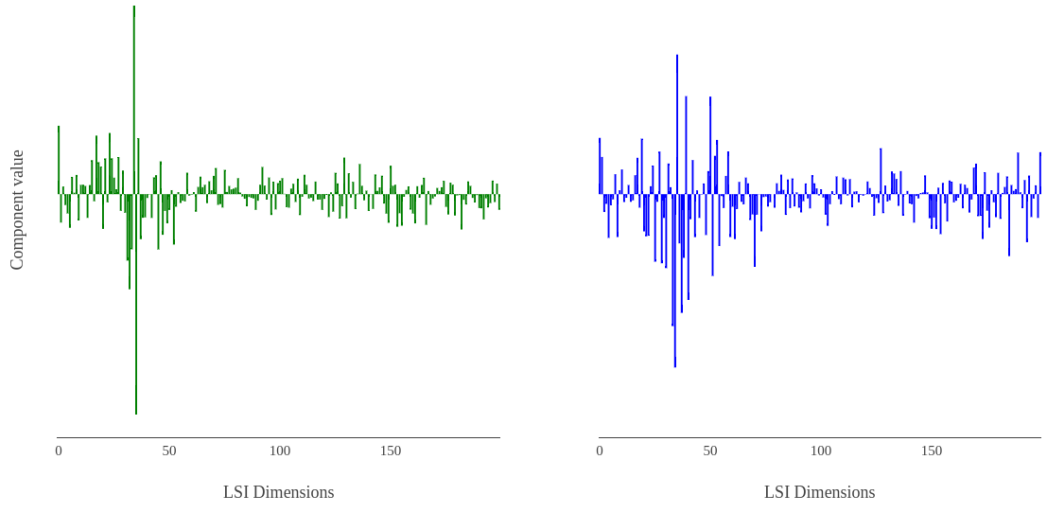

Figure 3.1: Latent semantic difference between documents. Left: Hungarian document. Right: English document.

In Figure 3.1 we can observe the difference between the first 400 latent concepts of a document in Hungarian and English. Although the document is the same this does not guarantee us that the first concept in Hungarian will be the first concept in English or even if there will be a pair for each concept. But the intuition is that because the two documents are about the same topics there must be some connection between them.

The following model is built on the assumption that this connection is linear and that every latent semantic dimension in one language can be approximated by the linear combinations of the latent semantic dimensions of the other language.

Linear approximation of concept spaces

Let $\mathbf{X} \in \mathbb{R}^{k_x \times n}$ and $\mathbf{Y} \in \mathbb{R}^{k_y \times n}$ be our LSI reduced concept-document matrices of languages L_x and L_y. In this representation every column of these matrices represent a document, which is not beneficial for the sake of the argument, so we denote their transposed document-concept matrices as $\mathbf{C}_x = \mathbf{X}^T \in \mathbb{R}^{n \times k_x}$ and $\mathbf{C}_y = \mathbf{Y}^T \in \mathbb{R}^{n \times k_y}$. In these matrices every column is a concept in their respective languages.

$$C_x = [\underline{\mathbf{c}}_{x,1}, \ldots, \underline{\mathbf{c}}_{x,k_x}]$$
$$C_y = [\underline{\mathbf{c}}_{y,1}, \ldots, \underline{\mathbf{c}}_{y,k_y}]$$

Our assumption is that a concept in L_x can be approximated fairly well by the linear combination of the concepts in L_y. This means that for every column $\underline{\mathbf{c}}_{x,i}$ of $\mathbf{C}_x$ there are $\beta_{1,i}, \ldots, \beta_{k_y,i} \in \mathbb{R}$ coefficients for which:

$$\underline{\mathbf{c}}_{x,i} \approx \sum_{j=1}^{k_y} \beta_{j,i} \underline{\mathbf{c}}_{y,j}.$$

Or by defining the coefficient vector of $\underline{\beta}_i = [\beta_{1,i}, \ldots, \beta_{k_y,i}]^T \in \mathbb{R}^{k_y}$:

$$\underline{\mathbf{c}}_{x,i} \approx \mathbf{C}_y \underline{\beta}_i$$

For this is assumed to be true for every $\underline{\mathbf{c}}_{x,i}$ $(i = 1, \ldots, k_x)$ concept of L_x we can create a $B \in \mathbb{R}^{k_y \times k_x}$ coefficient matrix with columns $\underline{\beta}_i$.

$$\mathbf{B} = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,k_x} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,k_x} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{k_y,1} & \beta_{k_y,2} & \cdots & \beta_{k_y,k_x} \end{bmatrix}$$

Using this matrix by our assumption we can linearly approximate the $\mathbf{C}_x$ concept matrix.

$$\mathbf{C}_x \approx \mathbf{C}_y \mathbf{B}$$

Returning to the problem of only one concept at a time, the equation

$$\underline{\mathbf{c}}_{x,i} = \mathbf{C}_y \underline{\beta}_i$$

is a linear system with the matrix $\mathbf{C}_y$ having n rows and k_y columns. Since we know that LSI is limited to the rank of the matrix we can safely assume that $k_y < n$ so this is an overdetermined system. From Chapter 2 we know that such system does not have an exact solution but we can determine the coefficients $\beta_{j,i}$ so that the resulting vector best approximates $\underline{\mathbf{c}}_{x,i}$ in a least squared sense.

In other words we are looking for the $\underline{\beta}_i$ vector that minimizes the squared error:

$$||\mathbf{C}_y \underline{\beta}_i - \underline{\mathbf{c}}_{x,i}||_2^2$$

In Chapter 3 it has been shown that this optimization problem can be solved by solving the normal equation of which can be solved by using the pseudoinverse:

$$\underline{\beta}_i^* = (\mathbf{C}_y^T \mathbf{C}_y)^{-1} \mathbf{C}_y^T \underline{\mathbf{c}_{\mathbf{x},\mathbf{i}}}.$$

Denoting the pseudoinverse by $\mathbf{P}_{\mathbf{C}_y} = ({\mathbf{C}_y}^T \mathbf{C}_y)^{-1} \mathbf{C}_y^T \in \mathbb{R}^{k_y \times n}$ we get the much simpler expression

$$\underline{\beta}_i^* = \mathbf{P}_{\mathbf{C}_y} \underline{\mathbf{c}_{\mathbf{x},\mathbf{i}}}.$$

For this is true for all $i = 1, \ldots, k_x$ concepts we can write the whole B matrix as

$$\mathbf{B} = \mathbf{P}_{\mathbf{C}_y} \mathbf{C}_x \in \mathbb{R}^{k_y \times k_x}.$$

Using this $\mathbf{B}$ we get the best linear approximation of the concepts $\mathbf{C}_x$ by the concepts of $\mathbf{C}_y$:

$$\hat{\mathbf{C}}_y = \mathbf{C}_y \mathbf{B}$$

.

This means that we can transform the k_y concept in the concept space of L_y into a linear approximation of the concepts of L_x. Therefore the transformed documents of L_y will also be more close to their counterparts. By transposing the previous equation we get the representation of the documents of L_y what we are looking for.

$$\hat{\mathbf{Y}} = \mathbf{B}^T\mathbf{Y} \in \mathbb{R}^{k_x \times n}.$$

And for a test document $\underline{\mathbf{y}} \in \mathbb{R}^{k_y}$:

$$\hat{\underline{\mathbf{y}}} = \mathbf{B}^T\underline{\mathbf{y}} \in \mathbb{R}^{k_x}$$

The similarity of documents $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ are:

$$\text{sim}_1(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \text{cos-sim}(\underline{\mathbf{x}}, \hat{\underline{\mathbf{y}}})$$

Since we have two languages we can do two separate concept approximations: one from L_x to L_y and another the roles switched. The first approximates the $\mathbf{C_x}$ concepts in the span of $\mathbf{C_y}$ and the second the other way around. This way every test document will have two representations and two similarities from each other. The final similarity will be the sum or the average of the two.

The same way as we defined $\mathbf{B}$ and $\hat{\mathbf{Y}}$ for L_y we can define an analogous to them for L_x the following way:

$$\mathbf{A} = \mathbf{P}_{\mathbf{C}_x}\mathbf{C_y} \in \mathbb{R}^{k_x \times k_y}$$

This $\mathbf{A}$ matrix then transforms the k_y concept-vectors of L_y to the span of $\mathbf{C}_x$. The new representation of $\mathbf{X}$ is then:

$$\hat{\mathbf{X}} = \mathbf{A}^T\mathbf{X} \in \mathbb{R}^{k_y \times n}$$

The representation of document $\underline{\mathbf{x}}$ is:

$$\hat{\underline{\mathbf{x}}} = \mathbf{A}^T\underline{\mathbf{x}} \in \mathbb{R}^{k_y}$$

The second similarity is analogue to the first one:

$$\text{sim}_2(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \text{cos-sim}(\hat{\underline{\mathbf{x}}}, \underline{\mathbf{y}})$$

And the final similarity between two documents is the average of these two:

$$\text{sim}(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \frac{\text{sim}_1(\underline{\mathbf{x}}, \underline{\mathbf{y}}) + \text{sim}_2(\underline{\mathbf{x}}, \underline{\mathbf{y}})}{2}$$

Interpretation of the transformation

The columns of the document-concept matrices $\mathbf{C}_x$ and $\mathbf{C}_y$ are the concepts that span two different subspaces relative to the training documents, which are common in the two languages. By the column interpretation of the least squares method what the transformation does is that it projects the concepts of L_x to the subspace spanned by $\mathbf{C_y}$. This is shown in Figure 3.2.

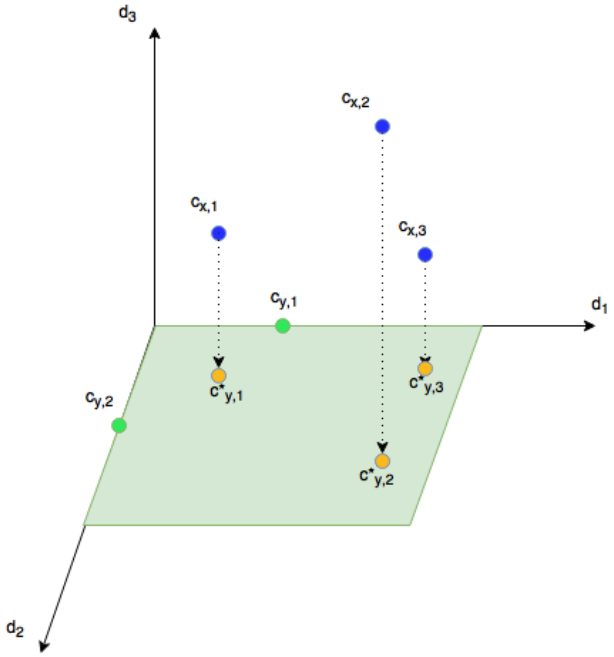

Figure 3.2: Least squares projection of the concepts of L_x to $\mathbf{C}_y$

By doing this what we are achieving is that the new concepts of L_y are going to be the best approximations of the concepts of L_x. Therefore the documents of L_y represented with the new concepts are going to be closer to their counterparts in the other language.

Since the documents are built up of concepts this brings the relevant documents closer also. In the concept-document view of the matrix the axes are the concepts and the documents are the points in the space stretched by them. By approximating the concepts the approximation of the documents is an adopted side effect of the transformation that we use.

The effect of the transformation

To have sense about what this transformation does Figure 3.3 shows a documents latent semantic vectors in English and Hungarian and in the transformed Hungarian. It can be clearly seen that by transforming the concepts of one language to the other the most relevant features are very well connected and thus their cosine similarity will be high.

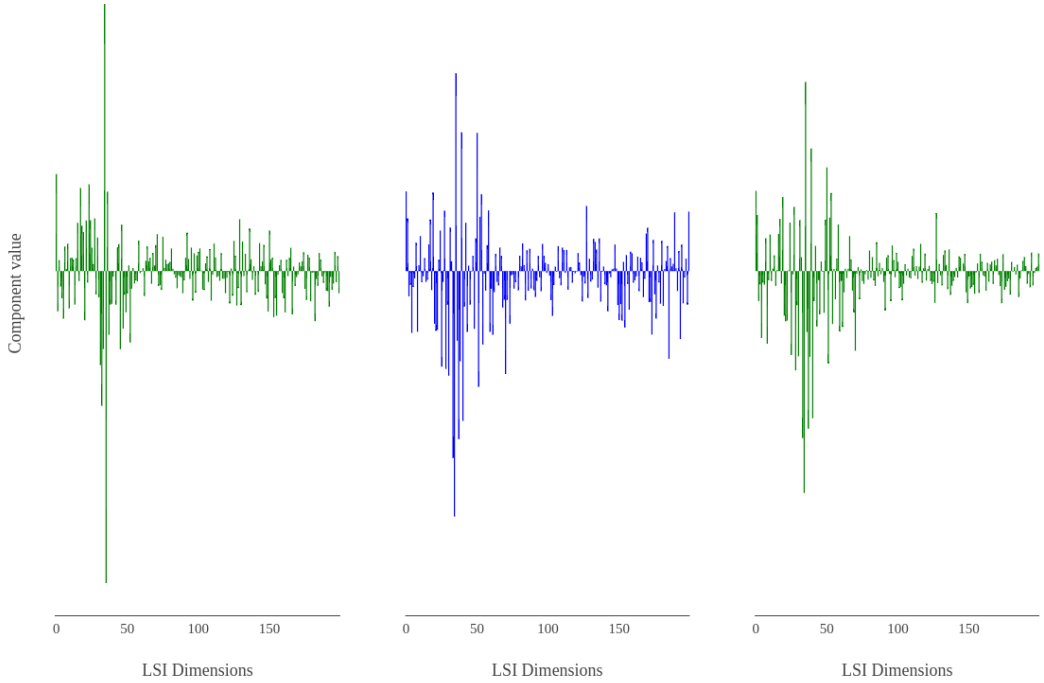

Figure 3.3: Impact of the transformation. Left: original Hungarian document, middle: original English, right: transformed Hungarian.

Another way to show the effectiveness of these transformations is to plot the first two components of a few documents in the original and the transformed semantic spaces. Figure 3.4 shows this.

Looking at the figure we see that the original representation of the documents had little in common. The documents are scattered over the space spanned by the first two semantic dimensions. The biggest issue is that the closest parallel documents are never their pairs. After the transformation this issue is resolved. The documents and their parallel pairs are nicely clustered together and closer to each other than to any other document. This means that they are alike in both of their semantic dimensions.

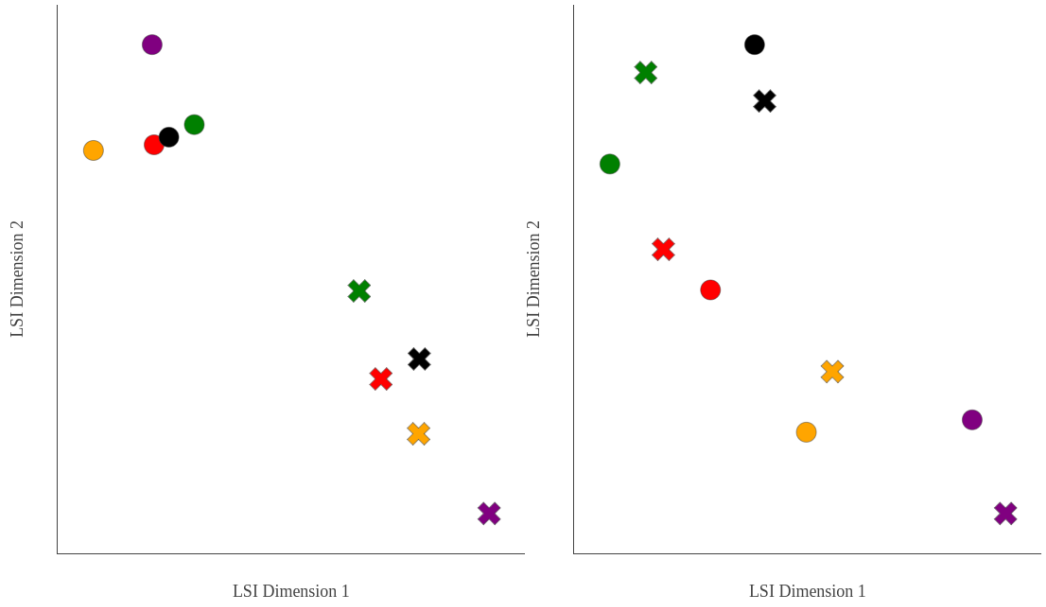

Figure 3.4: Effect of the transformation on the documents in the first two latent semantic dimensions. The colors represent different documents, the "X"-s are the English and the circles are the Hungarian representations. Left - the original documents. Right - the transformed documents.

Algorithm

In the formulas above I introduced a few notations and transpositions that were only for the sake of the presentation and are not needed in the final model. I leave these out here.

The whole model can be summarized as follows. I am assuming that the term-document matrices (and vectors for test documents) were already reduced by LSI to concept-document ones. Let $\mathbf{X} \in \mathbb{R}^{k_x \times n}$ be the concept-document matrix of L_x and $\mathbf{Y} \in \mathbb{R}^{k_y \times n}$ of L_y.

1. Construct the following coefficient matrices for the projections:

$$\mathbf{A} = \mathbf{P}_{\mathbf{X}^T}\mathbf{Y}^T \in \mathbb{R}^{k_x \times k_y}$$
$$\mathbf{B} = \mathbf{P}_{\mathbf{Y}^T}\mathbf{X}^T \in \mathbb{R}^{k_y \times k_x}$$

2. For (LSI reduced) document vectors $\underline{\mathbf{x}} \in \mathbb{R}^{k_x}$ from language L_x and $\underline{\mathbf{y}} \in \mathbb{R}^{k_y}$

from language L_y calculate:

$$\hat{\underline{\mathbf{x}}} = \mathbf{A}^T \underline{\mathbf{x}} \in \mathbb{R}^{k_y}$$
$$\hat{\underline{\mathbf{y}}} = \mathbf{B}^T \underline{\mathbf{y}} \in \mathbb{R}^{k_x}$$

3. Compute the cosine similarities:

$$\text{sim}_1(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \text{cos-sim}(\hat{\underline{\mathbf{x}}}, \underline{\mathbf{y}})$$
$$\text{sim}_2(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \text{cos-sim}(\underline{\mathbf{x}}, \hat{\underline{\mathbf{y}}})$$

4. The final similarity is the average of these two:

$$\text{sim}(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \frac{\text{sim}_1(\underline{\mathbf{x}}, \underline{\mathbf{y}}) + \text{sim}_2(\underline{\mathbf{x}}, \underline{\mathbf{y}})}{2}$$

Chapter 4

Experiments

In this chapter I will describe the experiments conducted to evaluate the models. The first section describes the datasets used and their sources. The second describes the evaluation measures which we need in order to be able to compare the models. In the third section I discuss the details about the implementation of the models described in Chapter 3. And the last section will detail the results and the judgment of the models.

4.1 Datasets

To conduct experiments we first need something to run our models on: data. Data in our case has to be parallel documents in multiple languages. This is much harder to come by than regular documents for other text mining applications such as text categorization or clustering.

Before choosing documents we have to choose languages, which is an important decision. As it will be shown in the results section some languages are much more similar than the others and one can achieve great performance using them. Therefore we picked the languages; two of which are very similar: French and English, and one that is very different from those: Hungarian. This lets us have three different scenarios for the three different pairings: Hungarian-English, Hungarian-French, and English-French.

The largest source of parallel documents are international organizations and governments that are obliged to translate their documents to multiple languages. One

of these organizations is the European Union where every piece of legislation and official document has to be translated to 27 languages. These professional translations are often used for cross-lingual and multilingual tasks such as machine translation or in our case cross-lingual document linking because of their accessibility, quantity and quality. Two of the datasets are from these document collections[21].

Although these official documents are good for the purpose of general cross-lingual information retrieval tasks, they are usually not book-length or book-like. Unfortunately as of writing there are no available multilingual corpus of books that has the necessary quantity of parallel pairs for the purpose of this research. So for the purpose of assessing our models in real-life situations I have constructed a our own dataset of English and Hungarian parallel novels from the classic literature.

4.1.1 JRC-Acquis corpus

The JRC-Acquis corpus consists of the body of the European Unions law and legislation papers. It was put together by an international group of researchers for the purpose of making the largest multilingual corpora freely available at the time. It contains documents from 1950 until the present and it is updated every few years as new legislations are passed in the EU. The documents are not only parallel but also sentence aligned. Therefore the corpus is mainly used for machine translation projects but because of its size and number of languages contained it is a popular dataset for every kind of text mining application[20].

The corpus has a total of over 4 million documents with an average of over 18,000 parallel documents per language-pair. We are interested in those that are available in both English, French and Hungarian. In the current version of the corpus there are a little under 20,000 such documents. Since text mining is a generally computationally heavy task this amount is too much for our experimentation. Therefore 2,000 documents were sampled at random of which 1,000 is used for training and 1,000 for testing (the split is also random).

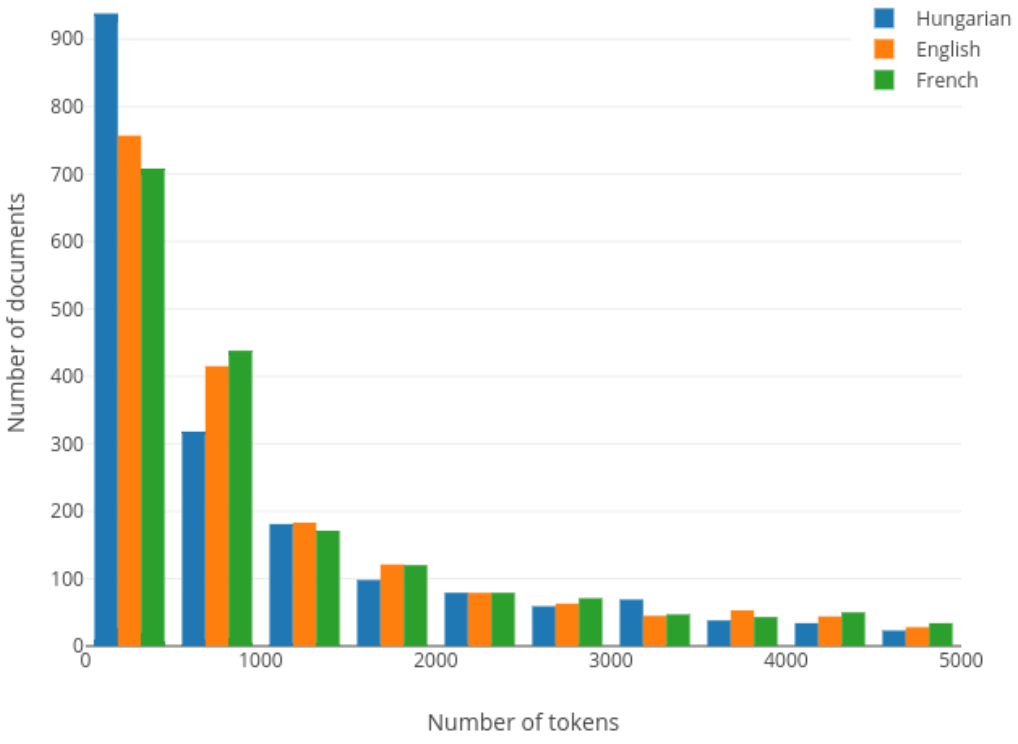

Figure 4.1: Histogram of tokens in the JRC-Acquis sample dataset

4.1.2 EU Bookshop dataset

The EU Bookshop is an online service and archive of publications from various European Union institutions. Many of these have been translated to multiple languages of the EU. In 2014 researchers crawled the website to extract these documents and create the EUBookshop corpus for machine translation purposes[19].

The corpus has over 40,000 individual documents, most of them available in English and French. The intersection of the documents available in our three target languages English, French and Hungarian accounts to almost 1,000 documents. Of these the ones that were excessively long or short were ignored and the left 860 documents were split in two randomly to a training and test set.

Because of the length of the documents in it (as shown on Figure 4.2) this corpus plays a middleground role between the JRC-Acquis and the novels.

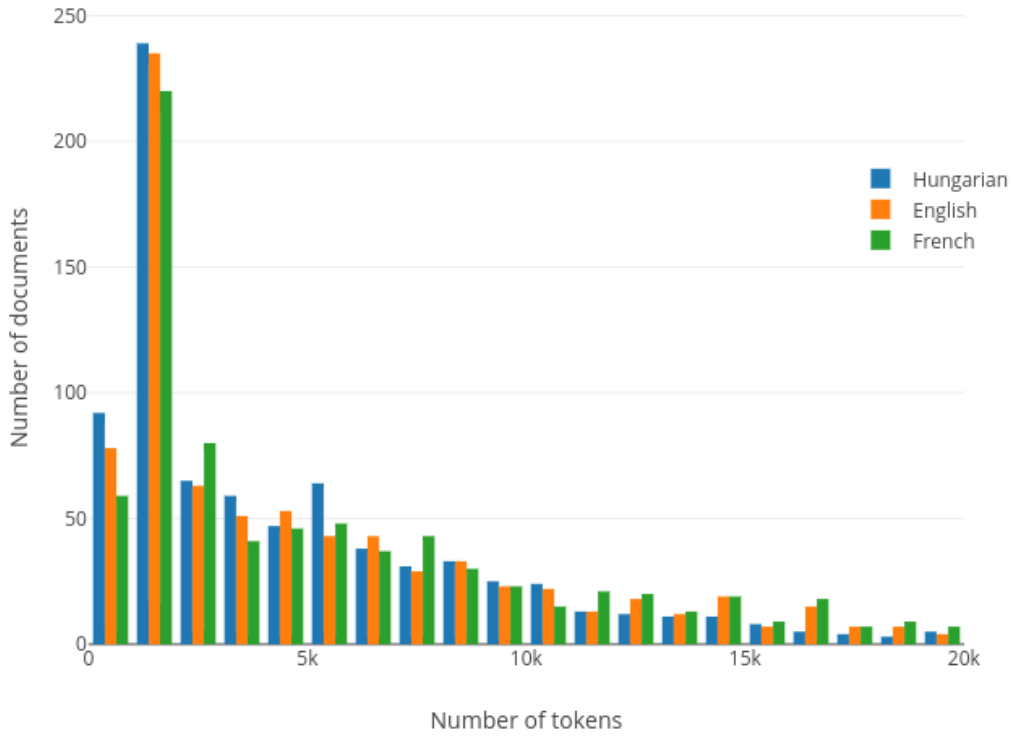

Figure 4.2: Histogram of tokens in the EUBookshop dataset

4.1.3 Novels dataset

As of writing this document only one multilingual book corpus is available freely for research purposes. The dataset collected by Andras Farkas consists of 64 books in a span of 16 languages, but the number of parallel pairs are very few. There are only 28 books available in both English and Hungarian which is too small amount to do experiments on. Therefore I collected 100 mostly classic, freely accessible books in English and Hungarian from various online archives such as the Hungarian Electronic Library and Project Guttenberg. Most of the books were not readily available in text format so first they were converted by Calibre which is an open source e-book manager software. After conversion those that were not had to be encoded in UTF-8 to avoid future problems with the different encodings.

The books were chosen to cover all kinds of genres from horror to romance. The writers are well known XIX. and XX. century authors and the titles are some of their most popular works. The resulting corpus has a minimum number of tokens of 5,336, a maximum of 316,561 with an average of 87,516.38 and a median of 67,364 in Hungarian. Those same numbers for English are 6,964 and 418,042 with an average of 113,923.89 and a median of 96,950. The histogram of the corpus is shown on Figure 4.3.

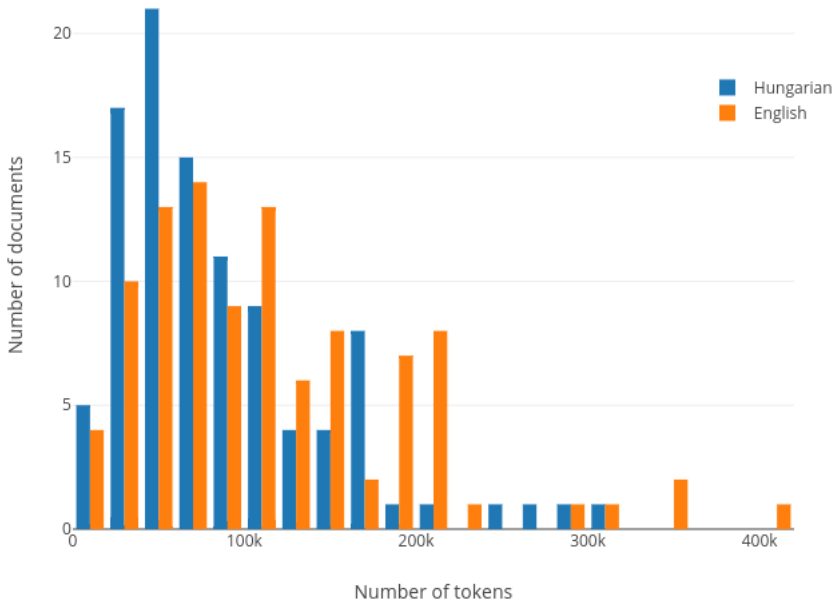

Figure 4.3: Histogram of tokens in the JRC-Acquis sample dataset

4.2 Performance measures

Every experiment needs metrics to measure the performance of the different approaches. These metrics differ in each subject and field but they always describe our view of what we look at as "good" performance.

In the field of information retrieval the most well known metrics are the recall and the precision. These are the metrics that measure the goodness of a search engine. Given a query and a set of target documents the precision of the system is the proportion of retrieved documents that are relevant to the query. Recall on the other hand is the proportion of relevant documents that are retrieved. The relevancy is usually a predefined phenomena that is given by hand or by some heuristic such as labels or tags in documents.

In the field of cross-lingual document recommendation there are rarely predefined relevancy. We do not know how relevant two documents are even if they are in the same language not to mention if they are not. Therefore researchers of this subject have come up with a number of metrics for this task of which two I am going to use to assess the performance of the baseline and the proposed models[10].

4.2.1 Mate retrieval rate

The only document that we are absolutely certain is relevant is the pair of the document in the other language. Therefore the mainly used metric in cross-lingual information retrieval tasks is the mate retrieval rate which is the proportion of test documents that are linked with their cross language pair. This can be formulated as the number of rows/columns in the test similarity matrix, where the diagonal element is the largest.

4.2.2 Mean reciprocal rank

This performance measure is commonly used in tasks where there is only one relevant document to retrieve. Define the rank of document i deonoted by r_i as the order in which its parallel pair is retrieved. The mean reciprocal rank is then:

$$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \frac{1}{r_i}$$

4.3 Implementation details

4.3.1 Packages used

The models and the data processing were implemented in Python 3.6. Python is the most popular programming language of scientific computing and machine learning because of it ease to use and abundant number of packages for the task. Libraries such as Scikit-Learn, NumPy, SciPy, or Pandas offer open source implementations of many tasks and algorithms related to these areas and the large community around them.

NumPy and SciPy are by far the most fundamental packages used in scientific Python. NumPy is the implementation of everything that is matrix or vector related. It offers an interface similar to MatLab which makes it easy and intuitive to use while most of the functions are implemented externally in C/C++ which makes it very fast and efficient. SciPy is the package of more sophisticated algorithms related to fields like signal processing and statistics but it also offers a sparse matrix implementation and it is fully compatible with NumPy. In our use-case the term-document matrices are generally huge with tens of thousands of dimensions but most part of them are zeros. Therefore using sparse matrices which are not only capable of storing data efficiently but also to do operations on them, is crucial in our application[22, 11].

Scikit-Learn is an open-source machine learning (ML) library maintained by the booming ML community of Python. It is built upon NumPy and SciPy and implements hundreds of algorithms for the whole range of tasks of a Machine Learning engineer. From preprocessing to cross validation to the most popular classification and clustering algorithms everything can be found in it. In this case its preprocessing methods and K-Means clustering were used[12].

GenSim is a Natural Language Processing and topic modeling toolkit that the author claims to be the most robust and efficient one available in Python and the community tend to agree. GenSim has very efficient state-of-the-art implementations of various algorithms related to text mining, such as TF-IDF, Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) and is able to process these tasks in a distributed way. Since we are dealing with huge documents GenSim was chosen to be the implementation of TF-IDF and LSI[14].

Plotly is a relatively recent web-based data visualization tool with an API in many languages such as Javascript, MatLab and Python. It can be used to create any kind of plots from NumPy arrays with great flexibility and save them in the cloud for further investigation and editing. Plotly is the tool that I have used to do data exploration and plot creating for the results and other figures in this document[4].

4.3.2 Preparing the data

After loading in the documents for the task the preprocessing stage begins. First we have to split the documents in each language to a training and a testing set. Of course these have to be aligned so the training set in English contains the parallel pairs of the training sets in French and Hungarian. A train-test split of 50%-50% is used here which means that in the JRC-Acquis dataset there are 1000 train and 1000 test documents, in the EUBookshop 430-430 and in the Novels corpus 50-50.

Then the documents (text files) have to be tokenized to represent each of them as list of tokens. There are a number of decisions to be made here. First the definition of what we call token. In text documents there can be many kinds of continuous sequence of characters not all of those are useful for us. For example numbers and punctuation are not really meaningful, they do not contribute to any topic on concept. So in the first step punctuation and numbers were removed along with any kind of tokens that contain non-alphabetic characters except hyphen.

From the list of tokens those that occur in too many or in too few documents have been filtered. This is a very popular way of reducing the dimensionality of the data, as the number of tokens that are only in a few documents is large. And since they only occur in so many documents they only contribute to those, but because of the way tf-idf weighting works the dimensions that these words construct will have a huge effect in these documents. This is a good thing if they are relevant because in that sense they will be close to each other, but not if those words are not actually important. Tokens that are present in too many documents are basically just noise and stopwords. Since they are everywhere they do not contribute much to the concepts either. The parameters that control this are denoted as $\min_{\text{df}}$ and $\max_{\text{df}}$. After data exploration the parameters were handpicked for each dataset to provide a reasonable number of tokens in each language shown in Table 4.1.

Dataset	$\min_{df}$	$\max_{df}$	Language	Tokens
JRC-Acquis	0.5%	80%	Hungarian	16,000
			English	7,300
			French	9,500
EUBookshop	2%	80%	Hungarian	16,600
			English	8,100
			French	10,400
Novels	10%	80%	Hungarian	41,400
			English	18,200

Table 4.1: Token parameters in the datasets

After the tokenization of both the training and the test sets we construct the TF-IDF vector of each document. We use the standard TF-IDF weighting scheme with (document) normalization. For this we can use either Scikit-Learn or GenSim, the output will be essentially the same. The inverse document frequencies are calculated on the training set only. It serves as a background population in this case. Only the words in the vocabulary (or in the training set) are going to count. Obviously the vectorization is done separately on each languages. The output of the vectorization is a training term-document matrix and a test term-document matrix. The whole process is presented on Figure 4. 4.

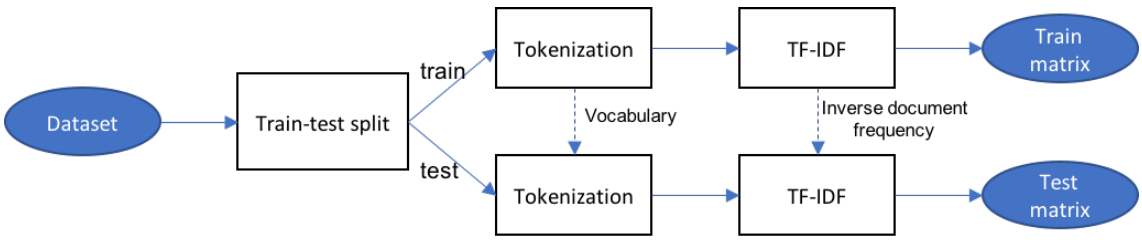

Figure 4.4: Construction process of the term-document matrices

4.4 Results

This section details the results of the conducted experiments. The implemented models get the same documents as training and testing sets in different languages from one dataset. The three language pairs (Hungarian-English, Hungarian-French and English-French) are evaluated separately. The models accept one language pair of training and test documents and return the similarity matrix of that language pair. Then the two performance measures (mate retrieval rate and mean reciprocal rank) are calculated from the similarity matrices.

All models have parameters that are dependent on the number of the training documents. For the models using latent semantic analysis it is the rank of the SVD or the number of latent semantic dimensions. Since the term-documents matrix maximal rank is limited by the number of documents it does not make sense to go over that. This is also the case with the K-means model where we cannot create more centroids than the number of documents. For this reason the results are also shown as a function of the dimensions and in the case of K-means the number of centroids to see where the results change relative to them.

4.4.1 JRC-Acquis

Figure 4.5 shows the results of the models on the JRC-Acquis dataset relative to the LSA dimensions. It is already clear that the cosine-similarity model is the weakest of them. The best results are achieved in the 400-700 range except for the K-Means for which 800 centroids seems to be the best choice.

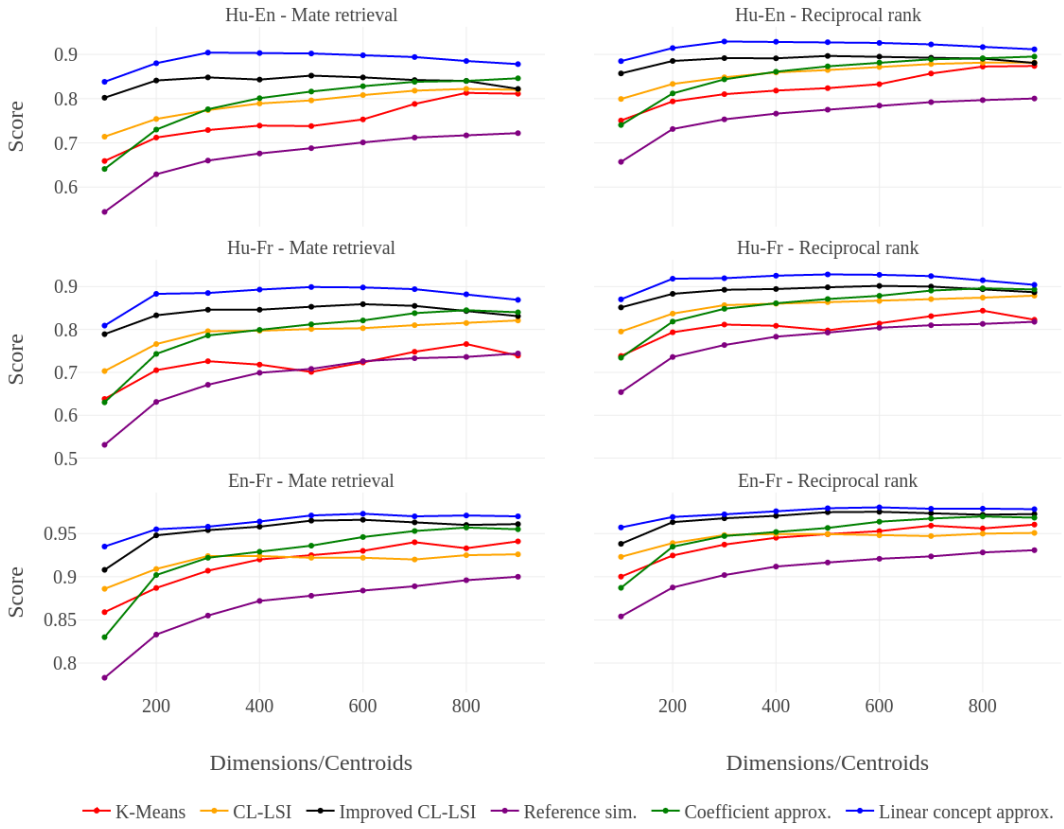

Figure 4.5: Results of the models relative to the dimensions on the JRC-Acquis corpus

On figure 4.6 the best results achieved by the models can be seen with the optimal parameters. One apparent thing is that the assumption that Hungarian is much different from the others is true. Every model achieved an average of 10% higher score on the English-French language pair than on the Hungarian-English or Hungarian-French. The least sensitive to this seems to be the Linear concept approximation (LCA) model and the most sensitive is the Reference similarity model. Reference similarity has an increasing tendency relative to to dimensions which could be because of the problem with LSI discussed in Chapter 2.

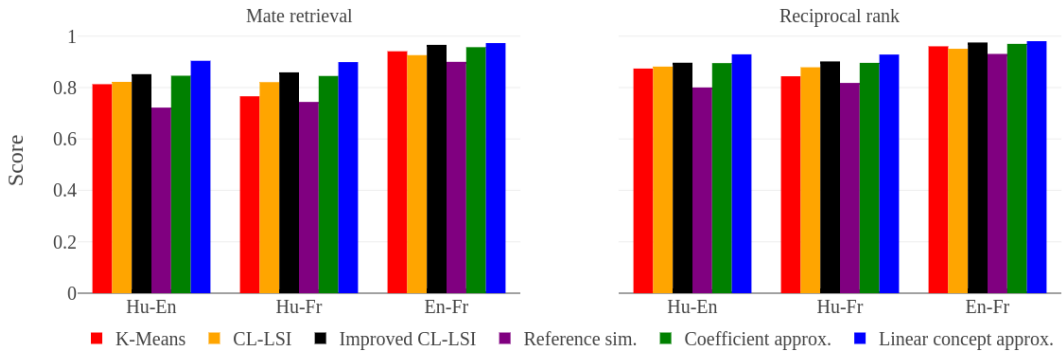

Figure 4.6: Best results of the models on the JRC-Acquis corpus

Table 4.2 shows the achieved scores on JRC-Acquis. The best model in every scenario and in every measure is the Linear concept approximation. It is particularly good in the Hungarian pairs where it has almost 5% higher score than the second best which is the improved CL-LSI. The other proposed method Coefficient approximation also shows promising results. The rest are behind them, with the worst being the Reference similarity model. K-means is remarkably sensitive to the languages in mate retrieval which could be due to the randomness of the method.

Model	Mate retrieval			Reciprocal rank		
	hu-en	**hu-fr**	**en-fr**	**hu-en**	**hu-fr**	**en-fr**
K-Means	81.3	76.6	94.1	87.38	84.38	96.04
CL-LSI	82.2	82.1	92.6	88.14	87.88	95.08
Improved CL-LSI	85.2	85.9	96.6	89.65	90.16	97.52
Reference sim.	72.2	74.4	90.0	80.02	81.79	93.08
Coefficient approx.	84.6	84.3	95.7	89.51	89.46	96.99
Linear concept approx.	**90.4**	**89.9**	**97.3**	**92.91**	**92.84**	**98.04**

Table 4.2: Results on JRC-Acquis

The experiment on JRC-Acquis showed that given enough data the proposed models are in par with the best baseline methods.

4.4.2 EUBookshop

The EUBookshop corpus has less than half the documents than the JRC-Acquis. This shows on the results, every model has lost 6-7% of its score. The trend is still the same, but the dimensions are reduced because of the size of the corpus. The scores peak in the middle of the dimension scale again, in the 200-300 region and stop improving at around the end of that. Figure 4.7 shows the results.

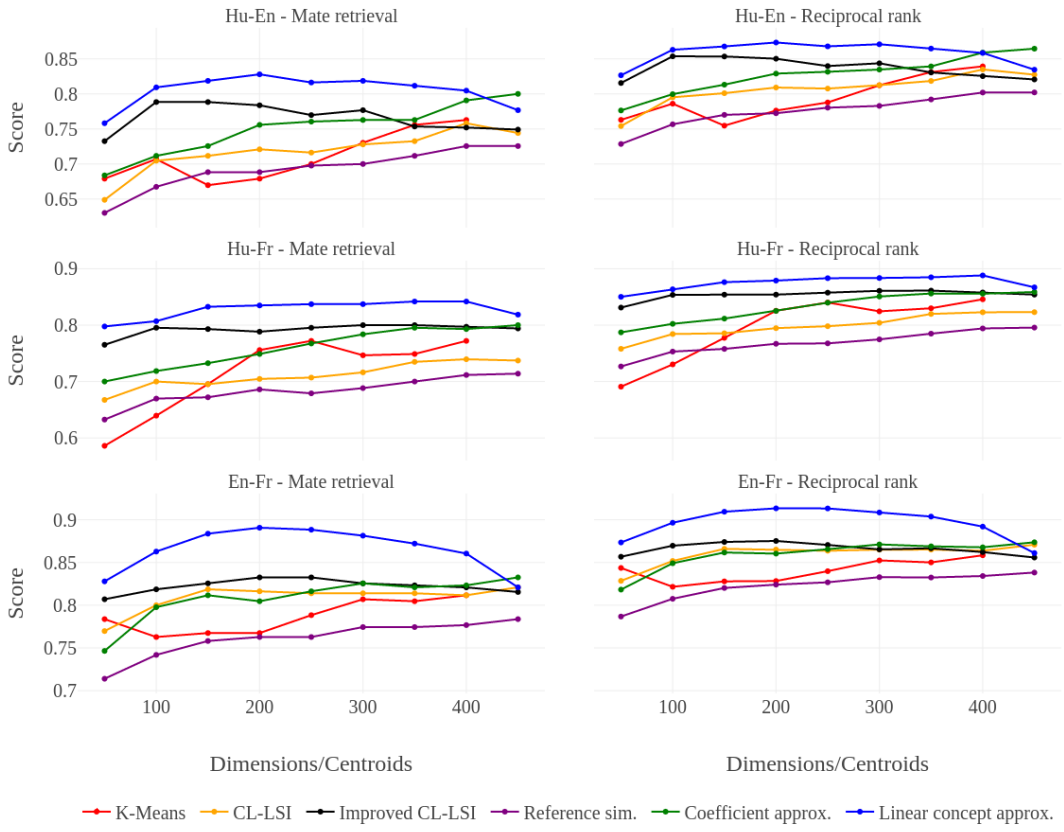

Figure 4.7: Results of the models relative to the dimensions on the EUBookshop corpus

On Figure 4.8 once again we see the effect of language dissimilarity, although this time it is less apparent. The LCA model still takes the first position yet again while the Reference similarity model has the lowest score once more, but its also less obvious now than on the first dataset. This might be due to the lengthiness of the documents in the EUBookshop to which the simple cosine-similarity model might be less sensitive. Also K-Means is now on par with the simple CL-LSI model which was not the case in the first dataset.

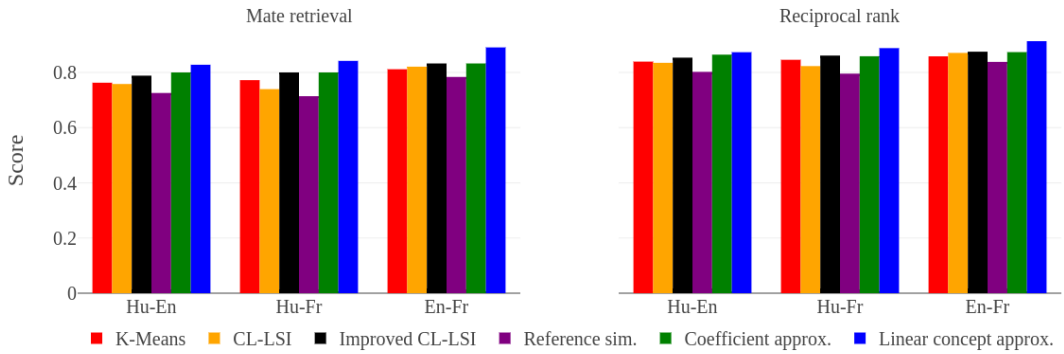

Figure 4.8: Best results of the models on the EUBookshop corpus

The results in Table 4.3 show that LCA is once again the best model by more than 4% in all scenarios. But this time the other proposed model came in at the second place. Coefficient approximation seems to be superior to the improved CL-LSI model in the Hungarian-English and English-French cases. In the Hungarian-French language pairs although their mate retrieval is equally 80%, in reciprocal rank the improved CL-LSI has 0.5% higher score, which means that although they found the best parallel pairs equally often, if they do not then CL-LSI ranks them higher.

Due to the increase in size and the less number of training documents the models lost about 5% of their accuracy in average. Exceptions from this is the Reference similarity and the K-means model which have even gained some in certain scenarios.

Model	**Mate retrieval**			**Reciprocal rank**		
	hu-en	**hu-fr**	**en-fr**	**hu-en**	**hu-fr**	**en-fr**
K-Means	76.28	77.21	81.16	83.93	84.58	85.84
CL-LSI	75.81	73.95	82.09	83.49	82.3	87.08
Improved CL-LSI	78.84	80.00	83.26	85.36	86.12	87.52
Reference sim.	72.56	71.40	78.37	80.21	79.56	83.82
Coefficient approx.	80.00	80.00	83.26	86.45	85.87	87.37
Linear concept approx.	**82.79**	**84.19**	**89.07**	**87.33**	**88.81**	**91.34**

Table 4.3: Results on EUBookshop

4.4.3 Novels

The self made Novels dataset is by far the most difficult of all the corpora. The novels are about 5 times longer with only 50 training documents and this appears on the results on Figure 4.9. The models lost about 40% from their accuracy and the rankings have changed. The best dimension number has increased to almost 80% of the document number, which is not surprising since even that is still way lower than the recommended 200-400 in the literature. But because of the lengthiness the token number is also about 3 times of the other datasets from which the LSI probably struggles to good concepts from so few training documents.

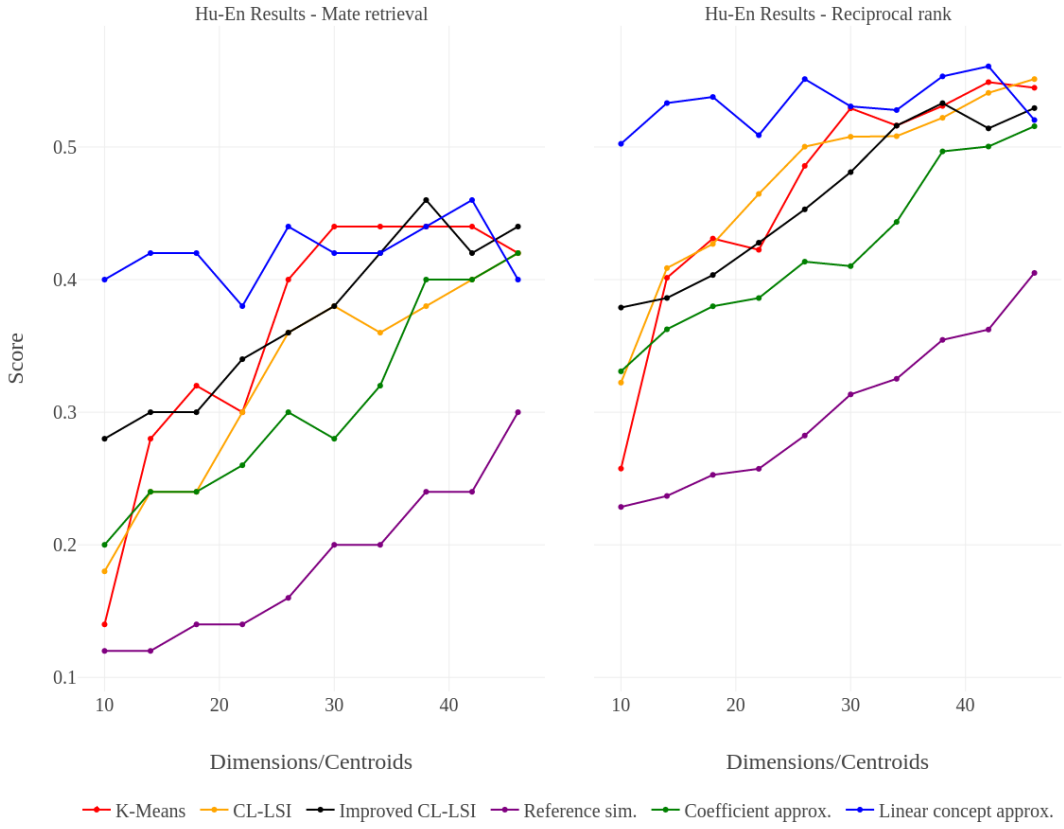

Figure 4.9: Best results of the models on the Novels corpus

In Figure 4.10 the best results are shown. Reference similarity produces the lowest scores again while the rest are about the same in 40% accuracy range. One change is that while the coefficient approximation method has fallen behind, K-means has achieved its best relative results so far both in mate retrieval and reciprocal rank. In the first two datasets it could not get even close to the improved CL-LSI but when faced with few and long documents K-means seems to be the better choice.

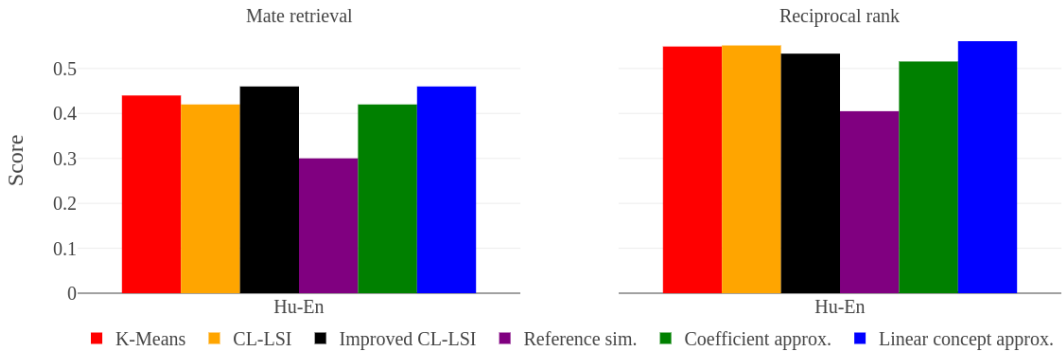

Figure 4.10: Best results of the models on the Novels corpus

Table 4.4 shows that the best method is LCA once again. Improved CL-LSI and LCA are tied in mate retrieval, but LCA has far better reciprocal rank. As I have said, although improved CL-LSI is superior to K-means in mate retrieval (by only one book), its reciprocal rank is in the lower half of the models and with as few documents as in our case this should be the deciding factor.

Model	Mate retrieval	Reciprocal rank
K-Means	44.0	54.89
CL-LSI	42.0	55.12
Improved CL-LSI	**46.0**	53.31
Reference sim.	30.0	40.5
Coefficient approx.	42. 0	51.56
Linear concept approx.	**46.0**	**56.08**

Table 4.4: Results on Novels

4.4.4 Conclusion of the results

From the conducted experiments and their results the superiority of LCA over the baseline methods is apparent. With an average of 4% increase in mate retrieval over the next best baseline method (improved CL-LSI) and 3% in reciprocal rank it is the best method that has been implemented in this thesis. Even in the case of the Novels corpus with very few training documents and lengthy texts it is superior to the other methods.

On the other proposed methods the picture is not so clear. Coefficient approximation has shown promising results with medium corpus size and length, but only managed to approach the scores of improved CL-LSI. Further investigations and experiments are needed on the idea behind it.

The reference similarity model was by far the weakest one of the examined methods. It is safe to say that in this form it is not a viable algorithm for cross-lingual document recommendation or linking.

Some conclusions can be drawn from the baseline methods too. K-means has shown great resistance to the absence of data and it can be a valuable alternative of others in situations with only a few training documents. But in cases with medium to high amounts of data it is no match to CL-LSI methods that have shown their accuracy again as many times in the literature.

Summary

The aim of the thesis was to review the problem of cross-lingual information retrieval and document recommendation and to contribute to the field with novel methods that show promising results for further investigation.

First I gave a general outline of information retrieval with a brief introduction to its challenges and how the cross-lingual scope relates to it. Then the theoretical backgrounds of the solutions were laid down in the second chapter. Three methods were detailed that form the backbone of the baseline and proposed models: linear least squares, K-means clustering and Latent Semantic Indexing. The latter was presented in greater detail for its importance to the field and the reliance of the models on it.

In the third chapter I presented three methods from the literature that are currently used to solve the cross-lingual document recommendation task: a K-means based and two cross-lingual LSI based models. These models constituted the baselines in the experiments. After these I introduced three novel methods to the field: one reference similarity based model, one that approximates the documents' combination coefficients and one that linearly approximates the concepts of a language with the concepts of another.

In the last chapter I detailed the conducted experiments to measure the performance of the proposed models and compare them to the baselines. A new dataset of novels was created and used along with two standard datasets. I utilized two performance measures from the cross-lingual recommendation field to see how the models perform. The results show that one of the models (reference similarity) has subpar performance compared to the baselines while another (coefficient approximation) achieves about equal performance with the best baseline (improved CL-LSI). The third proposed model (linear concept approximation) was seen to be superior

to the baselines in the experiments with about 3-4% higher score in both measures to improved CL-LSI.

Because these experiments were limited in size further investigations are needed of the model but the results are promising. The future of the project involves experiments on more data and comparison with other baseline and state-of-the-art models. The outcomes of those experiments are planned to be published in a scientific journal or presented in a conference relevant to the field.

Bibliography

[1] M. W. Berry, S. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *Society for Industrial and Applied Mathematics (SIAM) review*, 37(4):573–595, 1995.

[2] W. Cox and B. Pincombe. Cross - lingual latent semantic analysis. *ANZIAM Journal*, 48:1054–1074, dec 2006.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[4] P. T. Inc. Collaborative data science, 2015.

[5] A. Kontostathis and W. M. Pottenger. A framework for understanding Latent Semantic Indexing (LSI) performance. *Information Processing & Management*, 42(1):56–73, jan 2006.

[6] M. Lan, C.-L. Tan, H.-B. Low, and S.-Y. Sung. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *Special interest tracks and posters of the 14th international conference on World Wide Web - WWW '05*, page 1032, 2005.

[7] M. L. Littman, S. T. Dumais, and T. K. Landauer. Automatic Cross-Language Information Retrieval Using Latent Semantic Indexing. In *Cross-Language Information Retrieval*, pages 51–62. Springer US, Boston, MA, 1998.

[8] D. J. C. Mackay. *Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[9] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[10] A. Muhič, J. Rupnik, and P. Škraba. Cross-lingual document similarity. In *Proceedings of the ITI 2012 34th International Conference on Information Technology Interfaces*, pages 387–392, June 2012.

[11] T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] R. J. Price and A. E. Zukas. Application of Latent Semantic Indexing to Processing of Noisy Text. pages 602–603. Springer, Berlin, Heidelberg, 2005.

[14] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[15] B. Rosario. Latent Semantic Indexing: An overview. 2001.

[16] J. Rupnik, A. Muhič, G. Leban, B. Fortuna, and M. Grobelnik. News across languages - Cross-lingual document similarity and event tracking. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 5050–5054, dec 2016.

[17] M. Saad, D. Langlois, and K. Smaïli. Cross-lingual semantic similarity measure for comparable articles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8686:105–115, 2014.

[18] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, jan 1988.

[19] R. Skadiņš, J. Tiedemann, R. Rozis, and D. Deksne. Billions of parallel words for free: Building and using the eu bookshop corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[20] R. Steinberger, B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, D. Tufiş, and D. Varga. The JRC-Acquis: A Multilingual Aligned Parallel Corpus with 20+ Languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, number 124, 2006.

[21] J. Tiedemann. Parallel data, tools and interfaces in opus. In N. C. C. Chair), K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[22] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[23] E. M. Voorhees and E. M. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '93*, pages 171–180, New York, New York, USA, 1993. ACM Press.

[24] M. Yano, J. D. Penn, G. Konidaris, and A. T. Patera. Matrices and Least-Squares. 2012.