Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática

2012

Hugo Rafael Teixeira Soares Araújo

Exploração de Literatura Biomédica usando Semântica Latente

Exploring Biomedical Literature using Latent Semantic Indexing

Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática

2012

Hugo Rafael Teixeira Soares Araújo

Exploração de Literatura Biomédica usando Semântica Latente

Exploring Biomedical Literature using Lantent Semantic Indexing

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Mestrado Integrado em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor José Luís Guimarães Oliveira, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Sérgio Guilherme Aleixo de Matos, Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Dr. Joaquim Manuel Henriques de Sousa Pinto
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Prof. Dr. José Luís Guimarães Oliveira (orientador)
Professor Associado da Universidade de Aveiro

Dr. Sérgio Guilherme Aleixo de Matos (co-orientador)
Investigador Auxiliar da Universidade de Aveiro

Prof. Dr. Carlos Manuel Jorge da Silva Pereira
Professor Adjunto do Instituto Superior de Engenharia de Coimbra

agradecimentos / acknowledgments

Agradeço em primeiro lugar aos meus orientadores José Luís Oliveira e Sérgio Matos pelo acompanhamento e disponibilidade demostrados ao longo destes meses na elaboração deste trabalho e a todos os elementos do grupo de bioinformática do IEETA pelo acolhimento e apoio.

A todos os que estiveram mais de perto durante estes anos. Aos que me acompanharam durante o curso e aos que me acompanharam fora do curso.
Aos que tiveram que me aturar com mais frequência, aos que comigo partilharam boas, más e fantásticas experiências.
Aos que são como irmãos, aos que são como família. Aos que foram para longe mas continuam tão perto. Aos que comigo estiveram ancorados e viveram momentos ao mesmo tempo fraternos e radicais.
A todos, obrigado por estarem presentes e por fazerem este caminho possível.

Por fim agradeço aos meus pais pelo apoio incondicional e paciência que tiveram e têm comigo, não só nesta fase universitária mas em toda a minha vida. Obrigado!

palavras-chave

Recuperação de Informação, Mineração de Texto, Modelação de Tópicos, Indexação Semântica Latente, Anotação de Termos, Visualização de Tópicos.

resumo

O rápido crescimento de dados disponível na Internet e o facto de se encontrar maioritariamente na forma de texto não estruturado, tem criado sucessivos desafios na recuperação e indexação desta informação. Para além da Internet, também inúmeras bases de dados documentais, de áreas específicas do conhecimento, são confrontadas com este problema. Com a quantidade de informação a crescer tão rapidamente, os métodos tradicionais para indexar e recuperar informação, tornam-se insuficientes face a requisitos cada vez mais exigentes por parte dos utilizadores. Estes problemas levam à necessidade de melhorar os sistemas de recuperação de informação, usando técnicas mais poderosas e eficientes.

Um desses métodos designa-se por *Latent Semantic Indexing* (LSI) e, tem sido sugerido como uma boa solução para modelar e analisar texto não estruturado. O LSI permite revelar a estrutura semântica de um corpus, descobrindo relações entre documentos e termos, mostrando-se uma solução robusta para o melhoramento de sistemas de recuperação de informação, especialmente a identificação de documentos relevantes para a pesquisa de um utilizador. Além disso, o LSI pode ser útil em outras tarefas tais como indexação de documentos e anotação de termos.

O principal objectivo deste projeto consistiu no estudo e exploração do LSI na anotação de termos e na estruturação dos resultados de um sistema de recuperação de informação. São apresentados resultados de desempenho destes algoritmos e são igualmente propostas algumas formas para visualizar estes resultados.

keywords

Information Retrieval, Text Mining, Topic Modeling, Latent Semantic Indexing, Terms Annotation, Topic Visualization.

abstract

The rapid increase in the amount of data available on the Internet, and the fact that this is mostly in the form of unstructured text, has brought successive challenges in information indexing and retrieval. Besides the Internet, specific literature databases are also faced with these problems. With the amount of information growing so rapidly, traditional methods for indexing and retrieving information become insufficient for the increasingly stringent requirements from users. These issues lead to the need of improving information retrieval systems using more powerful and efficient techniques.

One of those methods is the *Latent Semantic Indexing* (LSI), which has been suggested as a good solution for modeling and analyzing unstructured text. LSI allows discovering the semantic structure in a corpus, by finding the relations between documents and terms. It is a robust solution for improving information retrieval systems, especially in the identification of relevant documents for a user's query. Besides this, LSI can be useful in other tasks such as document indexing and annotation of terms.

The main goal of this project consisted in studying and exploring the LSI process for terms annotations and for structuring the retrieved documents from an information retrieval system. The performance results of these algorithms are presented and, in addition, several new forms of visualizing these results are proposed.

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
DF	Document Frequency
IDF	Inverse Document Frequency
JVM	Java Virtual Machine
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
MAP	Mean Average Precision
MDS	Multidimensional Scaling
MESH	Medical Subject Headings
MTI	Medical Text Indexer
NCBI	National Center for Biotechnology Information
NLM	National Library of Medicine
NLTK	Natural Language Toolkit
PLSA	Probabilistic Latent Semantic Analysis
SVD	Singular Value Decomposition
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
UCLA	University of California Los Angeles
WEKA	Waikato Environment for Knowledge Analysis

1. Introduction

1.1. Motivation

Internet is the most used mean for accessing information, thanks to its evolution over the last years and to the growing amount of data available. Another advantage of the Internet is the fact that data is stored in digital format, usually in the form of text, which makes it easier to access and most important, to search for. This task is called information retrieval and it can be defined as the process of "finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections" [1]. Two well-known examples of information retrieval systems are Google and Yahoo. Information retrieval systems are not only used for searching over the Internet as a whole, but also for searching specific large digital databases and collections. Examples of these specific databases are repositories of literature, whose available contents are increasing over the time.

However, this rapid increase of the amount of data also creates some difficulties when retrieving information: most of the information exists in the form of non-structured text, making it harder to interpret and extract. Because of this, most part of the information retrieved is not exactly what the user intended.

Basic information retrieval systems work by searching for a query formed by some keywords given by the user, and retrieving the most relevant documents for that query. Usually these systems assume that the relevant documents contain those keywords or closely related ones. Yet, such big amounts of data create new challenges for searching for useful information, causing those retrieval systems to be less efficient.

Because of this, it becomes necessary to perform a deeper analysis to the texts, using text mining techniques, in order to extract information from them, and use more complex and automated methods to enhance browsing and searching digital libraries. This is done by using some text understanding techniques, such as natural language processing and topic modeling. One example of such techniques is Latent Semantic Analysis (LSA), which allows connecting documents to terms or concepts, even if they do not appear explicitly in the documents, by discovering relations between those terms and the ones actually in the documents. This technique also allows identifying the most relevant terms in a document or in a set of documents.

This way, documents which have a stronger relation with the searched terms - even if they are not present in those documents - will be more relevant to the search, making the information retrieval more efficient.

1.2. Goals

In order to improve information retrieval in face of the growing amount of data in digital databases, more complex and efficient techniques are needed.

The main goals for this project were:

- Explore and study semantic indexing techniques, especially *Latent Semantic Indexing*, existing implementations, and their advantages and disadvantages;
- Understand how *Latent Semantic Indexing* can be of use and advantages for information retrieval in several domains, especially in the biomedical domain;
- Develop a new approach that allows analyzing and exploring *Latent Semantic Indexing* results in a visual way.

1.3. Dissertation Structure

The dissertation consists of the following remaining chapters:

Chapter 2 presents the bibliographic research for the development of this project. It includes the reasons for the use of *Latent Semantic Indexing*, and an illustration of its procedure. It also contains an overview of some implementations of *Latent Semantic indexing*, comparing their features, advantages and disadvantages. Finally a brief reference to similar techniques is also included.

Chapter 3 describes an approach, using *Latent Semantic Indexing*, which consists in recommending MeSH terms for annotating biomedical articles. It starts by an explanation of how the MEDLINE database is structured and how it interacts with information retrieval systems. It continues with a description of how the experiment was performed, and finally ends with the presentation and discussion of the obtained results.

Chapter 4 describes an approach for improving the results of an information retrieval system, by finding different classes of documents for the same query, using *Latent Semantic Indexing* and by proposing several solutions for visually presenting those results. Once more an explanation of the problem is made, followed by the description of the experiment and the presentation and discussion of the results.

Chapter 5 provides the final conclusions of this research, discussion of the obtained results and remaining issues.

2. State of the Art

2.1. Overview

As stated, most information retrieval techniques are too simple and inefficient, basically relying on trying to match words of queries with words of documents. According to [2], "the problem is that users want to retrieve on the basis of conceptual content, but individual words are unable of providing a reliable evidence about the conceptual topic or meaning of a document." The concept or meaning of a document can be defined in many ways, being unlikely that one single term in a user's query may match the document's concept.

Besides, this kind of retrieval techniques has two main issues: *synonymy* and *polysemy*. *Synonymy* refers to the fact that multiple words may have the same meaning. This leads to a great problem: users may search for data describing the information with different terms depending on different contexts, needs or even different linguistic habits. Those terms may even not match the ones by which the information has been indexed. So when a user searches for information using a set of terms, if the retrieval system simply matches the terms of the query with the ones in the documents, it will only retrieve the ones containing literally those terms, but possibly discarding results that may be relevant for the user. For instance, if the user searches for the term "*car*", only documents containing "*car*" will be considered relevant, but it would be probably interesting for the user to get documents concerning "*automobile*". Synonymous words are only a small example to describe this problem, but if we think about it in a context level, the problem still remains. Documents in the same context of the query still can be relevant for the user, even if they don't have terms from the query or synonyms. Once more, if the user searches for the word "*car*", documents with "*car*" (or "*automobile*" if we solve the synonyms issue) will still be considered relevant, however documents in the same context but which don't have explicitly those words will be discarded. For instance, documents about *trucks*, *roads* or *motorcycles*, will possibly be related to the user's interests. To avoid these issues, there are some techniques used in Information Retrieval that help improving the retrieval performance, such as query expansion. This technique works by reformulating the query by adding new terms or replacing the original ones by terms with similar meaning [3]. This way, the probability of finding relevant documents increases.

Polysemy describes words that have multiple meanings, such as *bank* or *party*. This is a common problem regarding retrieval techniques as well, since users may get information that they are not interested in. For instance if the user searches for the word "*apple*", he will clearly get two kinds of results: the ones concerning the *Apple Corporation* and the ones concerning the fruit apple, which makes a great part of the results useless for the user. *Polysemy* leads retrieval techniques to have inaccurate results, since much of them will be not interesting for the user. On the other hand, *synonymy* leads to lower values of *recall*, being a great part of the relevant results out of the retrieved ones [2].

To minimize issues such as *synonymy* and *polysemy*, several techniques have been developed, which help interpreting and extracting information from documents. One of those techniques, that has deserved much attention and that is the scope of this project is the *Latent Semantic Analysis* (LSA) or, as usually known in the context of information retrieval, *Latent Semantic Indexing* (LSI) [2].

2.2. Latent Semantic Indexing

2.2.1. LSI Structure

LSI is "a high-dimensional linear associative model using no human knowledge for its learning mechanism" usually used to "analyze large corpora of natural text and generate a representation that captures the similarity of words and documents" [4]. This representation is done by mapping the words and documents (or other sets of words such as sentences or paragraphs) as points of a dimensional semantic space. LSI is closely related to neural network models, but is based on Singular Value Decomposition, a factorization technique for complex matrices [5].

The first step when using LSI is to create the documents-terms co-occurrence matrix from a corpus (Figure 2.1). A way of making it more efficient is to weight the terms so as to express how important that term is, not only in a document but also in the whole corpus. Usually a *term frequency-inverse document frequency* (TF-IDF) weighting scheme is used. TF-IDF combines two measures to indicate the term importance and produce a weight to each term in each document [1]. Those measures are: the *term frequency* (TF) and the *document frequency* (DF). TF indicates the number of occurrences of a term in a document, indicating how relevant that term is for that specific document. DF indicates the frequency of a term in the whole corpus (the number of documents containing that term).

The idea is to reduce the TF weight of a term by a factor that grows with its document frequency. This way, we give less important to terms like “the”, which occurs frequently in the corpus and more importance to terms which occurs frequently in a document but rarely in all documents. Thus, the *inverse document frequency* (IDF) is used, defined as follows, for the term t and a corpus with N documents [1]:

$$IDF_t = log \frac{N}{DF_t} \tag{2.1}$$

Combining both TF and IDF we obtain the TF-IDF weighting, assigning to term t a weight in document d, as shown in the following equation [1]:

$$TF - IDF_{t,d} = TF_{t,d} \times IDF_t \tag{2.2}$$

As expected, the co-occurrence matrix size will be proportional to the number of documents and to the number of different terms in the whole corpus. So, for instance, if we consider 1000 documents and 2000 different terms over all documents, there will be a 2 million cells matrix, which leads to performance problems when processing such matrix. Besides, for each document only a few different terms are used, so most of them have a frequency of zero. This makes the matrix to be sparse and a great part of it useless and redundant.

Terms		d1	d2	d3
a		1	1	1
arrived		0	1	1
damaged		1	0	0
delivery		0	1	0
fire		1	0	0
gold	A =	1	0	1
in		1	1	1
of		1	1	1
shipment		1	0	1
silver		0	2	0
truck		0	1	1

Figure 2.1 Example of a co-occurrence matrix. Adapted from [6].

The next step in the LSI implementation is reducing the matrix dimensions. To accomplish this, the *Singular Value Decomposition* (SVD) is used, where the matrix is factorized in a reduced number of dimensions. The result of this decomposition is the product of three different matrices, one representing the documents, other representing the terms and a third one, which is a diagonal matrix whose elements are the singular values of the original matrix. Besides making the matrix less sparse, SVD allows to group related words.

The values of both matrices (*eigen values*) corresponding to each term and each document can be represented in a dimensional semantic space. This way, terms and documents will be mapped so that we can see the relation between each term and the several documents and its relevance to each document (Figure 2.2). Moreover, we can apply a similarity method so it can be possible to rank the most relevant documents for some query. Usually the similarity calculation is done through the dot product, reflecting the angle between two terms mapped in the semantic space: this method is called *Cosine Similarity*.

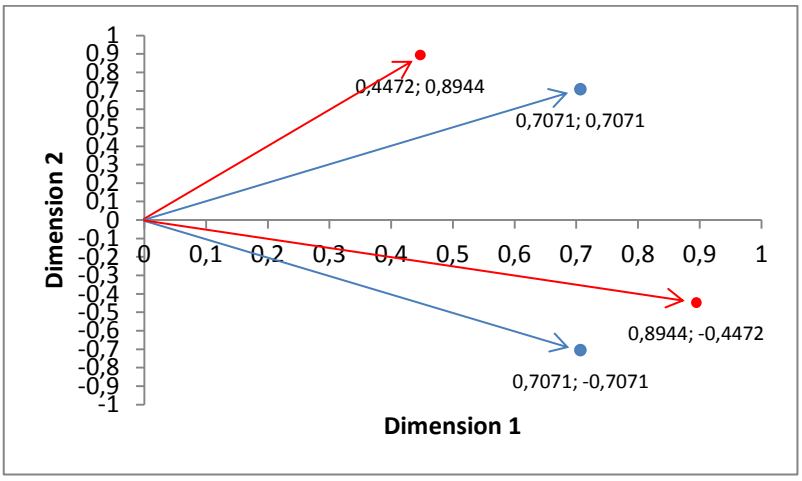


Figure 2.2 Terms and Documents mapped in a semantic space. Adapted from [7].

2.2.2. Singular Value Decomposition

It is easy to conclude that the most important and complex step in the LSI implementation is the dimensional reduction of the co-occurrence matrix through SVD. So, it is extremely important to understand its mechanism and its results.

SVD is a factorization technique for complex matrices used in linear algebra, closely related to several mathematical and statistical techniques in various other fields as eigenvector decomposition, spectral analysis and factor analysis [2]. Basically, the processed matrix is decomposed into the product of three different matrices containing *eigenvectors* and *eigenvalues*, by a process called *eigen analysis*:

$$A = USV^T \tag{2.3}$$

The columns of *U* are the *eigenvectors* of the AA^T matrix (*left eigenvectors*) and the columns of *V* are the *eigenvectors* of the A^TA matrix (*right eigenvectors*). V^T is the transposed matrix of *V* and *S* is a diagonal matrix which elements are the singular values of A, the nonnegative square roots of the eigenvalues of AA^T [8]. Usually the *U* matrix represents the terms of the corpus and V^T represents the documents (Figure 2.3).

$$\mathbf{U} = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & 0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix} \quad \mathbf{V^T} = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

Figure 2.3 Resultant matrices from SVD for the example in Figure 2.1. Adapted from [6].

The final step of SVD is the dimensional reduction. This can be done choosing a subset of dimensions, usually truncating the three resulting matrices, depending on the number of dimensions we need to keep (Figure 2.4) [9].

The number of dimensions that one should keep when using SVD has led to a big discussion over the time. There isn't any ideal number of dimensions for SVD and it must be decided based on trials and analyzing the obtained results. There are also some factors to consider, such as the size of our dataset (a larger dataset will probably need a bigger number of dimensions) or performance, since with such a complex technique as SVD, the more dimensions we keep, the more time it will need to execute. Nevertheless some studies concluded that for small datasets (about 1000 documents), 100 dimensions are enough [10]. For medium sized collections, formed by 1000 to 10000 documents, the best number of dimensions is about 300. For collections with millions of documents, about 400 dimensions should be used [11].

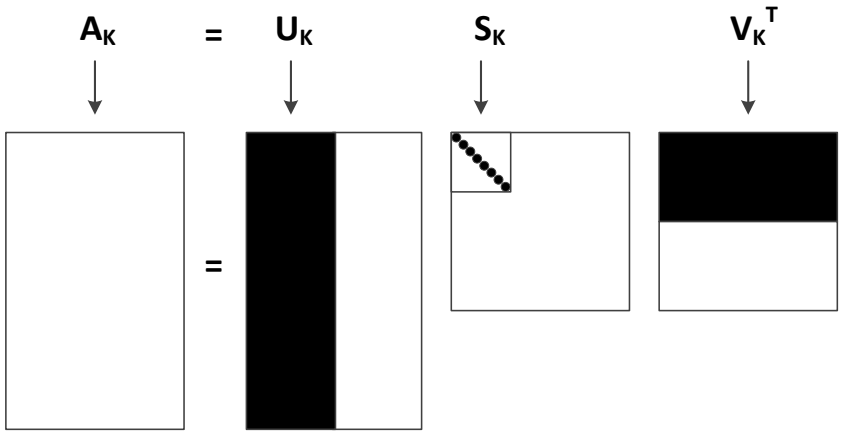


Figure 2.4 Dimensional Reduction. Adapted from [7].

2.2.3. Cosine Similarity

The SVD values are not easy to interpret by just looking at them. So it is important to find a method that allows us to show those results in a more understandable way.

There are two forms of doing that: The first one is to simply map the SVD values for the documents and terms as vectors into a vector space according to the number of dimensions used (Figure 2.5) and then just see the proximity between them to know if the documents are more or less similar to each other or if they are more or less related to each term. This way also lets us to know which documents and terms are more relevant for a certain dimension, an important aspect that will be discussed in further chapters.

The second way is to measure the similarity between documents. For that we used cosine similarity, a very common method in information retrieval systems. Cosine Similarity calculates the dot product of two vectors reflecting the angle between them in a vector space (Figure 2.6):

$$\vec{A} \cdot \vec{B} = |A||B| \cos\theta \Leftrightarrow \cos\theta = \frac{\vec{A} \cdot \vec{B}}{|A||B|} \tag{2.4}$$

So with cosine similarity it is possible to measure the similarity between two documents or even between a specific document and a set of other documents (for instance a user query against a corpus of documents), allowing us to rank them by similarity.

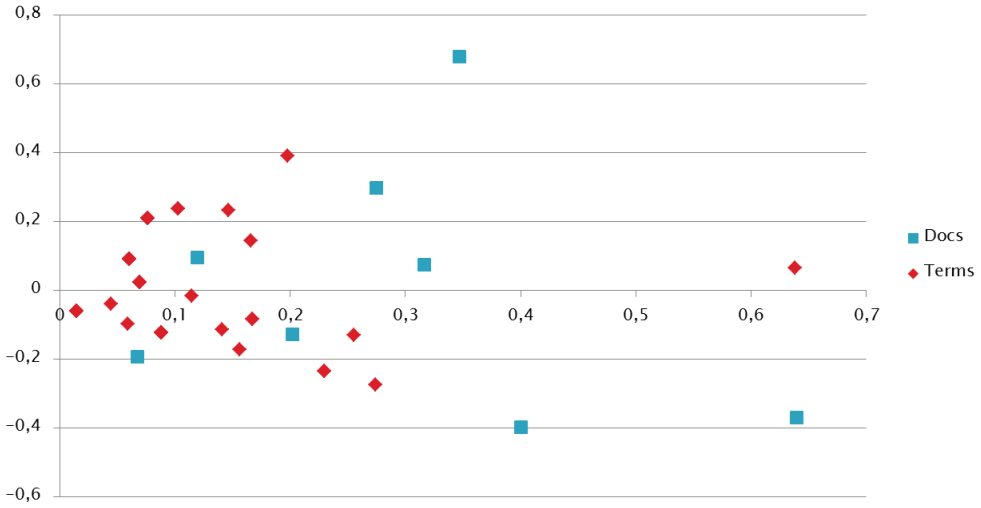


Figure 2.5 Documents and terms mapped in a vector space.

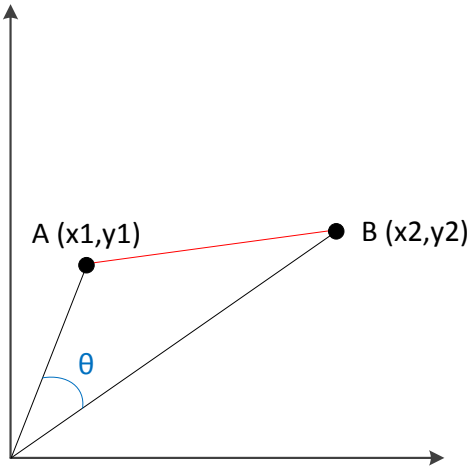


Figure 2.6 Cosine Similarity calculates the angle between two vectors in order to measure their similarity.

2.2.4. LSI Example

In order to better understand the process of LSI and its advantages, we use a typical example from [2] that shows the transformations and results after applying it over a small dataset.

Table 2.1 shows a sample dataset of documents and the frequency with each term occurs in each document.

The documents consist of the titles of nine Bellcore technical memoranda, and were divided in two different classes (*m* and *c*), with the ones from the same class having the same subject. Only the terms that occur in more than one title were considered for indexing.

As we can see, the co-occurrence matrix is too sparse and full of zeros, being that the main reason for reducing it using SVD. Since it is a small dataset, only two dimensions were kept when reducing the matrix.

Table 2.1 Term-Document matrix of a sample dataset, adapted from [2]. There are two classes of documents – five about human-computer interaction (c1-c5) and four about graphs (m1-m4).

Titles:

c1: Human machine interface for Lab ABC computer applications
c2: *A survey of user opinion of computer system response time*
c3: *The EPS user interface management system*
c4: *System and human system engineering testing of EPS*
c5: *Relation of user-perceived response time to error measurement*

m1: The generation of random, binary, unordered trees
m2: *The intersection graph of paths in trees*
m3: *Graph minors IV: Widths of trees and well-quasi-ordering*
m4: *Graph minors: A survey*

	c1	c2	c3	c4	c5	m1	m2	m3	m4
Human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
User	0	1	1	0	1	0	0	0	0
System	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
Survey	0	1	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	0
Graph	0	0	0	0	0	0	1	1	1
Minors	0	0	0	0	0	0	0	1	1

Figure 2.7 shows the geometric representation for terms and documents after the SVD has been applied. It is clear that the terms are closer to the documents where they appear, being even clearer the separation between the two different classes of documents.

Next, the query "*human computer interaction*" was used, in order to find its relevant documents. A simple term matching technique would return the documents c1, c2 and c4 since they share the same terms of the query. However documents c3 and c5 also belong to the same class and should be consider relevant as well.

The latent semantic structure method uses the derived factor representation to process the query [2]. It can then be represented as a "pseudo-document" in the factor space. As we can see, the query is near to the documents from class *c* and not to the ones from class *m*. Even the document c5, which has no common words with the query, is near to it in the factor space representation.

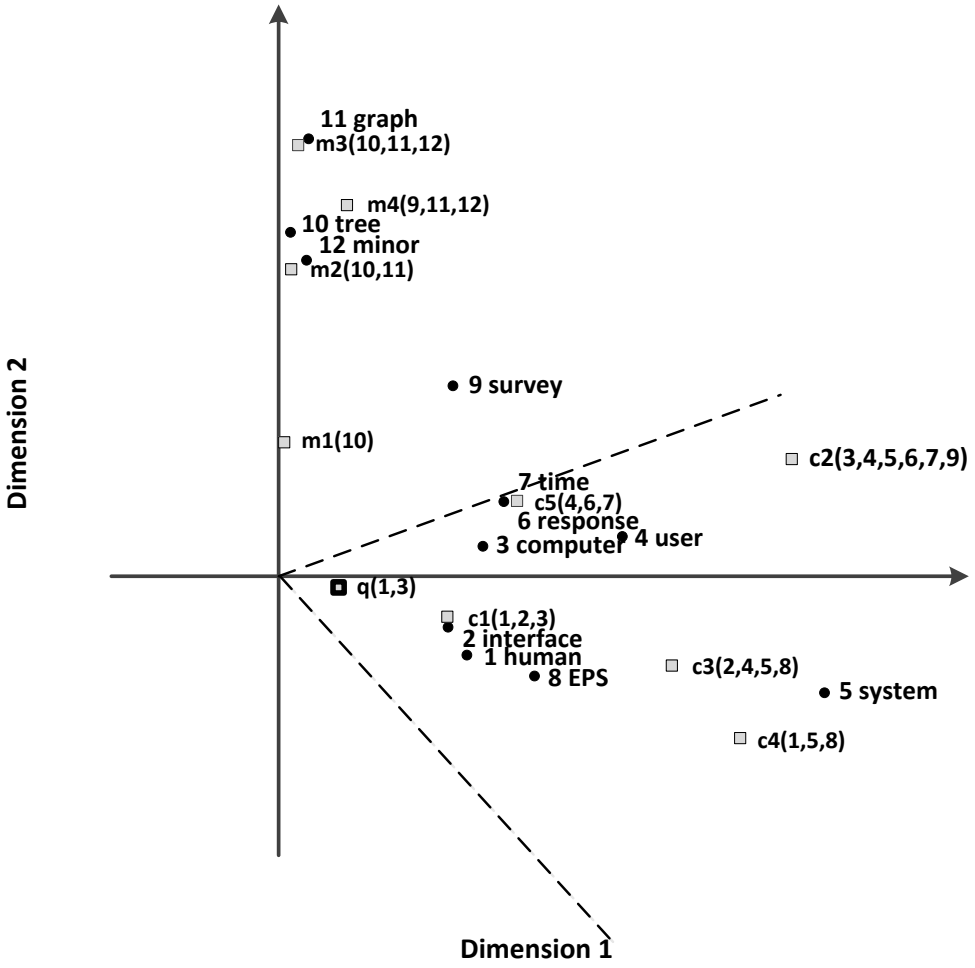


Figure 2.7 Two-dimensional representation of documents and terms from the sample adapted from [2]. Terms are represented by circles and Documents by squares. The query "human computer interaction", is represented by the point q as a pseudo-document. The dotted lines represent a cosine of 0.9 from the query.

Figure 2.7 shows clearly enough the similarity between the query and the several documents, regarding this particular example. However, using a larger dataset, the representation would be much harder to interpret. An alternative solution is to calculate the cosine similarity to retrieve the most similar documents for the query (Table 2.2).

This is a more understandable way to see the results, but the conclusions are the same: documents from class *c* are more relevant with similarity values near to 1, and documents from class *m* are not similar at all, even having negative similarity values.

Table 2.2 Similarity scores for the query "human computer interaction" using Cosine Similarity.

Query: *"human computer interaction"*		
c3:	*The EPS user interface management system*	0.998
c1:	Human machine interface for Lab ABC computer applications	0.998
c4:	*System and human system engineering testing of EPS*	0.986
c2:	*A survey of user opinion of computer system response time*	0.937
c5:	*Relation of user-perceived response time to error measurement*	0.907
m4:	*Graph minors: A survey*	0.050
m3:	*Graph minors IV: Widths of trees and well-quasi-ordering*	-0.098
m2:	*The intersection graph of paths in trees*	-0.106
m1:	*The generation of random, binary, unordered trees*	-0.124

2.3. LSI/SVD Implementations

Once we know how LSI proceeds (especially SVD), the next step relies on presenting some existing implementations of LSA/LSI or SVD that can be used in this project.

When exploring those solutions, we initially preferred the ones implemented in the Java programming language, because of it flexibility and portability so it would be easier to incorporate in the whole project. Nevertheless, same analyzed tools are implemented in other programming languages such as Python.

As stated, the base and most important part of LSI is the dimensional reduction through SVD. Therefore, more than LSI it is important to analyze tools that have an SVD implementation and can be easily integrated with the rest of the LSI steps.

2.3.1. Frameworks

After an initial analysis at the existing frameworks, some were selected for a deeper analysis: Weka, LingPipe, Apache Mahout, S-Space and Gensim.

It is important to highlight some positive and negative aspects of each one, before comparing their performance and results.

Weka

Weka[1] (Waikato Environment for Knowledge Analysis) is a collection of machine learning and data mining algorithms developed by the University of Waikato (New Zealand), implemented in Java. The Weka API has a LSI implementation. However, all the steps are done using methods provided by Weka, including the text pre-processing. For instance, the text tokenization is done using a tokenizer provided by Weka, with its output being an object of type *Instances*, which is also a Weka object type. This makes it harder to integrate Weka with other technologies, which may become an obstacle for future development of the project.

LingPipe

LingPipe[2] is a text processing toolkit for computational linguistics, also developed in Java. It doesn't provide a complete implementation of LSI, but it has one for SVD. As we mentioned, SVD is the most complex part of LSI. The remaining parts (TF-IDF algorithm and Cosine Similarity) are easy to implement. Therefore, the fact that LingPipe doesn't have a LSI implementation is not a major problem, since we can easily integrate SVD with other TF-IDF and cosine similarity implementations.

A positive aspect about LingPipe is that it provides a complete documentation about its tools and great tutorials that make it easy to use. However, its free use license is too limited, which may also become an obstacle for the project's development.

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://alias-i.com/lingpipe/

Apache Mahout

Mahout[3] is an Apache project which main goal is to develop scalable and distributed machine learning solutions for the Hadoop platform, also a Java implementation. As LingPipe, Mahout doesn't have a LSI implementation (just SVD) but has plenty of documentation and tutorials, making their tools development and use easier.

S-Space

The S-Space[4] project is a collection of algorithms for building Semantic Spaces. The research and development of this project is done by the Natural Language Processing group of the University of California, Los Angeles (UCLA).

S-Space has both LSI and SVD implementations. We choose to test just the SVD implementation so it would be easier to apply other tools for the corpus pre-processing and then have a more accurate comparison with the other solutions.

An important aspect about S-Space is the fact that it uses the *SVDLIBJ* library, an open source port of the *SVDLIBC* library (a C implementation of SVD) to Java. This aspect has a huge impact on the solution, in terms of speed.

Gensim

Gensim[5] is a Python framework (unlike the other solutions, which are implemented in Java) designed to automatically extract semantic topics from documents in an efficient way [12].

It offers a scalable solution for LSI as well, as a distributed version of that solution, which may be an important advantage. Another important aspect (that will be discussed in more detail) is the short execution time achieved by Gensim. These two aspects are due to the fact that the SVD algorithm used by Gensim is incremental, as we shall see further.

[3] http://mahout.apache.org/
[4] http://code.google.com/p/airhead-research/
[5] http://radimrehurek.com/gensim/

2.3.2. SVD Values

After an initial analysis, it is important to evaluate and compare the results and execution time of each solution. To easily interpret the obtained results, we used the same example from section 2.2.4 (Table 2.1). At this stage, the main goal was to compare the SVD results, so non TF-IDF was used.

Table 2.3 SVD values of the analyzed documents for each solution.

	Weka		LingPipe		Mahout		S-Space		Gensim	
c1	0.07	0.19	-0.07	0.19	0.07	-0.19	0.07	0.19	-0.09	-0.21
c2	0.64	0.37	-0.64	0.37	0.64	-0.37	0.64	0.37	-0.65	-0.43
c3	0.20	0.13	-0.20	0.13	0.20	-0.13	0.20	0.13	-0.31	-0.15
c4	0.34	-0.07	-0.32	-0.07	0.32	0.07	0.32	-0.07	-0.31	0.08
c5	0.40	0.39	-0.40	0.40	0.40	-0.40	0.40	0.40	-0.41	-0.42
m1	0.27	-0.29	-0.27	-0.30	0.28	0.30	0.28	-0.30	-0.25	0.30
m2	0.27	-0.29	-0.27	-0.30	0.28	0.30	0.28	-0.30	-0.25	0.30
m3	0.32	-0.68	-0.35	-0.68	0.35	0.68	0.35	-0.68	-0.32	0.67
m4	0.12	-0.09	-0.12	-0.09	0.12	0.09	0.12	-0.09	-0.27	0.15

Table 2.4 SVD values of the analyzed terms for each solution.

	Weka		LingPipe		Mahout		S-Space		Gensim	
Human	-0.02	0.07	-0.02	0.06	0.02	-0.06	-0.02	0.06	0.02	-0.07
interface	0.06	0.10	-0.06	0.10	0.06	-0.10	0.06	0.10	-0.06	-0.10
computer	0.16	0.17	-0.16	0.17	0.16	-0.17	0.16	0.17	-0.16	-0.17
User	0.27	0.27	-0.27	0.27	0.27	-0.27	0.27	0.27	-0.27	-0.27
System	0.07	-0.02	-0.07	-0.02	0.07	0.02	0.07	-0.02	-0.07	0.02
response	0.23	0.24	-0.23	0.24	0.23	-0.24	0.23	0.24	-0.23	-0.24
Time	0.23	0.24	-0.23	0.24	0.23	-0.24	0.23	0.24	-0.23	-0.24
EPS	0.11	0.02	-0.11	0.02	0.11	-0.02	0.11	0.02	-0.11	-0.02
Survey	0.17	0.08	-0.17	0.08	0.17	-0.08	0.17	0.08	-0.17	-0.08
Trees	0.20	-0.40	-0.20	-0.40	0.20	0.40	0.20	-0.40	-0.20	0.40
Graph	0.10	-0.24	-0.10	-0.24	0.10	0.24	0.10	-0.24	-0.10	0.24
Minors	0.10	-0.24	-0.10	-0.24	0.10	0.24	0.10	-0.24	-0.10	0.24

The presented results show that all solutions have the same values for the documents, except Gensim (Table 2.3), which has very close values, nevertheless. Regarding the analyzed terms, all solutions, even Gensim have the exact same values (Table 2.4).

2.3.3. Performance

Another important aspect about performance is the time taken by each solution to process large corpora. To evaluate this aspect, we used three datasets with different numbers of documents and terms, to analyze the execution time of each solution for each dataset (Table 2.5):

- **Corpus 1** – *The Reuters Transcribed Subset*, a subset of documents took from the Reuters-21578 collection. It is formed by 104 documents and, after being tokenized (using the same algorithm for most solutions, except Weka and Gensim, which have their own tokenization tools), 4095 tokens. The co-occurrence matrix is then 4095x104;
- **Corpus 2** – *The King James Bible Testing Corpus*. Formed by 1231 documents and after tokenization, 33462 tokens. The co-occurrence matrix is 33462x1231;
- **Corpus 3** – *PubMed Archive,* a dataset with 182972 documents and 44225 different terms extracted from the PubMed database, creating a 44225x182972 matrix. In this case, we used an index built with annotated concepts extracted from the documents instead of a regular tokenization and without stop words.

The performance was measured using a 64 bit, 2.53GHz Inter Core 2 Duo computer with 3GB of RAM. For testing the java technologies, the *Java Virtual Machine* (JVM) was configured to have a maximum heap size of 1.3GB.

Still, there are some issues concerning each toolkit that may influence the results:

- Except for Weka and Gensim, we used the same tokenization algorithm for all solutions, offered by LingPipe. In the case of Gensim, being a python solution we used python tools for tokenization. About Weka, since it has a LSI implementation, the documents tokenization must be done using Weka tools. Nevertheless, the number of tokens for these two solutions is very close to the others.
- Another aspect is the number of dimensions to use on SVD, especially when using bigger datasets. On those cases we used 300 dimensions, except once more, for Weka, which calculates the number of dimensions depending on the number of documents.
- LingPipe has another important feature: it allows processing the SVD several times. Each time is referred to as an “epoch”. The minimum and maximum numbers of epochs can be passed as parameters when computing SVD. Increasing the number of epochs, the error in computing the SVD is reduced. However, the computing time also increases.

For the smaller corpus (Corpus 1), the difference between all the solutions is not relevant, although S-Space and Gensim have better results. However, when concerning a bigger corpus (Corpus 2) the results show that, first of all, S-Space is the best Java solution. The fact that S-Space uses a C based library to implement SVD (*SVDLIBC*) may be the reason for being so fast to compute comparing to the other Java solutions. The results presented by Gensim concerning this corpus, are also much better than the rest of the solutions, being very similar to S-Space.

We can see that none of the java implementations were able to compute the matrix regarding Corpus 3. With such a big matrix, the java virtual machine heap space runs out of memory in all cases. Only Gensim was able to compute Corpus 3, and with a very acceptable processing time.

Table 2.5 Time taken by each solution computing each corpus.

	Corpus 1	Corpus 2	Corpus 3
Weka	8 seconds	45 minutes	Out of memory
LingPipe	6 seconds	2 hours	Out of memory
Mahout	4 seconds	3,5 hours	Out of memory
S-Space	2 seconds	40 seconds	Out of memory
Gensim	2 seconds	40 seconds	10 minutes

2.3.4. Gensim

So far, it was demonstrated that Gensim is the solution with better results, especially regarding performance. So, it is important to understand the reasons that make Gensim a better option.

The main characteristic of Gensim is the fact that its SVD implementation uses the Brand's algorithm, an incremental algorithm that allows the set of documents to compute, to be updated [13]. This way, documents don't need to be loaded into memory at the same time, which makes the factorized matrix at a given time, to be much smaller and so, increasing the computing speed. Besides, as the term-document matrix doesn't need to be loaded to memory, there is no limit to the size of the corpus. This approach allows LSI to be computed in distributed way reducing even more the computational effort and increasing the computing speed.

To reach this approach, first of all, the input matrix A, corresponding to the documents, must be divided in several and smaller sub-matrices, called jobs, $A^{m \times n} = \left[{A_1}^{m \times c_1}, {A_2}^{m \times c_2}, \cdots, {A_j}^{m \times c_j}\right]$, being $\sum c_i = n$, m representing the terms and n the document. Jobs are then computed independently, one by one or distributed for several nodes. As soon as all jobs are done, the resultant decompositions of each one are merged in one. This process is done by two different algorithms, one for the job decomposition – Base decomposition, and the other for merging the decompositions – Merge decompositions [14].

Besides, the Gensim framework has other interesting tools that may be very useful when using LSI: as stated, it has a distributed version of its own LSI tool, which allows processing the corpus using several machines, decreasing the computational effort and increasing the processing speed. It also provides a *Cosine Similarity* tool which complements and makes LSI easy to use.

2.4. Latent Dirichlet Allocation

LSI has turned into a well-known model among vectorial semantics and retrieval techniques. However, in the last years other techniques and methods have been developed and are now very reliable and frequently used alternatives.

Some of those techniques are topic models. The main purpose of topic modeling is similar to the ones already discussed: to evolve retrieval systems by creating more complex and automated methods of organizing, managing and retrieving information from collections of documents.

Topic models proceed by extracting topics from a document collection, discovering patterns of word use and connecting documents with similar patterns. This way is possible to create probabilistic word-topic and topic-document associations, being easier to extract and understand information from documents [15].

Some topic models derive from LSI, such as the *Probabilistic Latent Semantic Indexing* (pLSI) and *Latent Dirichlet Allocation* (LDA). In fact, LDA is probably the best known topic model and "has served as springboard for many others topic models" [15].

According to [16], LDA is a" generative probabilistic model of a corpus". Basically "documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words".

Figure 2.8 shows an example of four topics discovered automatically using LDA, from a collection of documents. Each topic doesn't have any kind of theme previously assigned to it but it is rather defined by a set of highly probable words extracted from the documents.

Documents are then associated to the created topics with more or less probability according to their nature. This way, they can be retrieved by the topics to which they are related, improving the retrieval results.

NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNET
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
TEACHER	PROGRAMS	PERCENT	PREESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY

Figure 2.8 Four topics from a 100-topic LDA model applied to a subset of the TREC AP corpus, adapted from [9].

2.5. Conclusions

In this chapter, we studied and choose the best implementation of LSI among a set of options, according to their performances and results similarities. Our conclusions suggest that the python framework, Gensim is the best solution.

Nevertheless, the tests and results presented so far, were made over small corpora only intended to compare frameworks, and are not good enough to prove, not only Gensim's accuracy but mainly LSI's reliability.

In the following chapters we investigate how useful and advantageous LSI can be in various domains, especially in the biomedical domain and what we can achieve using this technique.

3. Recommending MeSH Terms using LSI

In this chapter, our main goal is to test how LSI can be applied to some existing information retrieval tasks in the biomedical domain. One important challenge is that of recommending MeSH terms for annotating biomedical articles from MEDLINE[6].

3.1. MEDLINE, PubMed and MeSH Terms

MEDLINE is one of the largest bibliographic databases of life science and biomedical information, containing more than 22 million records (as of October 2012), including citations and abstracts (around 14 thousand new citations are added every week), from more than 5,400 biomedical journals published in the United States and worldwide since the year of 1950[7].

All this information can be accessed through PubMed, an information retrieval system developed and maintained by the National Center of Biotechnology Information (NCBI) at the National Library Medicine (NLM).

With such a great volume of literature and considering its rapid growth, PubMed has great difficulties providing a good service concerning the management, indexing and search capabilities. In order to improve their services, NLM created the Medical Subject Headings (MeSH), a controlled vocabulary for indexing articles in MEDLINE, describing various biomedical topics, such as diseases, chemical and drugs[8].

Every article indexed in MEDLINE has been read by a human indexer whose task is to identify the main topics related to that article and assign the most specific terms from a list of MeSH terms, describing its content. Each article in MEDLINE/PubMed has about 10-15 MeSH terms assigned to it [17].

With each article annotated with MeSH terms, it is possible to search not only by free-text words, but also by specific MeSH terms according to a relevant topic, which improves the results that are retrieved [17]. Besides, it is easy to conclude that this strategy also facilitates document clustering [18] [19] and bioinformatics research as well [20].

[6] http://www.nlm.nih.gov/pubs/factsheets/pubmed.html
[7] http://www.nlm.nih.gov/bsd/num_titles.html
[8] http://www.nlm.nih.gov/pubs/factsheets/mesh.html

3.2. Recommending MeSH Terms

Since assigning MeSH terms to documents is done manually, it becomes a complex process, which takes much time and requires human understanding and knowledge of the articles and MeSH terms, being too expensive as well [21].

Considering this limitations, many approaches have been tested to automatically suggest MeSH terms for biomedical articles. These are usually based on: selecting MeSH terms from the nearest neighbor documents, using probabilistic models or machine learning methods to associate the documents text to MeSH terms, and using domain-specific knowledge resources and trigrams, as stated for instance in [18].

Our first and main goal with this experiment would be to use LSI to suggest MeSH terms for each document of a specific corpus and then compare our results with the state of the art.

3.2.1. Methodology using LSI

In order to compare our results with other studies, we used the dataset presented in [18]. We also follow a similar methodology. Thus, we start by retrieving the most similar documents from a training set for each test document. Then in a second phase, we collect all the MeSH terms assigned to those documents and finally, in a third phase, we assign each MeSH term with a score value, ranking them and considering the top N terms as relevant to that test document. Figure 3.1 shows how this process is done.

The main differences about our solution rely on the first step, since we retrieve the most similar documents through LSI, and on the last step, because we don't use the same algorithm and features to get the most relevant MeSH terms.

Recurring to the Gensim framework, we create a LSI model using a training corpus. Then we calculate the similarity of each test document against the training corpus.

Since Gensim provides a tool for calculating Cosine Similarity, we use the test document has a query, applying the LSI model over it, so it can be in the same LSI space as the training corpus. This way the similarity for each training document against the test document is calculated and they can be ranked by their similarity.

In [18] several different features are used for the ranking model such as Neighborhood features, word unigram/bigram overlap features, query-likelihood features, etc. Since we focus mainly on the similar documents retrieval, we only used one kind of features for the MeSH ranking: the Neighborhood features.

We tested our approach for two different cases, using one different neighborhood feature for each one. In the first case, after sorting the documents of the training corpus and defining a threshold to get only the most similar ones, we get the MeSH terms assigned to those documents and for each one, we count the number of documents where that term appears, as follows:

$$freq(M, D_k) = |D_i | M \epsilon D_i,\ D_i \epsilon\ \Omega| \tag{3.1}$$

Where Ω is the set of the most similar documents for a target test document D_k. According to this counting, we can rank the MeSH terms, take the top *N* (*N* being 25, in line to the other studies) and consider them as our suggested MeSH terms for that target test document.

In the second case, we sum the similarity score of each document instead of counting, as follows:

$$sim(M, D_k) = \sum_{M \epsilon D_i; D_i \epsilon \Omega_k} sim(D_k, D_i) \tag{3.2}$$

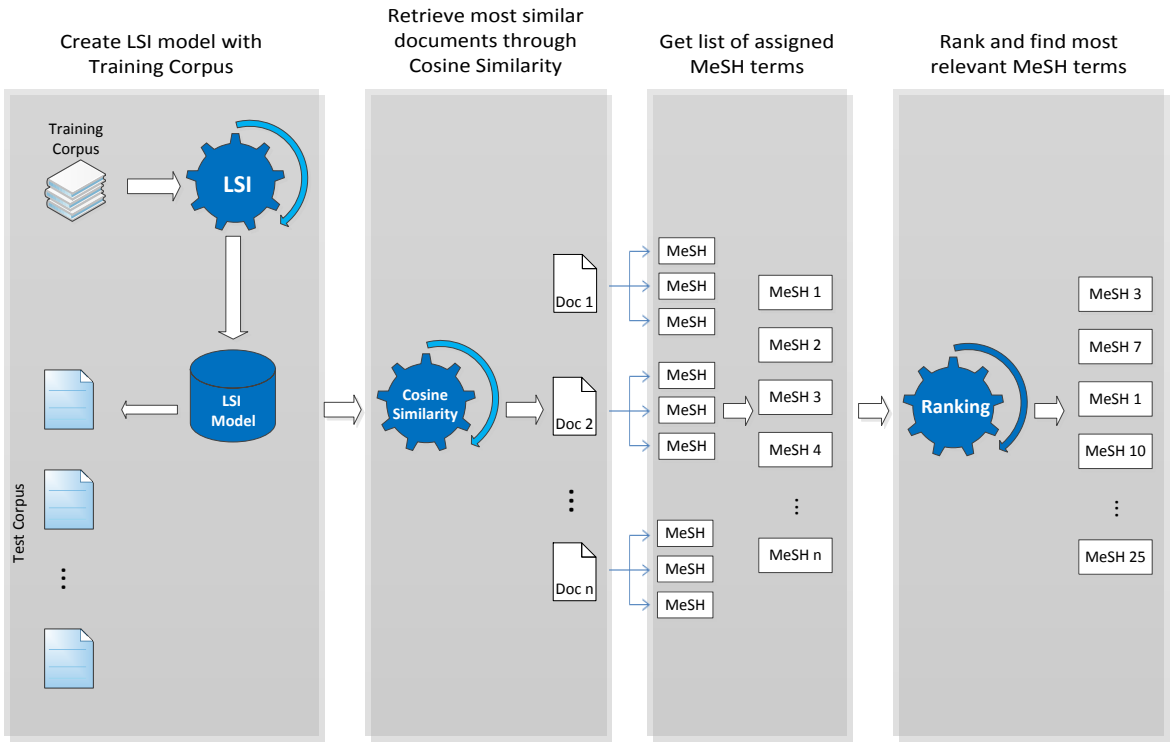


Figure 3.1 An illustration of our approach: from retrieving most similar documents to ranking suggested MeSH terms.

3.2.2. Methodology using LDA

Although our main goal with this experiment is to test the LSI capabilities to predict MeSH terms, the fact that Gensim provides a LDA implementation as well, allowed us to use and test LDA for the same purpose. However, unlike LSI, LDA is a probabilistic model and works in a different way. For that reason, the process to predict MeSH terms had to be a lot different from the one regarding LSI. For instance, cosine similarity can't be used to retrieve the most similar documents.

Figure 3.2 shows a general overview of our approach using LDA: as in the LSI case, we started by creating a LDA model using a training corpus. Being LDA a topic model, it creates a set of topics. Each document from the training set has a probability value of being related to each topic. Since each document has a list of MeSH terms assigned to it, the next step consists in, for each topic *T*, get a list of the MeSH terms assigned to the training documents and give each term *M* a score. That score consists of summing the probability values, for topic *T*, of all documents assigned with the term *M* (Figure 3.3).

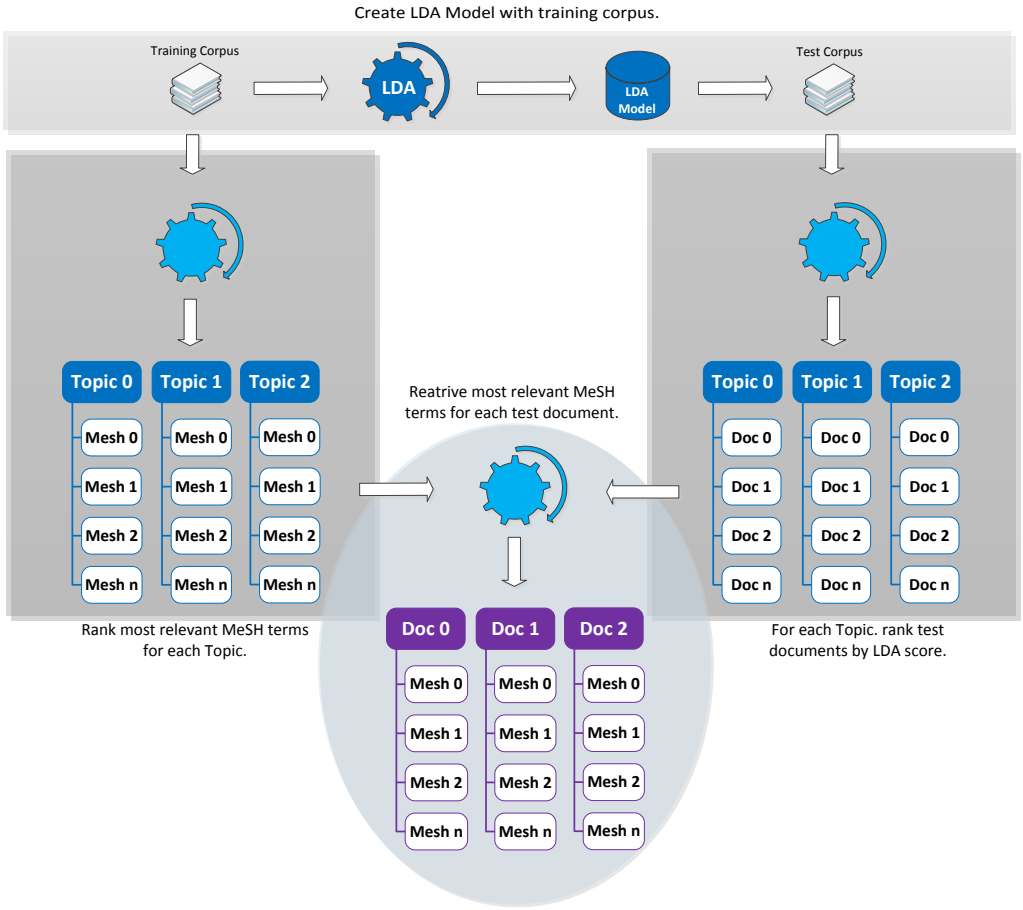


Figure 3.2 General overview of our approach using LDA.

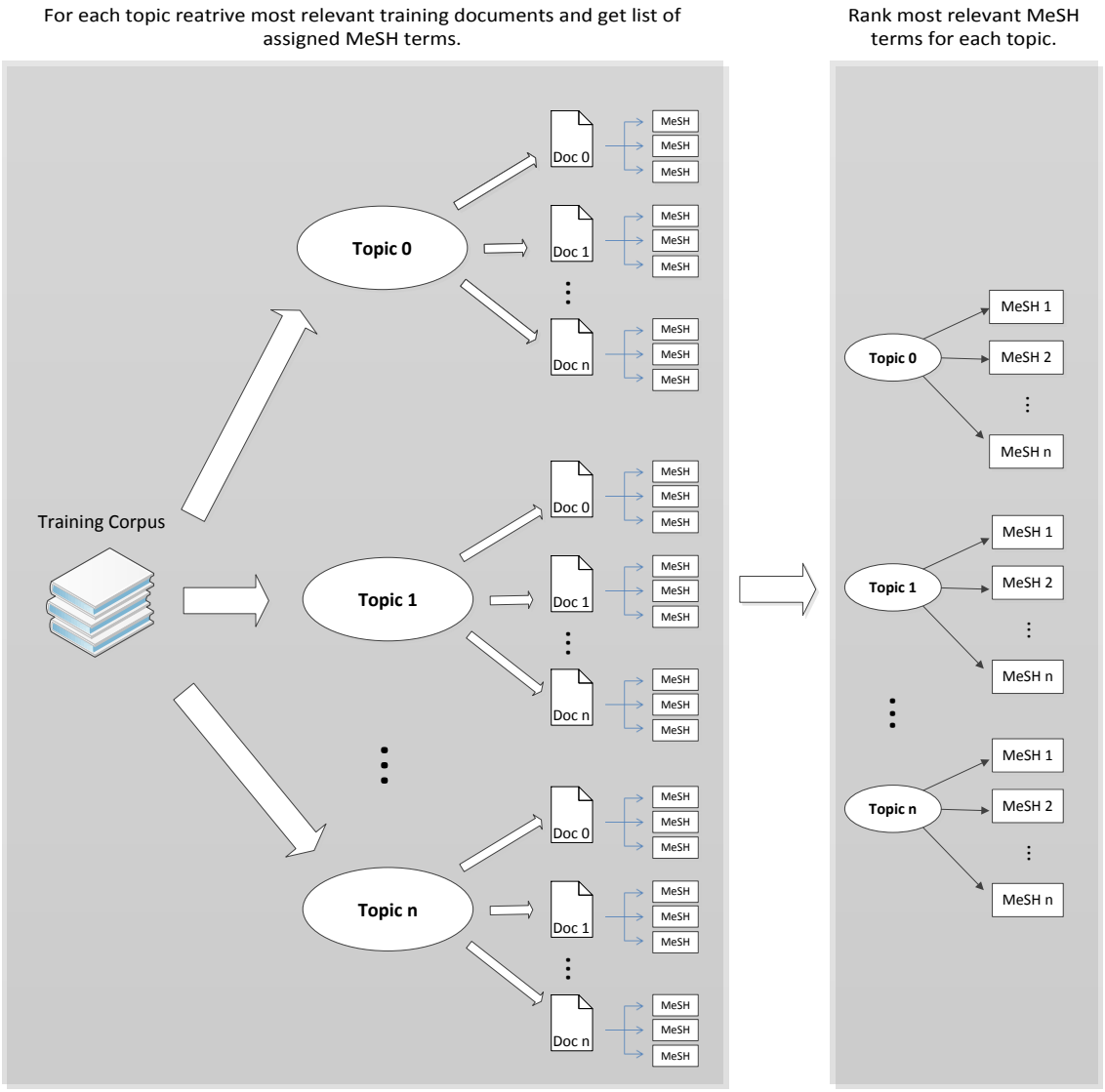


Figure 3.3 Process to get most relevant MeSH terms to each topic: First we get most relevant documents and then we rank the MeSH terms summing, for each one, the LDA score of each document.

Next, we run the LDA model over the test corpus, so we can get for each topic, a list of the test documents and their probability values (Figure 3.4).

The final step (Figure 3.5) is to calculate a score for each MeSH term and test document pair. This is achieved using the next formula:

$$S(M,D) = \sum_{T} P(D|T) * S(M,T) \tag{3.3}$$

Where $S(M,D)$ is the score of a MeSH term *M* given a test document *D*, $P(D|T)$ is the probability of a test document *D* given a topic *T* and $S(M,T)$ the score of a MeSH term *M* given a topic *T* calculated in the previous step.

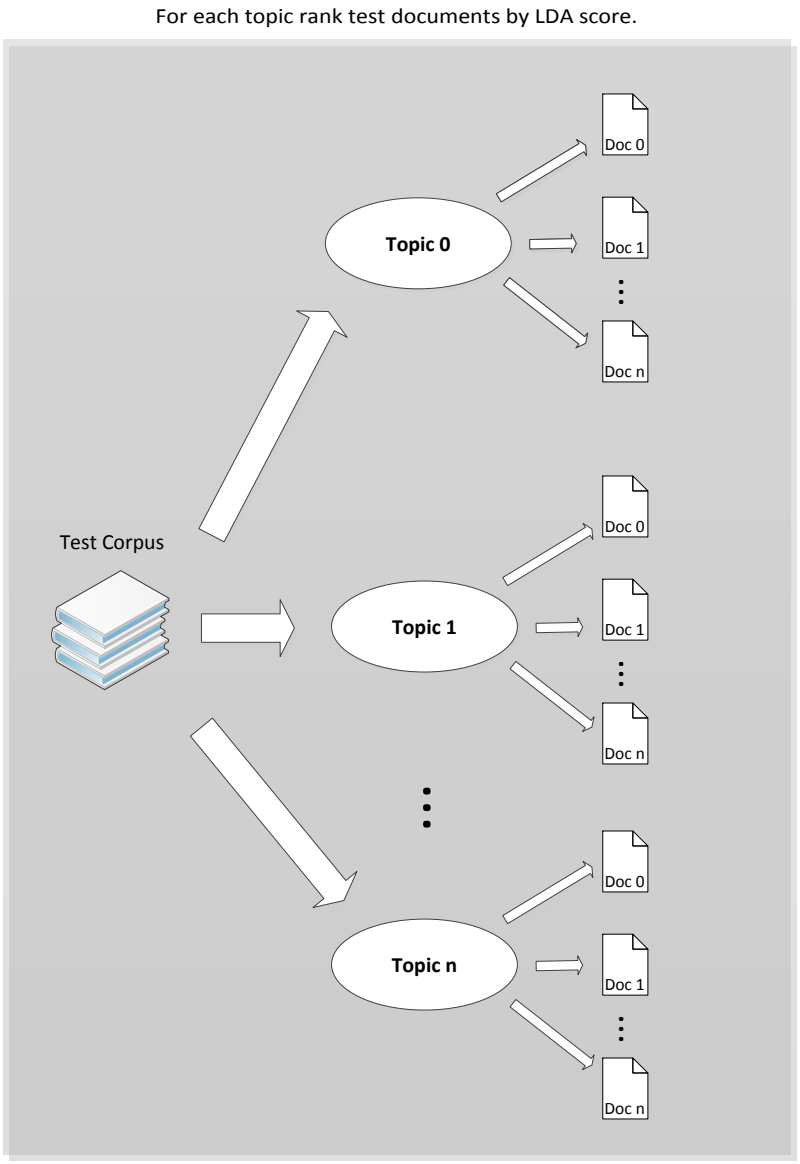


Figure 3.4 Process to rank test documents by Topic.

After this, we have a list of MeSH terms, for each test document where the terms are assigned with a score. This way we can apply a threshold to consider the top most relevant ones (25 as the experiment with the LSI and the other studies), being our predicted MeSH terms for that test document.

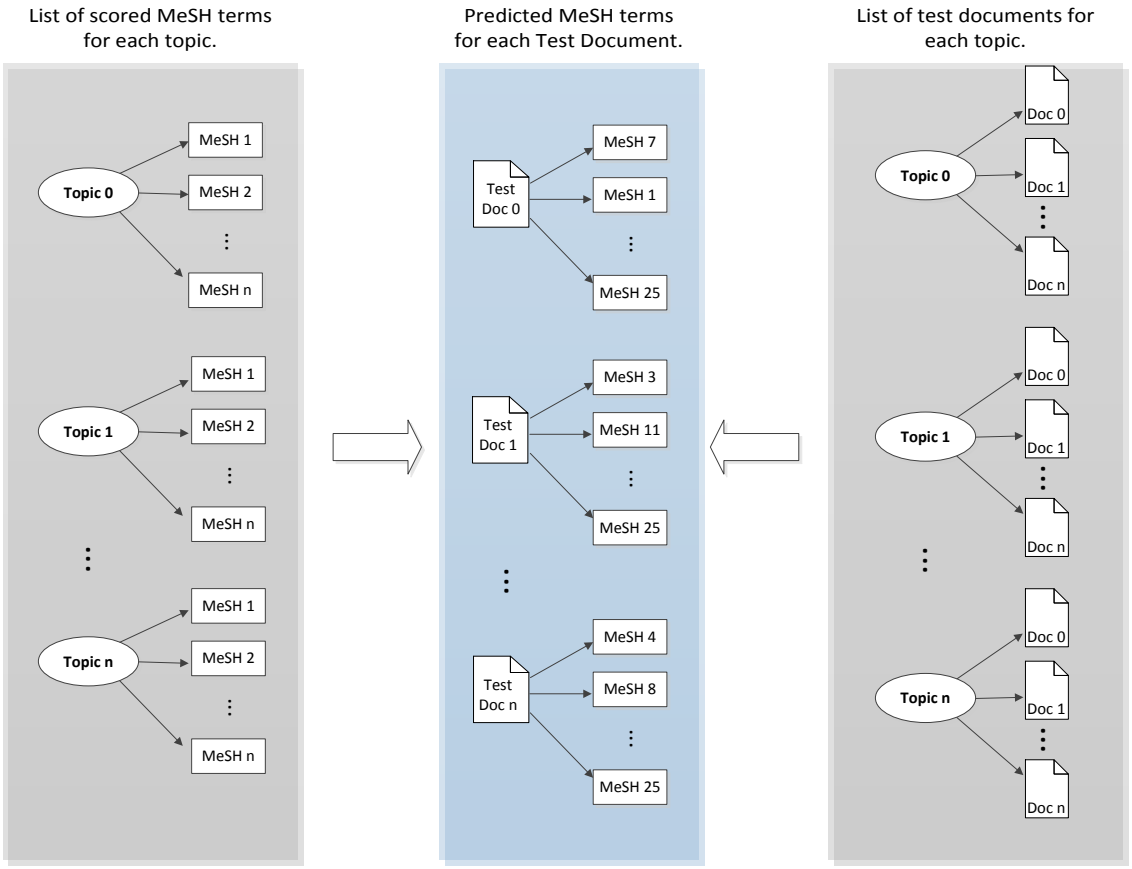


Figure 3.5 Process to retrieve predicted MeSH terms for each test document.

3.2.3. Datasets

In this experiment we used NLM2007, a dataset obtained from the NLM indexing initiative [22]. It was used for the same purpose by other methods in recent studies [23]. Thus, it was the best dataset to use, in order to compare our results with those methods.

The NLM2007 dataset contains 200 documents, each one composed by title and abstract. As stated in [18], information in full text was found to be of limited use for the automatic production of MeSH indexing recommendations, so it was not consider in the documents. We use the NLM2007 as our test corpus and as training corpus to create the LSI model we used a dataset formed by the 50 nearest neighbors of each test document, which makes a dataset with nearly 10000 documents. The neighbors of a specific document are those documents that are most similar to it. This similarity between documents was measured only by the words (title and abstract) they have in common [24].

3.2.4. Methods

Besides using two different models, as LSI and LDA, we took in consideration some other factors that may affect the results or the performance of our solution. Namely, the corpus processing, the number of dimensions or topics used to calculate both LSI and LDA and the threshold applied on the similarity score of documents retrieved by LSI.

Corpus Pre-processing

For the corpus pre-processing we used some natural language processing techniques, like tokenization, regularization and normalization over both corpora. All these techniques were done using NLTK[9] tools, like NLTK' *WordPunctTokenizer* and *Porter Stemmer*.

Besides removing punctuations and digits we also used a list of stop words provided by PubMed, to "clean" the documents for irrelevant terms. We also removed tokens with less than three alphanumeric characters, considering them irrelevant as well.

Corpus Processing

Besides testing LSI for recommending MeSH terms, we had a second purpose with this experiment, by using two different methods of our solution concerning the documents processing: we compare the use of tokens to the use of dictionary terms, with each document being assigned and composed just by relevant terms of that dictionary.

So, for instance, if one of the documents is "*We studied 92 consecutive end-stage renal disease (ESRD) patients receiving their first permanent hemodialysis vascular access at initiation of hemodialysis to identify variables that determine assignment of either a PTFE graft or a native AVF.*"

For the first case, the relevant tokens would be: '*studied*', '*consecutive*', '*end*', '*stage*', '*renal*', '*disease*', '*esrd*', '*patients*', '*receiving*', '*first*', '*permanent*', '*vascular*', '*access*', '*initiation*', '*hemodialysis*', '*identify*', '*variables*', '*determine*', '*assignment*', '*PTFE*', '*graft*', '*native*', '*AVF*'.

And for the second case the assigned terms would be: *'end-stage renal disease*', '*ESRD*', '*vascular*', '*PTFE*', '*graft*'. Being clear that using the second feature there are much less terms than using the tokens extracted from the documents text.

For now on we will refer to the first method as using *tokens*, and to the second method as using *annotated terms*.

[9] http://nltk.org/

Our initial expectations were that the results with the annotated terms would be better making it an advantage to use this approach.

Number of Dimensions/Topics

As we will see, the number of topics (or dimensions in the LSI case) will have a great influence on the results. As stated in Chapter 2, there is no optimal number of dimensions to keep when using LSI or topics in LDA, depending much on the size of the dataset and on the number of terms, being necessary to test it several times until we find results that better fit in our purpose.

Considering this, we tested our solution with different number of topics, from 100 to 1000 topics.

Similarity threshold

The similarity of the documents retrieved in our solution is another factor that can influence our results. If we consider, for instance, one document with a similarity score of 0.3, we don't know for sure, if that document is similar enough to the target test document and should be considered as a relevant document, or if it should be discarded. With this idea in mind, one condition in our experiment relies on applying two different thresholds on the retrieved documents to get the most similar ones. In one case we only hold documents with at least a 0.2 score of similarity and in the other case, we hold documents with 0.4 of similarity or greater. It is clear without analyzing the results, that in the second case we will get fewer similar documents, which will probably increase the precision of the results.

3.2.5. Performance Measures

To evaluate our results we used the performance measures precision, recall, f-score and mean average precision (MAP), using as Gold Standard the list of the MeSH terms actually assigned to each test document.

Precision, Recall and F-score are defined as follows:

$$Precision = \sum_D c(M, D) \Big/ \sum_D n(D) \tag{3.4}$$

$$Recall = \sum_D c(M, D) \Big/ \sum_D t(D) \tag{3.5}$$

$$F\text{-}score = \frac{2*Precision*Recall}{Precision+Recall} \tag{3.6}$$

Where $c(M, D)$ is the number of correct MeSH terms *M* predicted for a document *D*, and $n(D)$ is the total number of predicted MeSH terms for that document. $t(D)$ is the total number of MeSH terms assigned for the document *D* according to the Gold Standard.

Precision will then reflect how many of the MeSH terms predicted by our solution are correct, while recall will reflect how many of the MeSH terms really assigned were predicted. F-score computes de accuracy of our solution considering both precision and recall.

MAP is defined as:

$$AP(D) = \frac{1}{t(D)} \sum_{r} I(h_r) * \frac{c(r, D)}{r}$$
$$MAP = \frac{1}{|\Omega|} \sum_{D \epsilon \Omega} AP(D) \tag{3.7}$$

Where $AP(D)$ is the average precision of document D, with $c(r, D)$ being the number of correct MeSH terms among the top r ranked MeSH terms predicted for a document *D*, and $I(h_r)$ an indicator function, whose value is 1 if the r-th MeSH term h was predicted to the document and 0 otherwise. Ω is the document collection of the test dataset [18].

Computing time

Besides de regular performance measures, precision, recall, f-score and MAP, we also considered another factor that may be an evidence of the performance of our solution: the time needed to perform all the process. It won't affect directly the obtained results but it can be used as a tiebreaker if two approaches have very close results for the other measures.

It is important to note that this specific measure is only used when comparing the different approaches of our solution and the changes made from one experiment to another since we don't have information about the performance of the other solutions.

3.3. Results

The results obtained are presented in two different parts: first we tested the different models (LSI and LDA) and methods (tokens and annotated terms, different topics, etc.) concerning our solution, so we could conclude which approach was better. After choosing the best methods, we compared our solution to other studies.

3.3.1. Comparison of our methods

As stated, both for LSI and LDA, we trained the model on a corpus composed by the 50 nearest neighbors of each document of NLM2007, our test corpus. Thus, we perform six different experiments regarding the different models and methods we wanted to test (Table 3.1). We run each experiment 20 times, calculating the mean and standard deviation for each performance measure. These values are presented in Appendix A.

It is important to refer that in the cases of experiments 5 and 6, the two regarding LDA, we only could perform with a maximum of 500 topics, unlike the ones regarding LSI where we achieved the 1000 topics. This is due to the fact that the LDA tool of Gensim requires a much greater computational effort than LSI, which leads to memory problems.

Figure 3.6 to Figure 3.9 show the obtained results for the different performance measures, from which we can take some conclusions.

Table 3.1 Performed experiments regarding model, corpora processing and similarity threshold

Experiment	Model	Corpora processing	Similarity threshold
1	LSI	Annotated Terms	0.4
2	LSI	Annotated Terms	0.2
3	LSI	Tokens	0.4
4	LSI	Tokens	0.2
5	LDA	Annotated Terms	-
6	LDA	Tokens	-

Results for LSI

- **Number of topics**

As we can see, when using LSI, the results improve with increasing number of topics, especially until 400 topics. From that point they only get slightly better, starting even to decrease in some cases from 800 topics. This shows that the number of topics to use should be 400 or more, with no substantial differences from there on, being important to balance this with the computing time.

- **Similarity Threshold**

As expected, the results for precision, f-score and MAP are better when using a similarity threshold of 0.4. This means our solution performs better when considering a more restricted number of most similar documents. However, the results for recall are also higher, which is odd, since if we are retrieving fewer documents, the recall should have lower values. Yet, it is important to note that these measures are regarding the MeSH terms and not the documents. Thus, retrieving more (and less similar) training documents may help increasing the score of MeSH terms that are assigned to those documents, but not relevant for the target test document. Then, those MeSH terms will be incorrectly predicted to the target test document, lowering the recall values.

- **Annotated Terms and Tokens**

In this case, the experiments regarding the tokens present better results than the ones regarding the annotated terms. Nevertheless, the differences are not very significant, even matching at some points.

Using a dictionary of annotated terms instead of a regular tokenization of text, leads to having less features for computing LSI (in our case about 35% less) and the number of occurrences of each feature will be regularly lower which will possibly influence the LSI computing and similarity results and so, may be a reason for the presented performance results.

Nevertheless, since the difference between both experiments is not that significant, the use of the annotated terms is an option to consider.

- **Neighborhood frequency and similarity**

The two neighborhood features used for the ranking of MeSH terms don't present significant differences in neither the performance measures (Figure 3.10). In fact, when using higher number of topics, they almost match, being clear that is not relevant which feature is used.

Results for LDA

- **Number of topics**

In the case of the experiments using LDA, we have two kinds of results: when using the tokens in the corpora preprocessing (experiment 6), the results improve with increasing number of topics. We don't know if at some point over the 500 topics it would start decreasing, but due to the computational effort and consequently, the time it takes, at that point, it doesn't seems to be a good option as we will see further.

In the case of experiment 5, the performance results get worse with increasing number of topics.

- **Annotated Terms and Tokens**

As in the experiments regarding LSI, also here the use of annotated terms instead of tokens does not produce better results. However, in this case the differences are even bigger (more than 30% different in some cases), with experiment 5 having really poor results (13.8% for Precision and 25% for Recall).

Using documents formed by dictionary terms instead of their original text, we may be reducing the complexity of the text structure, influencing negatively the LDA process. This may be a reason for the poor results when using the annotated terms.

LSI versus LDA

All experiments regarding LSI present better results than the ones regarding LDA. Since the LDA model is an alternative to pLSI, which is an evolution of LSI, these results are somehow unexpected. We already mentioned a possible reason for this, mainly regarding the use of dictionary terms. Yet, it is also possible that our approach for applying LDA to this specific problem may not be the most correct one, leading to worse results.

Computing time

As we can see in Figure 3.11, the computing time of each solution increases along with the number of topics. As stated, LDA needs a greater computational effort to perform so it is easy to understand why it computing time is greater than LSI.

Since when using the tokens method the number of features is significantly bigger, the computing time in these situations is naturally higher. However in the case of LDA the difference is much more notorious, taking as much as 9 minutes to execute when using 500 topics.

It is important to note that these values refer to the whole process. However, the LSI or LDA processing presents lower values. This is an important aspect, since both LSI and LDA are processed offline, which means they only have to be executed once.

When using LSI, with the dictionary terms, on average 33% of the total time is for the LSI factorization, while with the tokens, this value is around 43%. The rest of the process is mainly for calculating the similarity between the target test document and each training document, also increasing with the number of topics.

When using LDA, the time concerning only its process is, on average, about 65% for the two cases: tokens and the dictionary terms. The complete results regarding the computing time are presented in appendix A.

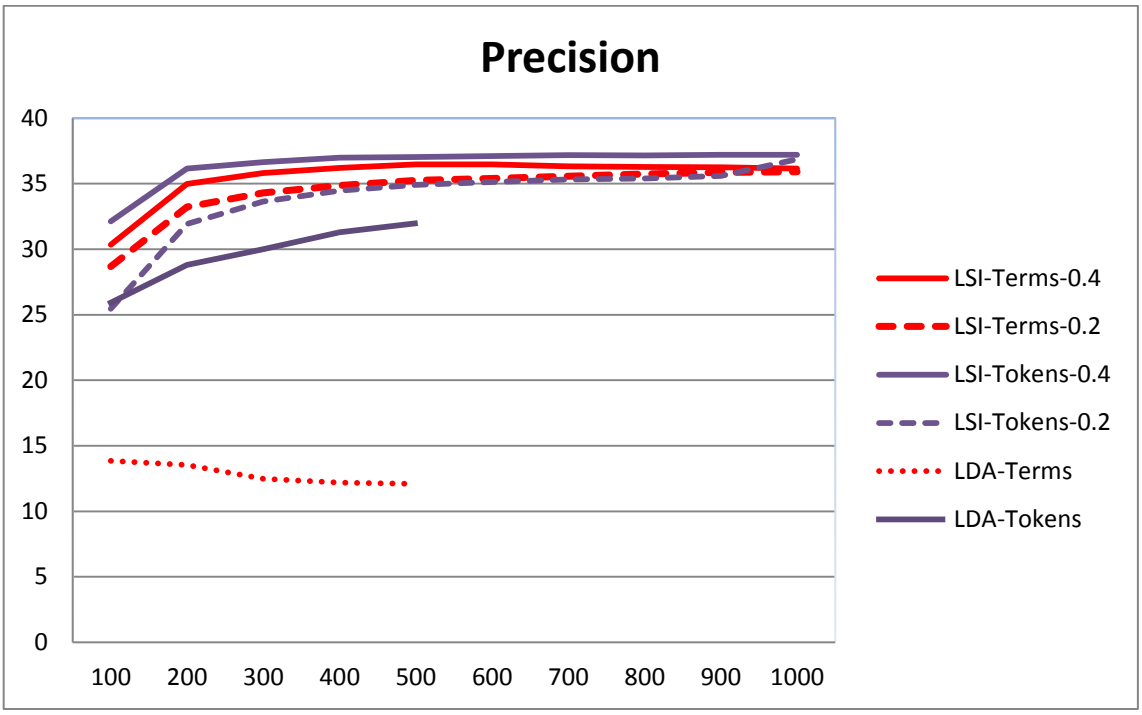


Figure 3.6 Illustration of the obtained results for Precision: red lines represent use of tokens and purple lines the use of annotated terms. Full lines represent LSI with similarity threshold of 0.4, dashed lines represent LSI with similarity threshold of 0.2 and dotted lines represent LDA.

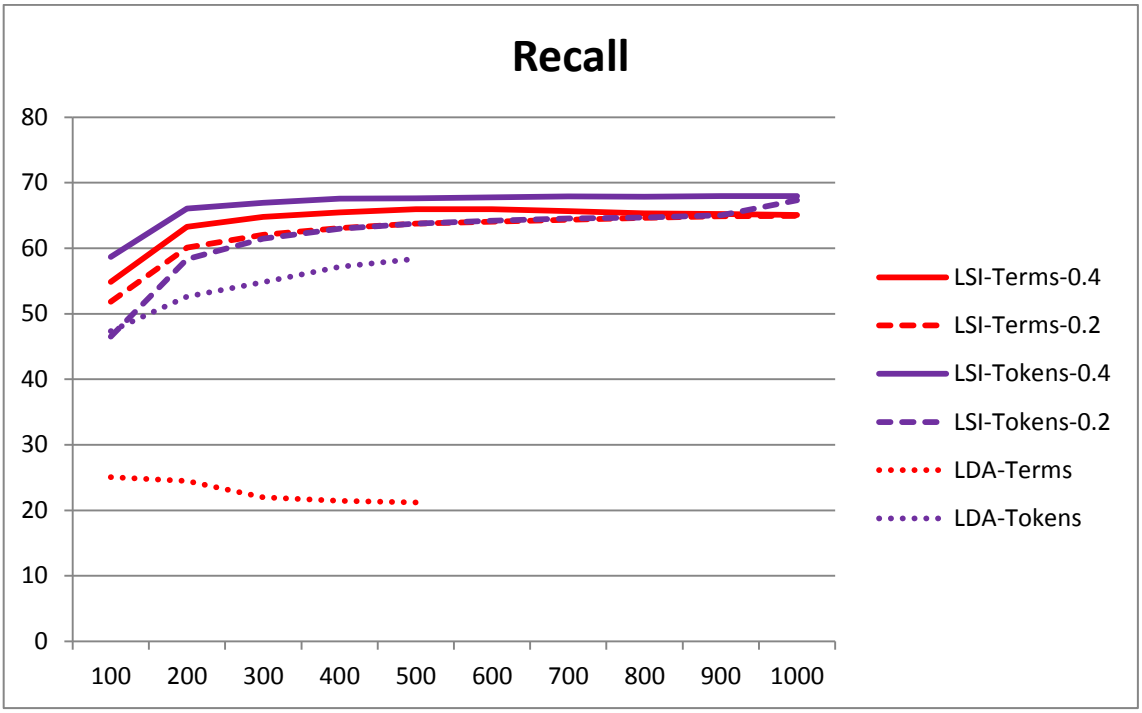


Figure 3.7 Illustration of the obtained results for Recall: red lines represent use of tokens and purple lines the use of annotated terms. Full lines represent LSI with similarity threshold of 0.4, dashed lines represent LSI with similarity threshold of 0.2 and dotted lines represent LDA.

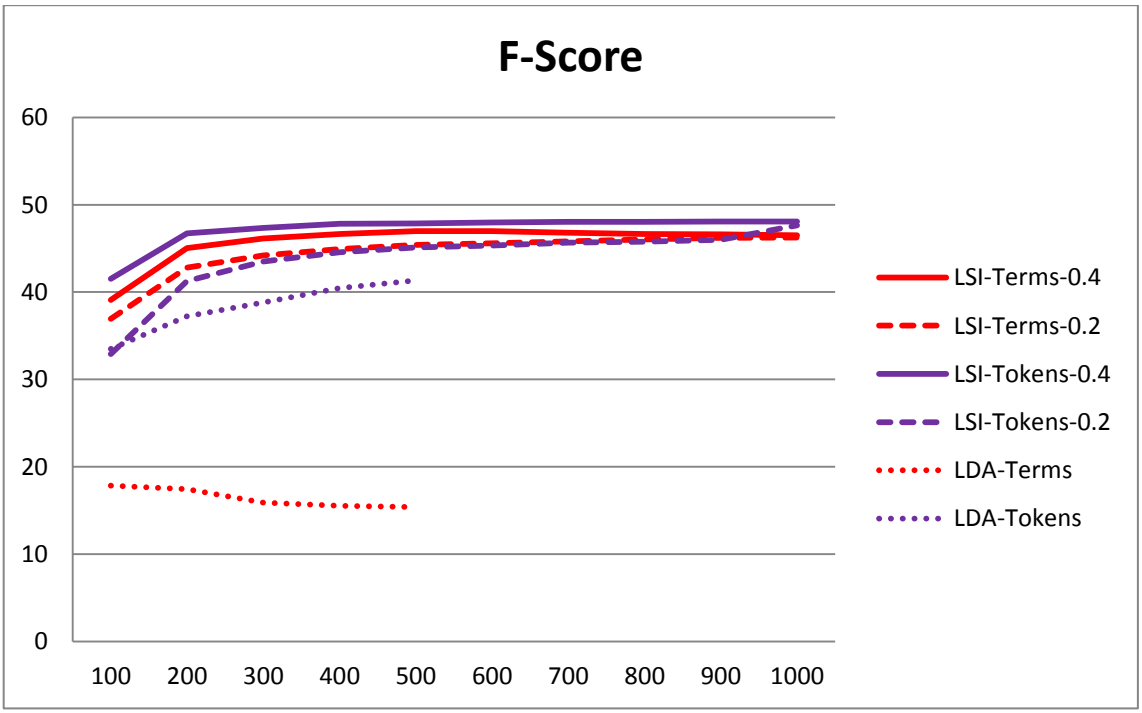


Figure 3.8 Illustration of the obtained results for F-Score: red lines represent use of tokens and purple lines the use of annotated terms. Full lines represent LSI with similarity threshold of 0.4, dashed lines represent LSI with similarity threshold of 0.2 and dotted lines represent LDA.

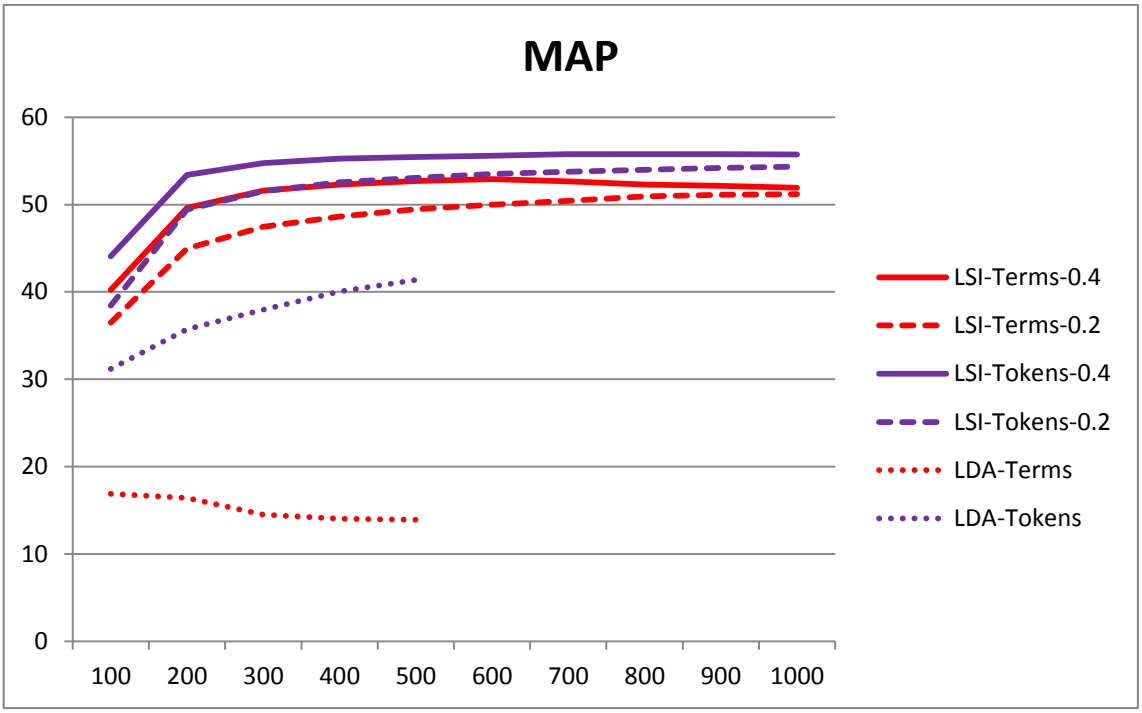


Figure 3.9 Illustration of the obtained results for MAP: red lines represent use of tokens and purple lines the use of annotated terms. Full lines represent LSI with similarity threshold of 0.4, dashed lines represent LSI with similarity threshold of 0.2 and dotted lines represent LDA.

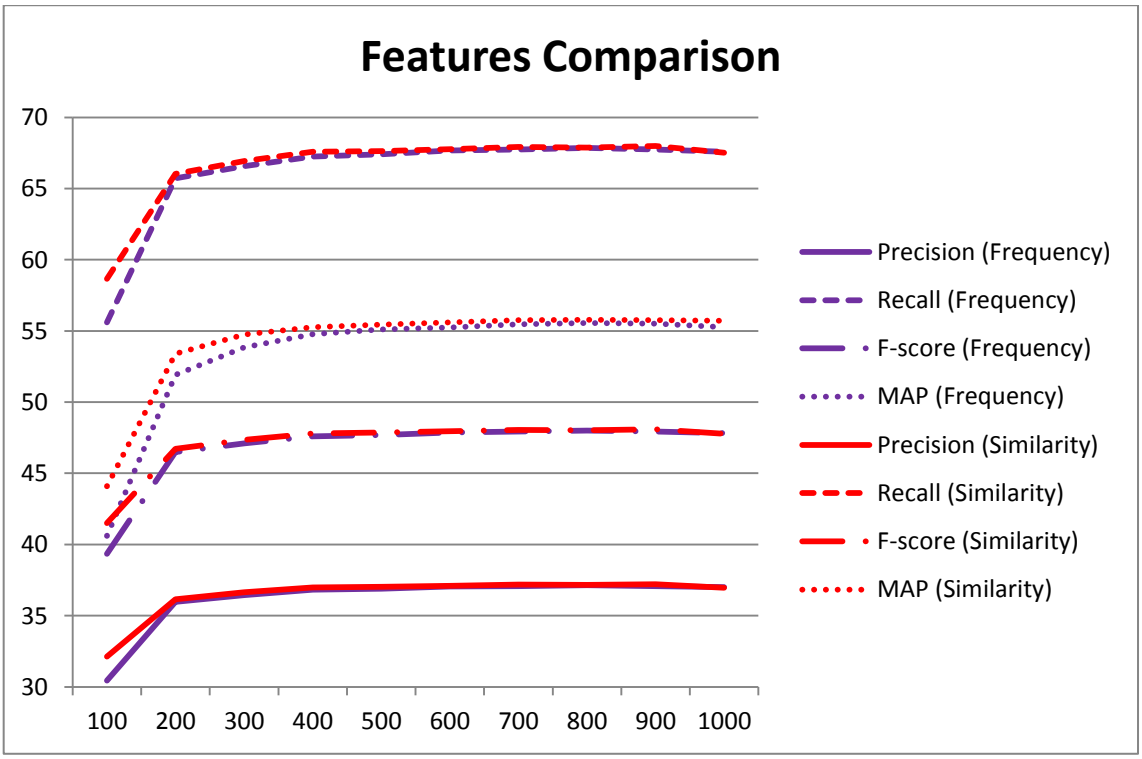


Figure 3.10 Comparison between the two neighborhood features used for ranking MeSH terms: Frequency and Similarity. The example in the illustration uses the LSI model with tokens and similarity threshold of 0.4.

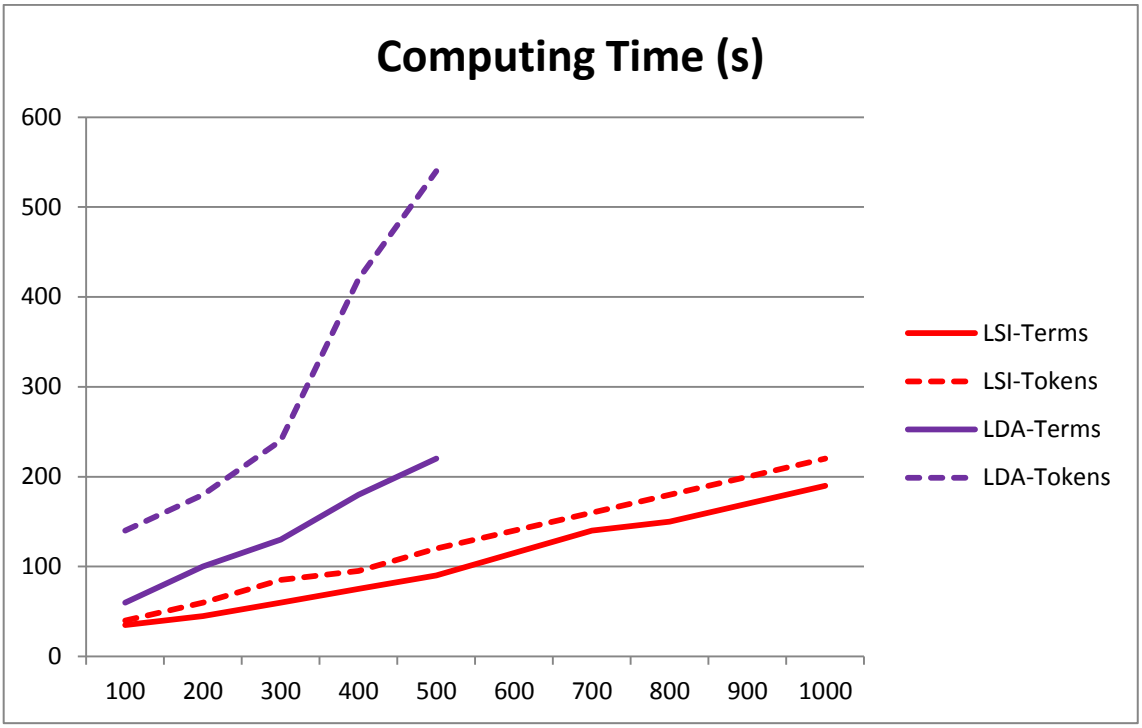


Figure 3.11 Illustration of the processing time needed by each experiment: red lines refer to LSI and purple lines to LDA. Full lines refer to experiments using tokens and dashed lines to annotated terms.

3.3.2. Comparison to other methods

According to our results, we concluded that experiment 3, using LSI with, a similarity threshold of 0.4 and a regular tokenization of the documents is the best solution. Despite the difference isn't very significant, the results for 900 topics are the best ones, as we can see in appendix A, and so we consider those to our next step.

Thus, for the next step, we compared our approach to five methods (Table 3.2), including the Learning-to-rank algorithm, presented in [18] and other four methods also used for comparison in the same article: the NLM's MTI system [22], a system using reflective random indexing to find similar documents [23] and two baseline ranking strategies based on neighborhood features: neighborhood frequency and neighborhood familiarity.

We also did two different comparisons regarding the Learning-two-rank: one considering its results with all the features, and another considering its results with only the neighborhood features.

As we can see in Table 3.2, our approach has similar results to the two baseline strategies (Neighborhood frequency and Neighborhood familiarity) and to the learning-to-rank algorithm, when using only the neighborhood features. In the best case, although precision and recall are similar, the value for MAP is lower, which points to the benefits of using a learning-to-rank algorithm. All these methods present better results over the two others, Reflective Random Indexing and especially the MTI system. Only the Learning-to-rank algorithm when using all features reveals an improving over all the other methods.

Table 3.2 Precision, recall, f-score and MAP for different methods including our approach.

Method	Precision	Recall	F-score	MAP
MTI system	0.318	0.574	0.409	0.450
Reflective Random Indexing	0.372	0.575	0.451	N/A
Neighborhood frequency	0.369	0.674	0.476	0.598
Neighborhood familiarity	0.376	0.677	0.483	0.604
Learning-to-rank (all features)	0.390	0.712	0.504	0.626
Learning-to-rank (neighborhood features)	0.370	0.677	0.478	0.602
LSI-tokens-0.4 (our solution)	**0.372**	**0.679**	**0.481**	**0.558**

3.4. Conclusions

One of the goals of this project was to discover useful applications for LSI, especially in the biomedical domain. So, in this chapter we conducted an experiment to test LSI in such task, by using it for recommending MeSH terms for annotating biomedical articles.

We used LSI to retrieve the most similar documents for a target document, in a process that gets the MeSH terms from those documents and through a ranking algorithm selects the most relevant ones to the suggest MeSH terms for the target document.

We tested not only LSI, but also LDA as the model for getting the most similar documents. Besides, other factors were also considered, namely corpora preprocessing, the use of tokens versus terms from a dictionary, number of topics created by the model, similarity threshold and computing time.

The results show that our approach does not have an improvement in recommending MeSH terms comparing to other methods. Our results are similar to two baseline strategies, and worse than the Learning-to-rank algorithm.

Another important issue about our experiment which we believe has conditioned the performance results is related to the corpus used to train our model. As stated in section 3.2.4, we used a dataset formed by the 50 nearest neighbors of each document of the test corpus, since they were the ones used in the other studies. Though, since they are already similar in some way to the test documents, we believe that such specific set may have limited LSI performance making its effect less meaningful.

Despite not presenting an improvement over the state of the art, LSI shows an identical behavior and similar results, which makes it a valid alternative to consider for recommending MeSH terms. More experiments should be conducted with this and other datasets in order to verify and try to improve these results.

4. Structuring Literature Search using LSI

In line with the previous chapter, we continue exploring LSI's capabilities and utility in various domains with special attention to the biomedical domain.

In this chapter, our main purpose is to use LSI as a retrieval technique and discover different topics among the retrieved results. When using LSI for getting the most similar documents for a specific query, instead of having one list of relevant documents, we could have several lists, each related with a different subject or theme.

To achieve this, a set of experiments were conducted, exploring the regular results of LSI and Cosine Similarity. New methods to group in different topics, the most similar documents for a query were developed.

4.1. Overview

In chapter 2 we describe some issues about information retrieval systems, such as the possibility of a searched term having more than one meaning (polysemy) or multiple terms having the same meaning (synonymy). We discussed ways to solve or minimize these problems, recurring to some retrieval techniques, including LSI.

However, even solving polysemy or synonymy issues, it would be interesting to improve the results by retrieving more specific documents for a query having in count different subjects or topics. For instance, if we search for the term "dopamine" on a biomedical literature database, it can be referred in the scope of an Alzheimer's disease treatment or a new drug discovery. Using LSI (or other retrieval methods) we will get the most similar documents for "dopamine" in a general way, without distinguishing these different subjects.

Also in chapter 2 we demonstrated how LSI proceeds, from building the co-occurrence matrix to the use of cosine similarity. As we have seen, with cosine similarity it is possible to retrieve a list of documents ranked by their similarity for a specific query. This is done by calculating the distance between the query and the documents in a vector space, regarding all the dimensions previously computed by SVD.

Looking back to Figure 2.7, an illustration of the documents and terms of a sample corpus mapped in a vector space, we can see that one of the classes of documents is more relevant to dimension 1 (class *c*) and the other is more relevant to dimension 2 (class *m*). So, apparently if somehow we compute the similarity between a query and the documents for each one of the dimensions instead of all of them, we will have different results: for dimension 1 the retrieved documents will be based on class *c* and for dimension 2 they will be based on class *m*, having the most similar documents for two distinct topics. This way, we can have different classes or clusters of documents which are similar to the query in different ways. This can be an improvement to LSI results since instead of having a list of similar documents, we can have several different lists for different subjects.

For now on, we will use the terms LSI dimensions and topics interchangeably. However, it is important to note that our purpose with this experiment is not to use LSI as a Topic Model (at least not in the same way as other techniques like pLSI or LDA), since we are not using LSI to create new topics through probabilistic methods but disassembling its process and discovering different sets of documents associated to the several dimensions (topics).

4.2. Clustering similar documents by topic

This experiment has two phases: the first one is to find a way of clustering and ranking the most similar documents for a query by each LSI topic (Figure 4.1). First we use LSI over a corpus to create the vector space and then, we use one of three different methods to group the most similar documents for the query by each topic.

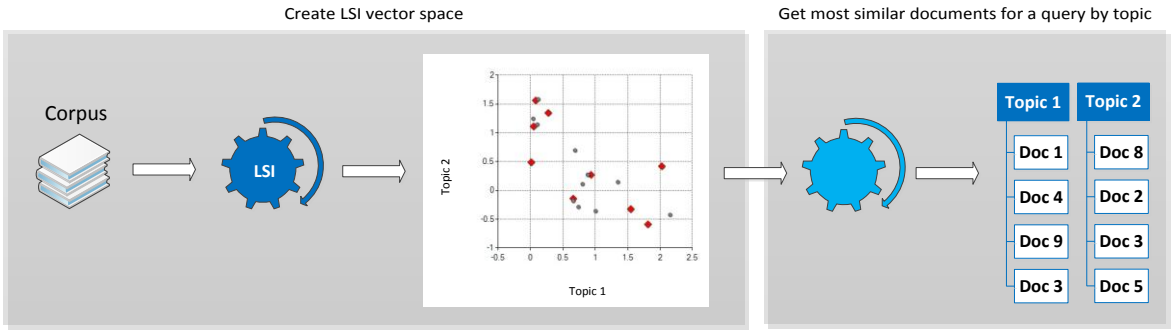


Figure 4.1 Process to get most similar documents by topic

4.2.1. Method 1

Cosine similarity calculates the dot product of two vectors (query and document) reflecting the angle between them in a vector space. Our first approach for finding the most similar documents is simply to project the documents and query values in each LSI dimension, calculating the distance between them, as follow:

$$Score(D,T) = |V(Q,T) - V(D,T)| \quad (4.1)$$

Where $V(Q,T)$ is the value of the query *Q* for topic *T*, $V(D,T)$ the value of document *D* for topic *T* and $Score(D,T)$ the score of document *D* for topic *T*, used to rank the documents. Figure 4.2 illustrates the projection of the documents and query in the different topics.

Then, for each of the topics we have a different rank of documents based on their proximity to the query (Table 4.1).

Table 4.1 Ranking of similar documents for the query q regarding each topic.

Topic 1	Topic 2
Doc A	Doc C
Doc F	Doc B
Doc E	Doc F
Doc B	Doc A
Doc D	Doc E
Doc C	Doc D

Despite being easy to understand, this approach presents some issues. First of all, the query will not be equally relevant for all topics, which means that we should take in consideration the position of the query in the vector space considering each topic. Looking again to Figure 2.7, we are able to see that, in this specific example, the query *q* is almost entirely related with topic 1, having a very low value regarding topic 2 and therefore being almost unrelated with this topic.

To minimize this issue, we define a threshold to consider only the topics for which the query is relevant (i.e. if the query value for that topic is above the threshold).

It is important to state that until now we are using small examples with small corpora and few topics. We will face other issues when testing with bigger corpora, but we will discuss that further.

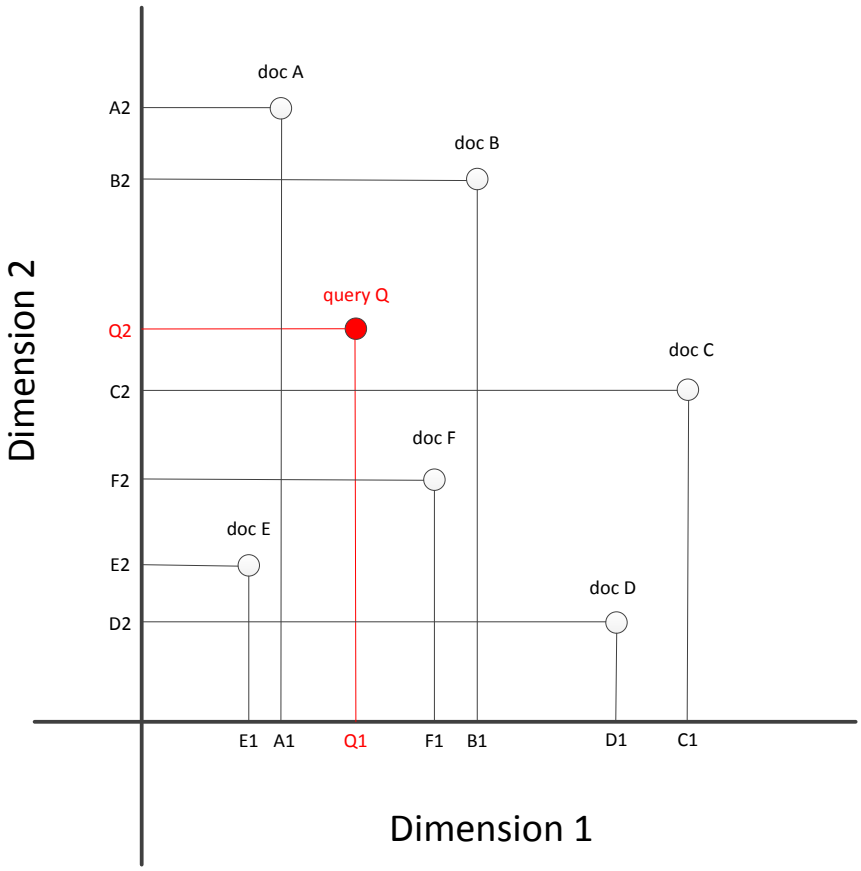


Figure 4.2 Projection of each document and query values in each topic.

4.2.2. Method 2

Regarding the first method, there is an issue concerning the relevance of each topic for the query, being necessary to define a threshold considering only the topics for which the query is most relevant. However, there is another important factor to take into account: the relevance of the documents for each topic. According to the example in Figure 4.3, by the previous method document A would be more similar to the query Q than document C in topic 1. Yet, document A is very little related with topic 1 and should not be considered.

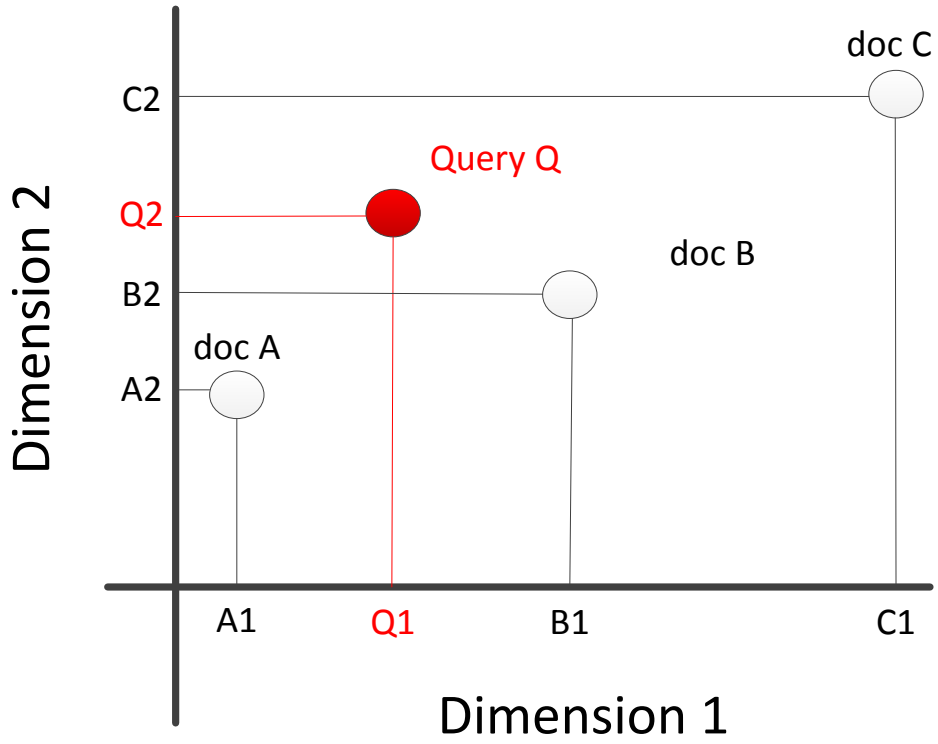


Figure 4.3 Projection of each document and query values in each topic. For topic 1, document A is closer to the query than document C, which according to the first method, makes it more similar to the query.

Regarding this, we developed a second approach where instead of calculating the difference between the query and the documents for each topic, we calculate a score based on the product of the query value and each document value, as follows:

$$Score(D,T) = |V(Q,T) * V(D,T)| \tag{4.2}$$

Where $V(Q,T)$ is the value of the query *Q* for topic *T*, $V(D,T)$ is the value of document *D* for topic *T* and $Score(D,T)$ is the score of document *D* for topic *T*. This way, if the query, or the documents are unrelated to the topic, their value will be very small and consequently, the calculated score will also be low. After this, we can define a threshold to consider only documents with a higher score, having a set of most similar documents for a query regarding each topic.

4.2.3. Method 3

Method 2 overcomes an issue identified for Method 1. However, another issue becomes clear. The score calculated for each document reveals the relevance of the query and the document for a specific topic, but not the similarity between the query and the document.

In other words it shows that both query and document are relevant for that specific topic, but it doesn't mean that the query and the document are themselves similar. For instance, in Figure 4.3, using the second method, document *C* will have the highest score for topic 1, since it is the most relevant document for that topic.

But document *B* is closer to the query *Q* and so is similar to it, despite not being as relevant to topic 1 as document *C*.

For that reason we considered a third method as an alternative to the other two. The main difference is the use of another factor for calculating the document score: the similarity value between the query and the document when using cosine similarity in the common way (i.e. considering all the LSI dimensions).

Even though we're calculating similarity for each topic separately, we do not discard completely the global similarity between documents and query. Thus, besides considering the relevance of documents and query for each topic, as in method 2, we also consider the similarity between documents and query.

First of all, we verify if the value of the query *Q* for topic *T*, $V(Q,T)$, is greater than a first threshold, filtering the topics for which the query is relevant.

Then, considering only this set of dimensions, we calculate each document score as follows:

$$Score(D,T) = |Sim(D,Q) * V(D,T)| \tag{4.3}$$

Where $Sim(D,Q)$ is the similarity value of document *D* for the query *Q*. Finally we used a second threshold to filter these scores, obtaining the most similar documents for the query regarding each of the considered topics.

4.3. Identifying Representative Terms

The second phase of this experiment is to evaluate each method in order to conclude which suits better our purpose. Since our goal is to create topics of documents, each one related to a specific subject, then each topic should contain related documents, and the topics should be as distinct as possible from each other. Hence, this phase relies on assessing the dissimilarity between topics and so, comparing the capability of each method to create topics with distinct subjects.

Our first approach to conduct this evaluation was simply to see the documents associated to each topic and compare the number of unique documents of each one. However, this approach is too naïve, because, despite knowing the documents that are unique to each topic, that does not tell us if those documents have any kind of similarity between each other.

So it became clear that to evaluate our methods, we needed a way to assure the documents are really similar and related to each other. Therefore, we use the MeSH terms assigned to each document, as gold standard to know if they are similar when comparing each other: if two documents have exactly the same MeSH terms assigned to them, they will have very similar content. Conversely, we also want documents in different topics to be dissimilar, that is, they should have different content and therefore different MeSH terms associated to them.

In short we have, for each topic, a list of most relevant documents (defined by one of three different methods) and for each document of the corpus a list of assigned MeSH terms. Our next step was, according to this data, to create for each topic, a list of most relevant MeSH terms (Table 4.2). Then we compared each topic to the others, analyzing their dissimilarity regarding the assigned MeSH terms. Figure 4.4 shows an overview of this process.

The ranking of MeSH terms by relevance for each topic is the most complex part of this phase, which again led to the evaluation of different strategies, to achieve that ranking.

Table 4.2 Example of most relevant MeSH terms by topic with score.

Topic 2		Topic 23		Topic 74	
Dopamine/*physiology	0.42	Dopamine/*metabolism	0.27	Brain/*physiology	0.33
*Semantics	0.39	Serotonin/*metabolism	0.14	Dopamine /*metabolism	0.32
Reaction Time/drug effects/*physiology	0.33	Motor Activity /*physiology	0.13	*Nerve Tissue Proteins	0.12
Dopamine/*metabolism	0.23	Scrapie/*metabolism /psychology	0.13	*Membrane Glycoproteins	0.11
Parkinson Disease/drug therapy/*physiopathology	0.17	Avoidance Learning /*physiology	0.12	*Membrane Transport Proteins	0.10
…		…		…	

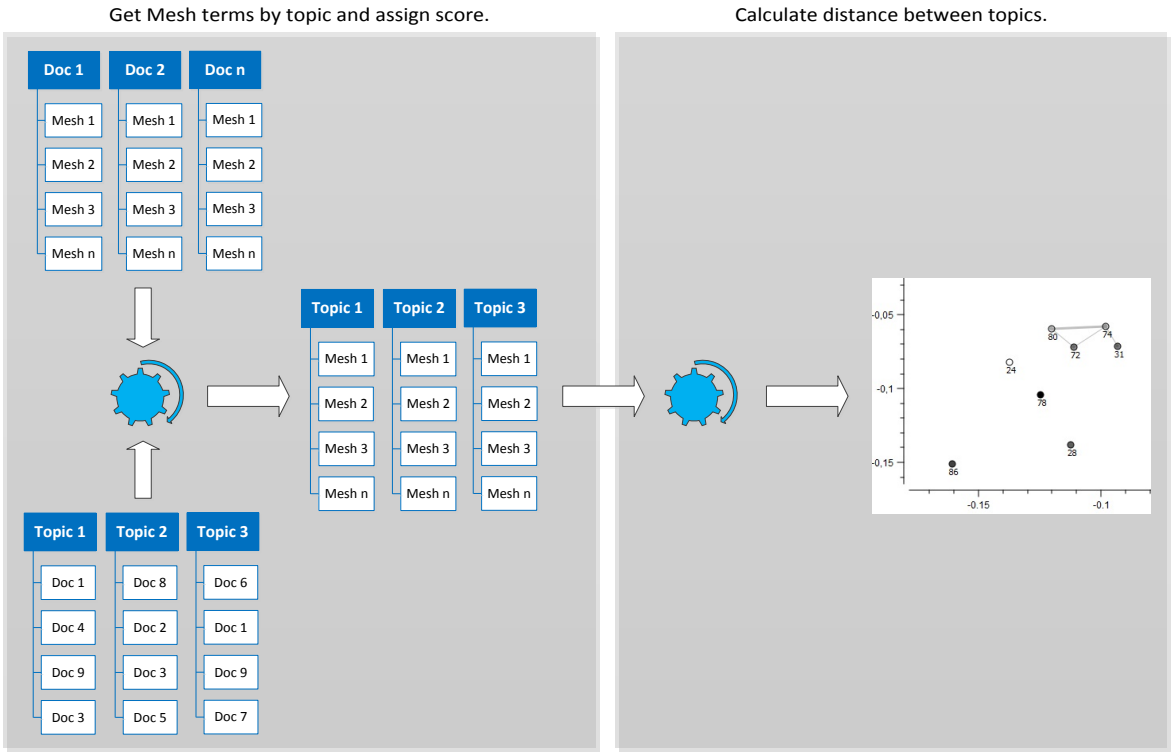


Figure 4.4 Overview of process to calculate topics dissimilarity.

4.3.1. Ranking of MeSH Terms

Strategy 1

Our first strategy is to, for each MeSH term, simply count the number of documents in which that term appears, among the list of relevant documents for a specific topic, as follows:

$$Score(M_i, T_j) = |N(D), M_i \in D, D \in T_j| \quad (4.4)$$

Where $Score(M_i, T_j)$ is the score of the MeSH term M_i for the T_j and $N(D)$ is the number of documents assigned with M_i and in the list of relevant documents of T_j.

Strategy 2

Our second strategy is similar to the first one, but instead of counting the number of documents, we summed their similarity scores for the considered topic:

$$Score(M_i, T_j) = \sum_{k=1}^{D} Sim(D_k, T_j), M_i \in D_k, D_k \in T_j \quad (4.5)$$

Where, $Sim(D_k, T_j)$ is the similarity score of D_k for topic T_j.

Strategy 3

The two firsts strategies only consider if the MeSH terms are assigned to the documents or not. Yet, the documents ranking is also an important aspect, since the top ranked ones should be consider more relevant. For instance, when using the two first strategies, if the same document appears in two topics, being the most relevant for topic 1 and the 10th most relevant for topic 2, it will have the same weight in both topics. But if we consider its position in the ranking, it will have a higher weight in topic 1.

Thus, our third strategy is based on the first one, but summing over the reciprocal of each document rank value instead of counting, as follows:

$$Score(M_i, T_j) = \sum_{k=1}^{D} \frac{1}{Rank(D_k, T_j)}, M_i \in D_k, D_k \in T_j \quad (4.6)$$

Where $Rank(D_k, T_j)$ is the ranking position of document D_k for topic T_j. This way the better ranked documents will contribute with higher values for the score of MeSH term M_i.

Strategy 4

Our last strategy, as the previous one, uses the documents ranking to calculate the score of each MeSH term. It is based on the second strategy, but dividing the similarity value by the ranking position:

$$Score(M_i, T_j) = \sum_{k=1}^{D} \frac{Sim(D_k, T_j)}{Rank(D_k, T_j)}, M_i \in D_k, D_k \in T_j \quad (4.7)$$

Where $Sim(D_k, T_j)$ is the similarity score of document D_k for topic T_j and $Rank(D_k, T_j)$ the ranking position of document D_k for topic T_j.

4.3.2. Topic Dissimilarity

After getting the most relevant MeSH terms for each topic, we need to measure the dissimilarity between topics to evaluate how well they identify different clusters of documents, hence different subjects. To measure the distance between topics, we represent each topic as a vector of MeSH terms, where each entry is the score for the MeSH terms in each topic. We then calculate the inner product of each pair of topics as follows:

$$Dist(T_i, T_j) = 1 - \frac{A \cdot B}{\|A\|\|B\|} = 1 - \frac{\sum_{k=1}^{m} Score(M_k, T_i) * Score(M_k, T_j)}{\sqrt{\sum_{k=1}^{m} Score(M_k, T_i)^2} * \sqrt{\sum_{k=1}^{m} Score(M_k, T_j)^2}} \quad (4.8)$$

Where $Score(M_k, T_i)$ is the calculated score of MeSH term M_k for the topic T_i and $Dist(T_i, T_j)$ the distance between topic T_i and topic T_j. This way, we will have a matrix with the distances for each pair of topics (Table 4.3). Thereafter we can display those distances in different ways, as we will see further, making it easier to identify and interpret the associations between the topics.

Table 4.3 Example of calculated distances between topics.

	Topic 2	**Topic 12**	**Topic 20**	**Topic 23**	**Topic 24**	**Topic 25**	**…**
Topic 2	-	0.17746	0.19252	0.21546	0.11444	0.19465	…
Topic 12	0.17746	-	0.21325	0.19248	0.12448	0.22247	…
Topic 20	0.19252	0.21325	-	0.12307	0.06705	0.26656	…
Topic 23	0.21546	0.19248	0.12307	-	0.06592	0.21226	…
Topic 24	0.11444	0.12448	0.06705	0.06592	-	0.1494	…
Topic 25	0.19465	0.22247	0.26656	0.21226	0.1494	-	…
…	…	…	…	…	…	…	…-

4.4. Results

After describing the developed methods and strategies to perform this experiment, we conducted a set of tests to compare the different methods and analyze the obtained results.

In short we have three different methods to get the most relevant topics for the query and the most relevant documents by topic, and four different strategies to get the most relevant MeSH terms by topic. This makes twelve different experiments, identified in Table 4.4.

Table 4.4 Different experiments concerning the developed methods and strategies.

	Strategy 1	Strategy 2	Strategy 3	Strategy 4
Method 1	Experiment 1.1	Experiment 1.2	Experiment 1.3	Experiment 1.4
Method 2	Experiment 2.1	Experiment 2.2	Experiment 2.3	Experiment 2.4
Method 3	Experiment 3.1	Experiment 3.2	Experiment 3.3	Experiment 3.4

To conduct this set of experiments we used a corpus about neurodegenerative diseases constituted by nearly 135 thousand documents (title and abstract) extracted from PubMed. The Medline query used to obtain the documents was: "Neurodegenerative Diseases"[MeSH Terms] OR "Heredodegenerative Disorders, Nervous System"[MeSH Terms]. Articles in languages other than English or not containing an abstract were discarded. The fact that the documents are indexed in PubMed allows us to use their MeSH terms, essential for the second phase of the experiment. As in chapter 3, we used a dictionary of annotated terms, being each document composed by those terms instead of processing the original text.

As seen in previous chapters, one important feature of the LSI process is the number of topics. In a first stage, we used 100 topics, for comparing the different methods. After selecting the best approach, we varied the numbers of topics for a more complete analysis.

Singularly, we used the query "*dopamine*" for all the experiments, which means that all the results concern that term. After determining the best solution we tested it with different queries.

4.4.1. Strategies for ranking MeSH terms

In this section we compare the different strategies for ranking the MeSH terms within each topic. The results obtained are presented in the form of boxplot diagrams. An example of a matrix with the numeric values of the distances is given in Appendix B.

Each chart represents the 25^{th},50^{th} and 75^{th} percentile of the distances between each topic and the others. The end of the whiskers represents the higher and lower values for each topic. It is important to note that the shown topics are the ones consider relevant for the query by the tested method which may change for the other methods.

These graphs allow identifying the best strategy, since the objective is to obtain topics which are dissimilar between each other.

Method 1

Figure 4.5 and Figure 4.6 show the results of experiments regarding the first method in the form of boxplot diagrams. The diagrams for strategy 1 and 2 are not shown because the distances values are very low (varying between zero and $5{,}0 \times 10^{-5}$), which would make the graph hard to interpret.

The fact that experiments 1.3 and 1.4 have much higher values than the other two, means that the distances between topics are generally higher and so, the topics are less similar to each other in these cases. Since the main feature of those experiments is the use of the ranking value for calculating the final score of each MeSH term, we can conclude that is a fundamental factor to take into account.

Without considering the ranking position of the documents, it is the same to have document A as the 10^{th} most relevant or as the 100^{th}. This increases the probability of two topics being similar, since they only need to have the same documents in their most relevant list. On the other hand, using the ranking value, besides having the same documents, they would have to be equally ranked. The rationale for including the ranking is that, from an information retrieval perspective, it is more important to distinguish the documents at the top of the list, since these are the ones the users will consider.

Nevertheless, comparing experiments 1.3 and 1.4 we can see that 1.4 presents better results. This shows that the documents similarity score is also an important factor, affecting the similarity between topics, since each document has different scores for each topic.

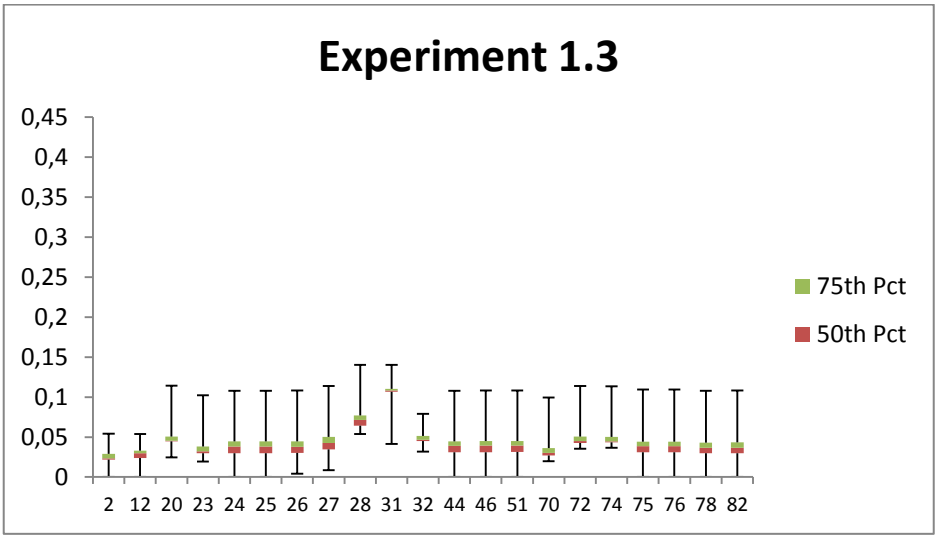


Figure 4.5 Dissimilarity measure between each topic and all the others, using boxplot diagrams. Results for experiment 1.3.

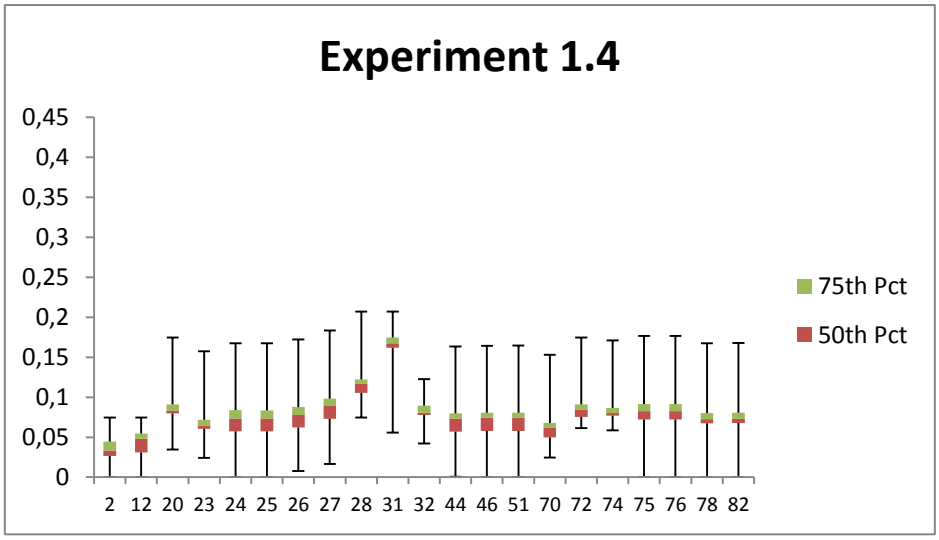


Figure 4.6 Dissimilarity measure between each topic and all the others, using boxplot diagrams. Results for experiment 1.4.

Method 2

The second method calculates the document score through the product of the query similarity value and the document similarity value for each topic. Figure 4.7 shows that, as in method 1, experiment 2.4 presents the best results followed by 2.3, which highlight the importance of the documents relevance score and mainly the documents ranking. Though, these differences are not as evident as in method 1.

One important aspect about the experiments regarding method 2 is the fact that the plots of each topic are less uniform than in method 1. This is due to the number of relevant documents of each topic, which is more irregular, being just one document in some cases, as for instance topic 12. With only one document, the MeSH terms assigned to that topic will be much more limited, making it totally similar to topics with also that small set of MeSH terms, having a dissimilarity score of zero, and totally dissimilar to topics which don't have those terms. This is quite evident in the plots presented in Figure 4.7, with some of the topics having the lower whisker reaching the zero value and the rest with the higher whisker reaching much higher values. Despite this explanation we will discuss the differences between methods further with more detail.

Method 3

Plots regarding method 3 (Figure 4.8) show, once more, the importance of the document ranking and document relevance score, with experiments 3.3 and 3.4 presenting the better results. Yet, the difference for experiments 3.1 and 3.2 are more evident than in method 2, with the plots presenting very low values.

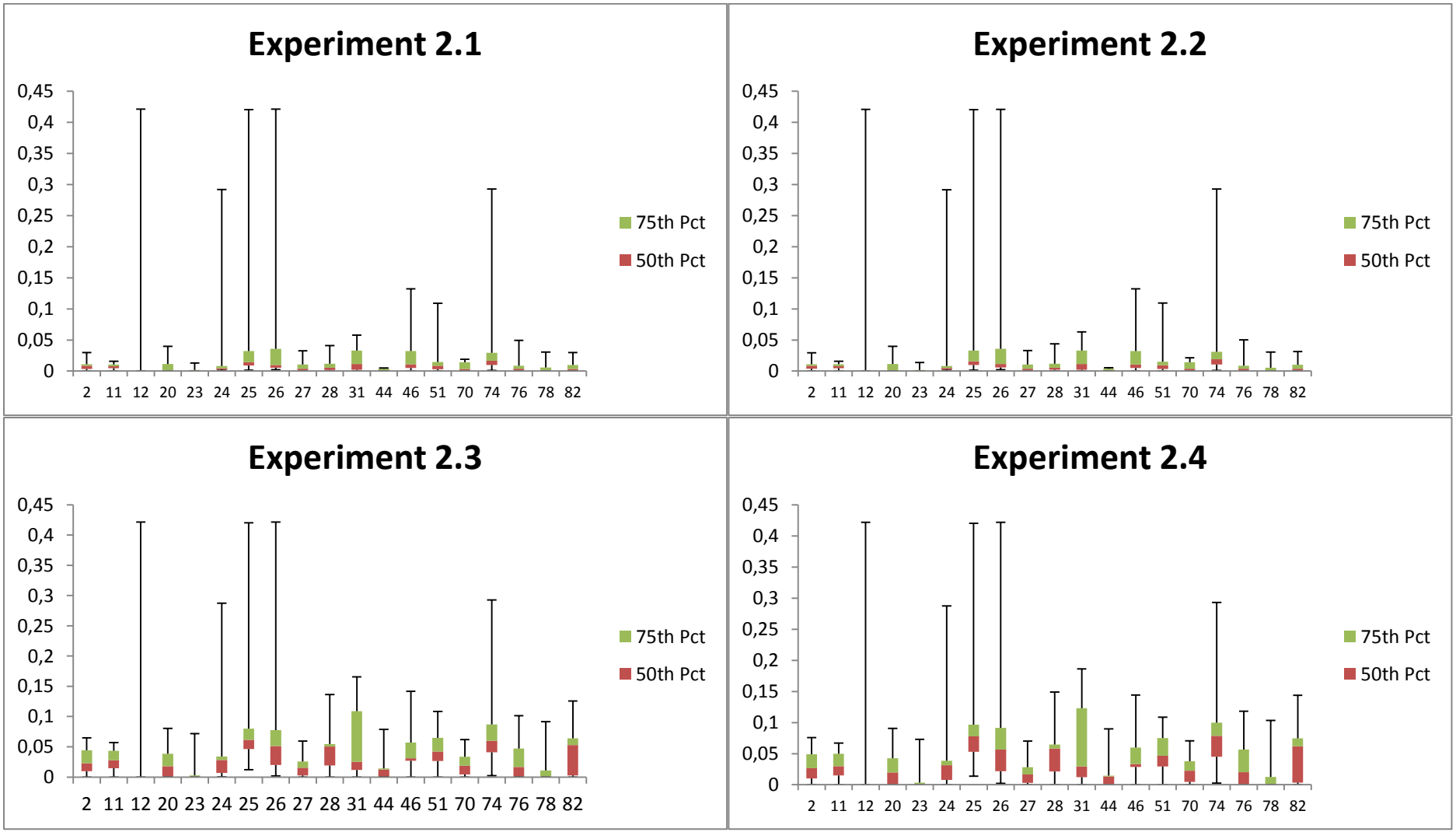


Figure 4.7 Dissimilarity measure between each topic and all the others using boxplot diagrams. Results of experiments regarding Method 2.

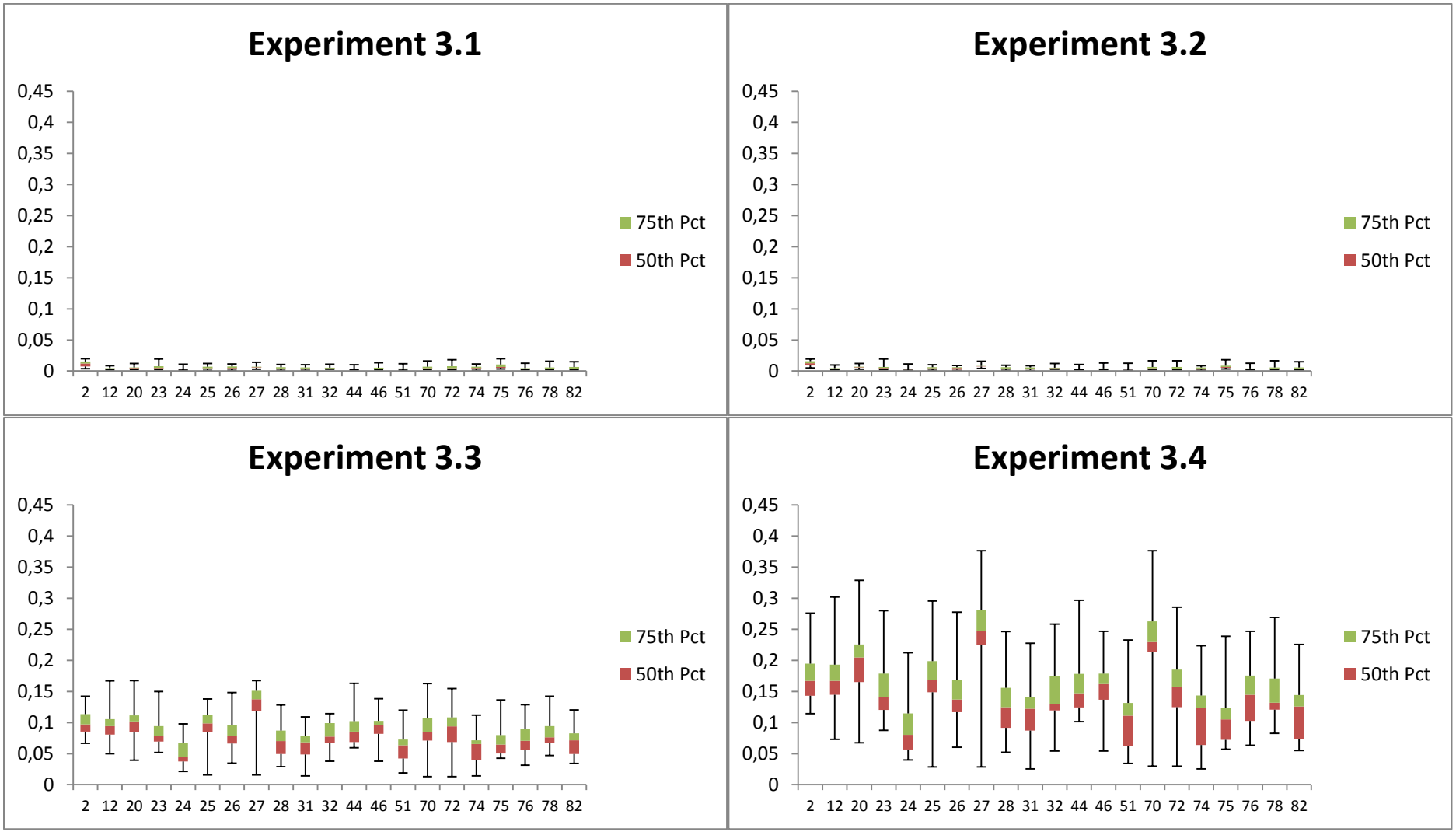


Figure 4.8 Dissimilarity measure between each topic and all the others, using boxplot diagrams. Results of experiments regarding Method 3.

4.4.2. Methods Comparison

It is clear that, no matter what method we are using, the documents considered relevant are almost the same in every topic, which causes the two first strategies to result in apparently very similar topics. However the documents are not always ranked in the same way across topics, and when we consider the documents ranking and relevance score for getting the most relevant MeSH terms, the topics can be better discriminated. Hence, after comparing the different ranking strategies for each method, we compared our methods using only strategy 4.

When describing each method we mentioned the use of a threshold for selecting the documents considered relevant for each topic. This threshold had the same value for all the methods, when conducting the several experiments. Thus, only documents with a score above that value are considered relevant and topics with no relevant documents are discarded. We also defined another threshold to previously select the most relevant topics for the query. Therefore, for each experiment we have a limited number of topics, instead of the initial 100.

Besides this one, we used other thresholds: in method 1, we use a threshold to select the closer document to the query for a particular dimension and in method 3 we use another for selecting the most similar documents for the query, given all LSI dimensions (we discarded the ones with negative similarity score, which means the ones which were not similar at all).

Figure 4.10 shows our results for the three different methods using boxplot diagrams. Analyzing the plots we can see that method 3 presents the best results, with the topics having higher median values, which means they are usually more dissimilar from each other than using other methods. We can also see that in method 3 the lower whisker never reaches the zero value unlike method 1 and especially method 2. This means that with method 3, in no case two topics are completely similar (i.e. have exactly the same documents and ranked equally).

Another aspect already mentioned in the case of Method 2, is the fact that the results for each topic are not uniform. This has to do with the variance of the number of relevant documents by topic. As shown by Figure 4.9, Method 2 has a big variance of the number of documents, going from topics with only 1 document to topics with more than 2000 relevant documents. Since there are topics with very few documents, the probability of they being assigned with MeSH terms that are relevant to other topics or on the contrary, to none of them, is much higher, making the results less accurate.

On the other hand, Method 3 shows more uniformity between the topics, with every topic having at least 100 documents and, except for a few cases, less than 1000. In the case of Method 1, the number of document by topic is also very regular. However it is too high compared with the other methods, being around 130000 relevant documents by topic, which is almost the total number of documents of the entire corpus. This means that using this method, practically all the documents are consider relevant for each topic, what justifies they being total similar when using strategies 1 and 2. This is attenuated using the ranking position in the formula for getting the MeSH terms (as strategy 4), since, despite the documents are all considered relevant, they are not necessary always ranked in the same way. The data displayed in Figure 4.9 can also be consulted in appendix C.

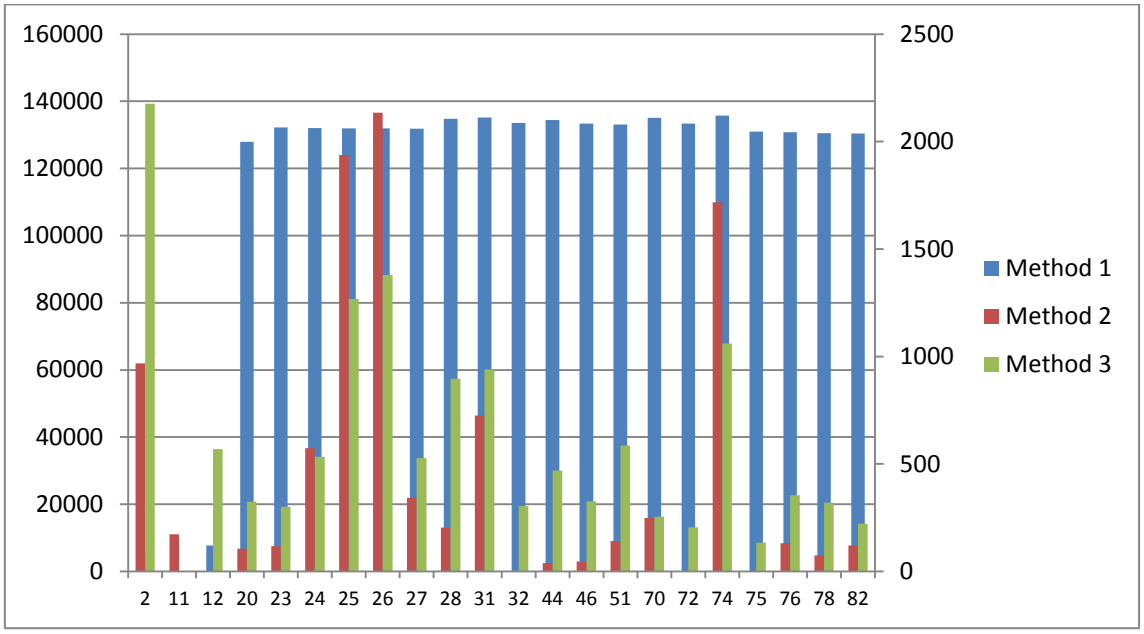


Figure 4.9 Number of documents considered relevant by topic for each method. Values of Method 1 are relative to the scale in the left and values of Method 2 and 3 are relative to the scale in the right. Each topic has only information for the methods they are relevant to.

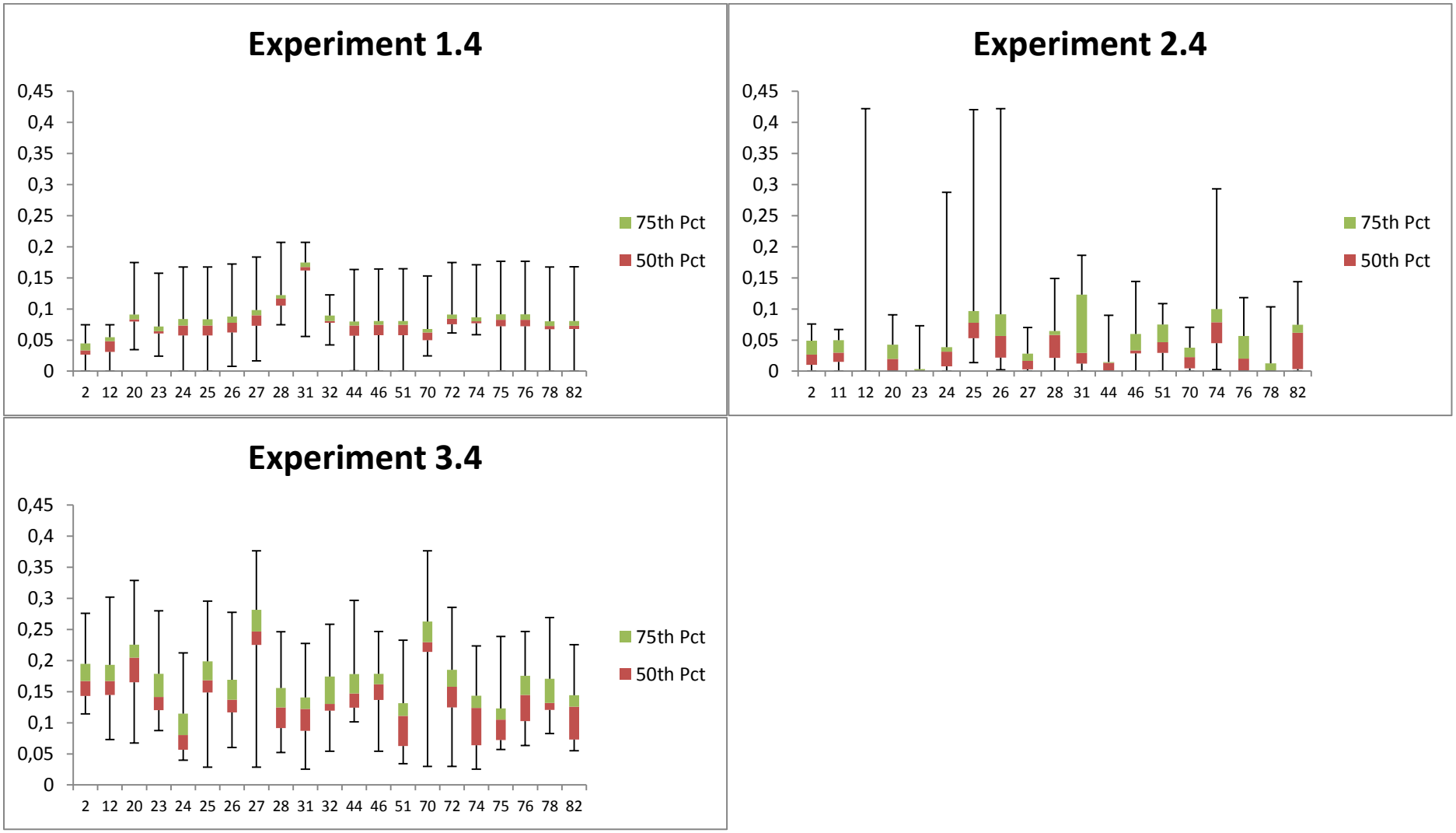


Figure 4.10 Comparison of the different methods using the best strategy for selecting relevant MeSH terms.

4.4.3. Comparison using heat maps

In the previous sections we used boxplot diagrams to illustrate our results concerning the distances between topics and to compare our different methods and strategies. Nevertheless, those results can also been shown in a more clear and understandable way. Thus, as shown in Figure 4.11, we used heat maps to represent the distances between each pair of topics giving a perspective of their dissimilarity.

These new graphs show, that with method 3 we are able to get more distinct topics, emphasizing the conclusions previously reached: methods 1 and 2 present lower values in general, while method 3 presents higher values, meaning the topics are more dissimilar from each other, being even possible to discriminate the most unique ones.

Despite there is no changes in our conclusions, we can visualize with more detail the dissimilarity concerning each pair of topics independently. For instance, regarding method 2, we have seen previously that topic 12 had only one relevant document, being totally similar with some topics, and almost totally dissimilar with others. Here we can see that topic 12 is almost totally dissimilar with topics 25 and 26, which means that very few of the MeSH terms assigned to topic 12's document is relevant to topics 25 or 26. On the other hand they are relevant to almost other topics.

Regarding Method 3, we can see that topic 27, despite being very distinct to other topics, has one that is very similar to it, topic 25. The same happens to topic 70, which is very similar to topic 72.

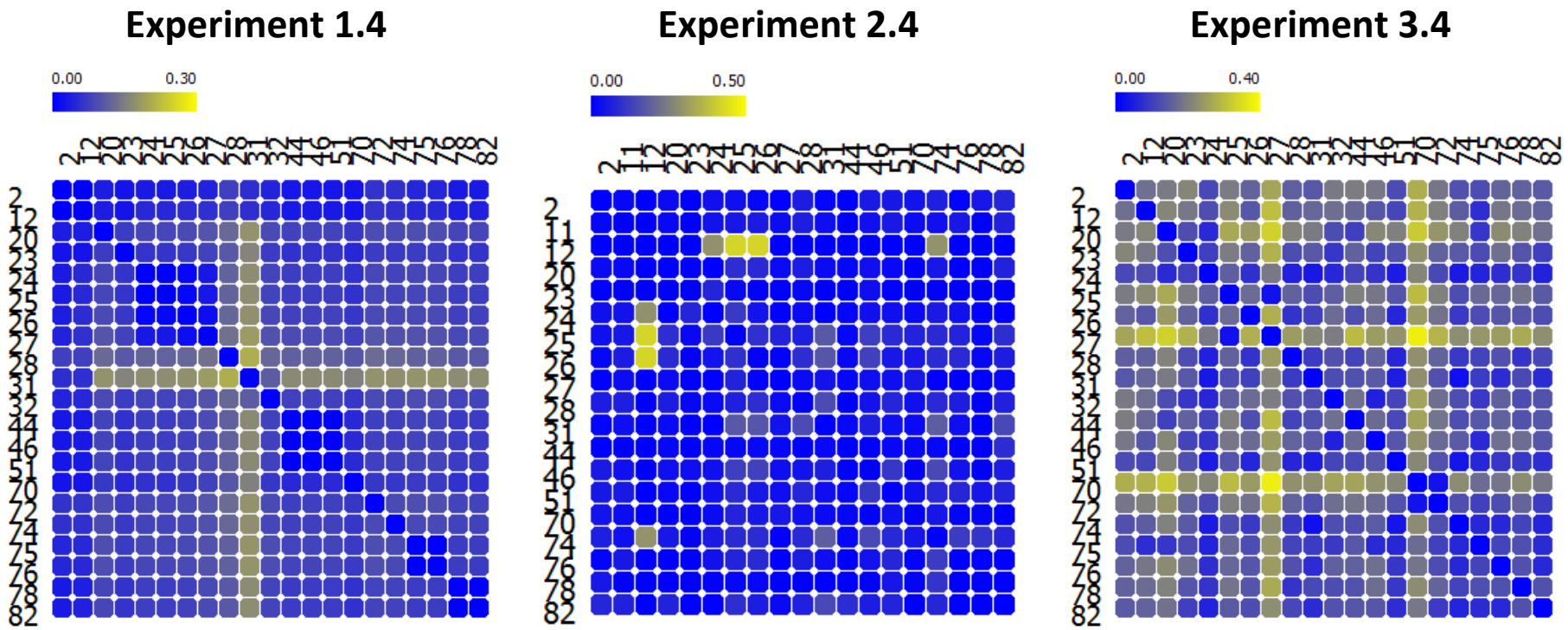


Figure 4.11 Distances between each pair of topics for the different methods using heat maps. Cells with lower values mean more similar topics and cells with higher values mean more dissimilar topics. Diagonal pairs correspond to the same topic having the value of 0. The maps were created using the Orange framework[10].

[10] More information about Orange at: http://orange.biolab.si

4.4.4. Davies-Bouldin Index

Another way to assess our methods, and mainly their capabilities of clustering the documents, is to use a known cluster evaluation metric. We used the Davies-Bouldin Index, a metric for evaluating clustering algorithms [25]. Here, we assume that each topic is a cluster of documents and try to evaluate how distinct those clusters are.

This metric can be divided in two steps: the first one is to measure the distance between the centroid and each individual vector (in our case each document) for each cluster, and then to measure the separation between each pair of clusters. The second step relies on measuring how good the clustering scheme is, considering for each pair of clusters the measures mentioned before. The calculation of Davies-Bouldin Index is shown with more detail in Appendix D.

Hence, firstly we find the centroid of each topic, our clusters. For that we used once again the MeSH terms, with each document being a vector formed by the MeSH terms occurrences, which value is 1 for the ones assigned to that document and 0 for the others. The centroid will be a vector formed by the mean of the document's elements, as follows:

$$A_i = \frac{1}{N}\sum_{k=0}^{N} D_{k,i}$$
$$A = (A_1, A_2, A_3, \dots, A_m) \tag{4.9}$$

With A_i being the value of centroid A for the MeSH term i, $D_{k,i}$ the value of document k for the MeSH term i and N the number of documents.

We calculated the Davies-Bouldin Index, based on its definition for our three different methods and the list of relevant documents for each topic retrieved by each one. To test the importance of the ranking order of the documents, we considered only the top *N* documents of each topic, with *N* being 25, 50 and 100.

By its definition, the lower the values of the Davies-Bouldin Index, the better, meaning the clusters are more separated. The results from Table 4.5 indicates that Method 2 is the best method for clustering, presenting the lower values. This means that using Method 2 the relevant documents will be more distinct from topic to topic, making them less similar. However the differences between the three methods are not very considerable especially between Methods 1 and 3. It is important to remember that this metric doesn't consider our strategies, only the list of relevant documents for each topic. Hence, as we have seen before, the relevant documents don't differ much from topic to topic, especially in Method 1, being essential to consider the documents ranking position. This justifies the poor results for method 1 and 3.

Table 4.5 Davies-Bouldin Index for the three different methods considering different number of documents.

	Top 25 documents	Top 50 documents	Top 100 documents
Method 1	2.323	2.405	2.428
Method 2	1.633	1.721	1.799
Method 3	2.051	2.242	2.403

As we also have seen, the number of documents by topic differs from method to method, being very irregular in the case of Method 2. This may justify the better results presented by Method 2, since some topics have very few relevant documents increasing the probability of being more distant.

We can also see that, whatever the method, the results improve considering fewer documents. This shows that the top documents are more distinct from topic to topic, reinforcing the importance of considering the documents ranking.

4.4.5. Other features

Through the tests presented so far, it is clear that Method 3 is the solution that better achieves our purpose of distinguish the several topics, considering the documents ranking. For that reason, the tests and results presented from now on only consider Method 3 and Strategy 4 (as in experiment 3.4).

However, there are some features that haven't been tested because, since we were comparing methods and in order to have some conformity, those features should be the same for all of them. One of those features is the number of topics used to process LSI. In the previous tests we used 100 topics, especially because some of the methods, namely Method 1, need a great computational effort to process, running out of memory when using a large number of topics. Even with Method 3 we could only reach 200 topics, having memory problems with 300. Nevertheless, the differences are not too considerable as we can see in Figure 4.12.

Another important aspect is the query to use. So far our results are concerning the same query *"Dopamine"*. Though, it is important to know if our solution behaves in the same way for other queries. Thus, we tested our method using different queries, besides the one already used, as shown in Figure 4.13.

As we can see, the results differ much from query to query, with some of them presenting more dissimilar topics than others. It is important to note that the queries don't have the same similarity scores for the several documents, with some being more relevant for the corpus than others. Furthermore, we remember that the documents are formed by annotated terms, instead of their original text, which requires the queries to be more in line with those annotated terms in order to be more relevant. For instance in Figure 4.13, the query "*Alzheimer heart*" is formed by two different annotated terms, while the query "*Parkinson*

disease", represents one annotated term. Thus, the second one is more relevant for the corpus, having better similarity results among the documents.

Another difference relies on the number of relevant topics that are considered in each case. This aspect is related to the one mentioned before, with each query being differently relevant for the corpus, and is related to one last feature: the threshold used on the query's score for each topic.

In Method 3 we used a threshold to consider only the most relevant topics for the query. The threshold can't be too low because, in that case, to many topics will be consider relevant, and it can't be too high, because it may not consider any topics as relevant at all. However, these values vary from query to query, being necessary to adjust that threshold depending on the query to use, until we find an acceptable number of relevant topics. For instance, for the query "*Dopamine*", a threshold of 0.1 was enough. But for the query "Alzheimer heart", we had to use a much lower threshold (0.02) in order to have a list of relevant topics.

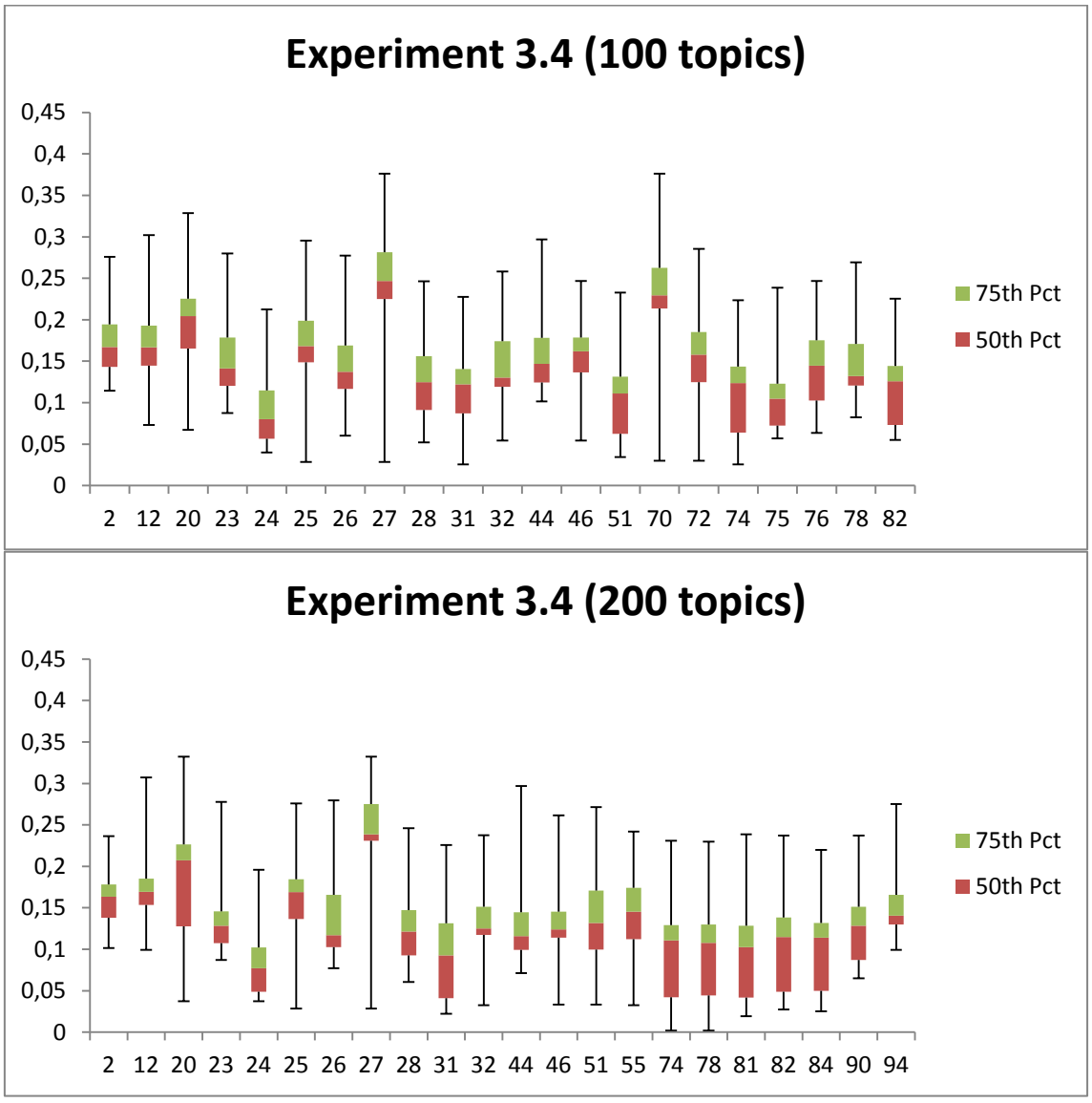


Figure 4.12 Experiment 3.4 using 100 topics in the LSI process and using 200 topics.

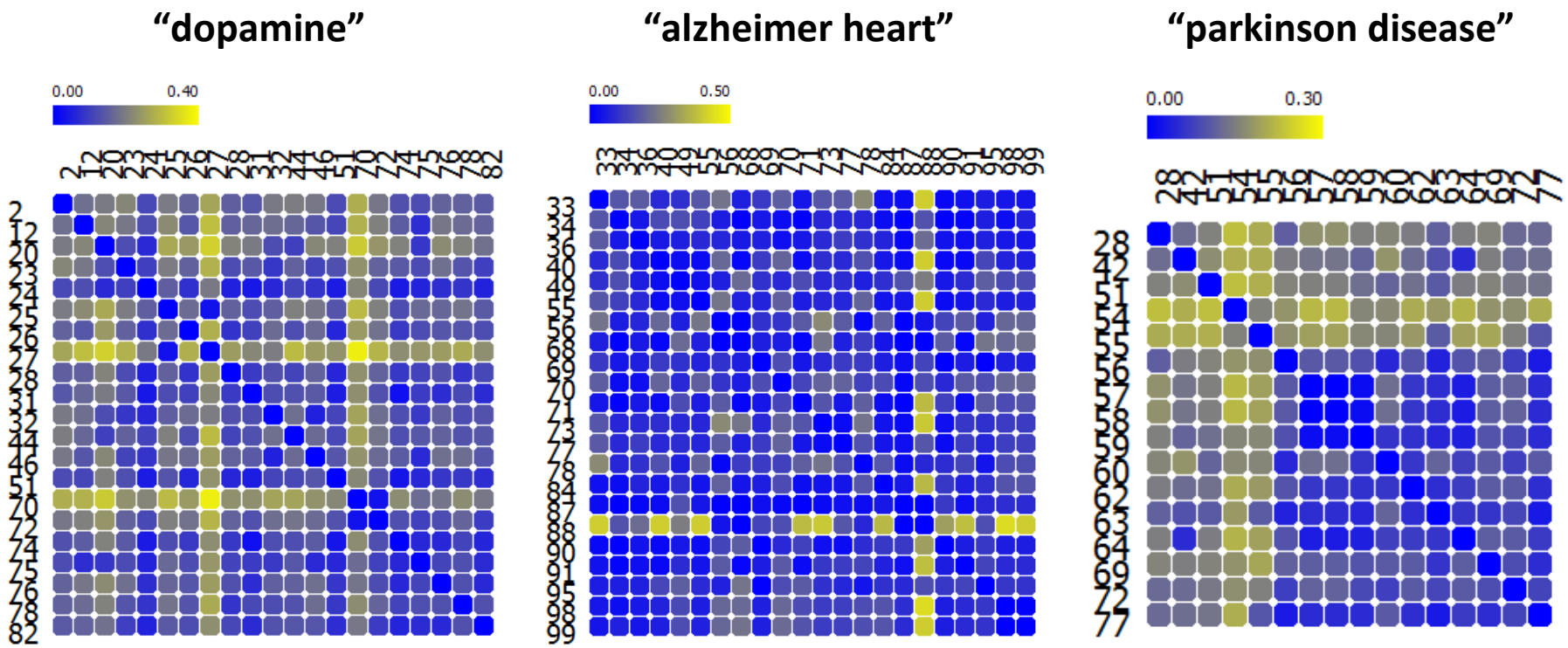


Figure 4.13 Comparison of our solution using different queries.

4.4.6. Conclusions

In this section we compare and test different methods and strategies able to discriminate the most relevant LSI topics regarding a query and to distinguish those topics by their dissimilarity as best as possible. For that we conducted several experiments combining the different methods and strategies, and presented the results in two distinct graphical forms: boxplots and heat maps.

We concluded that our third method, having in account not only the relevance of each document for each topic but also the relevance of each document for the query, is the best method for retrieving the most relevant documents by topic, and our fourth strategy, considering the similarity score and ranking position of the documents, is the best strategy for collecting the most relevant MeSH terms used in the rest of the study.

We also tested our solution for several queries and different numbers of LSI dimensions, but there are no relevant differences from one case to another.

4.5. Visualization

Usually the LSI or topic model results are presented as tables, being hard to understand and analyze them. Our main goal with this experiment is to show the different LSI topics in a visual and perceptible way, so it can be easier to understand several aspects, as the retrieved documents, the distinction between the created topics or even the LSI process itself.

Besides, despite our solution being able to find the most relevant and distinct LSI topics for a query, it is unlikely to have totally dissimilar topics from each other, with some being closer than others. For instance in experiment 3.4 shown in Figure 4.11, we can see that topic 70 is very dissimilar from almost the others but is very similar to topic 72.

Therefore, even with the topics being the most relevant ones for a specific query already, we can order them by their similarity, joining the closer ones and see how much related they are. Thus, even if each topic refers to a different subject, we can see which subjects are more related and if they are too close, cluster and consider them as one.

Regarding these aspects, we present several ways of visualizing the results of our solution, highlighting the topics dissimilarity and if so, the different clusters of topics that can be created.

4.5.1. Heat Maps

In the previous section we already presented two different solutions, although the boxplots diagrams are only used to show statistical results of each topic regarding all the others. On the other hand, with the heat maps we are able of distinguish each pair of topics and have a more perceptible idea of which ones are more or less unique, being a good visual way of showing the different LSI topics.

Furthermore, recurring to heat maps we can also cluster the relevant topics by their similarity, as shown in Figure 4.14. Using hierarchical clustering, it is also possible to see clearly their relations at different levels.

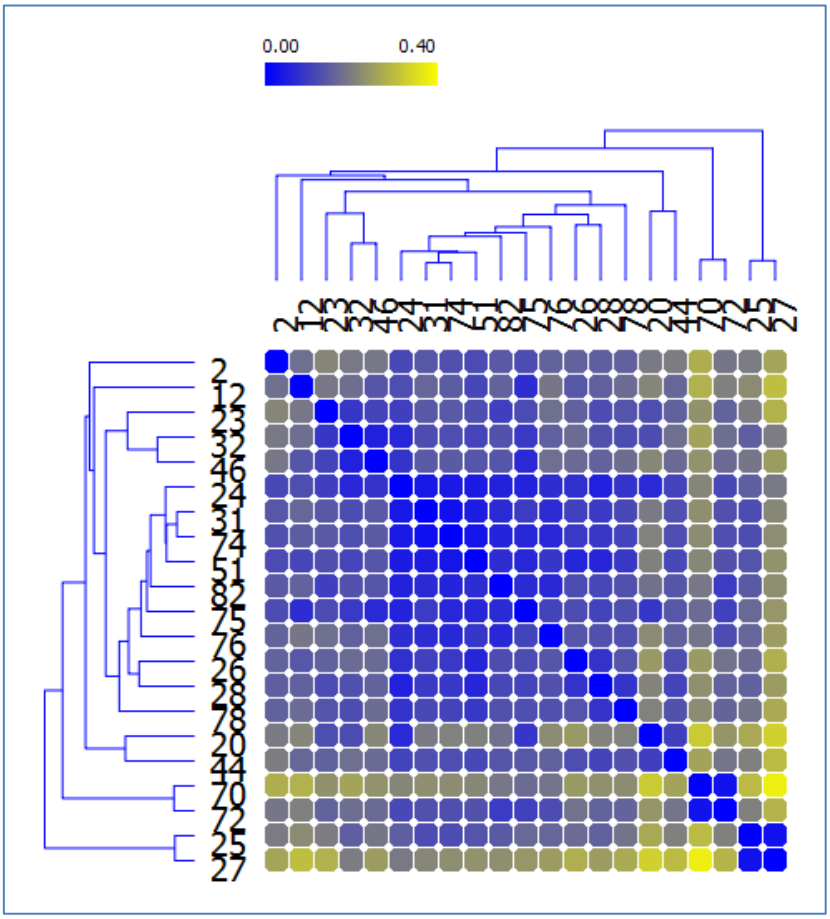


Figure 4.14 Experiment 3.4 is presented as a heat map, with topics clustered by similarity. The different levels of clustering are also shown.

4.5.2. Multidimensional Scaling

Following the topics clustering, we used another form of visualizing the topics dissimilarity recurring to Multidimensional Scaling (MDS).

MDS is an exploratory technique used to visualize proximities in a low dimensional space [26]. It uses a set of data analysis methods to detect underlying relations between entities from a correlation matrix and represents them in a geometrical space, being possible to visualize those relations. If the values of a correlation matrix can be translated into dissimilarity or distance measures, they can be used by MDS and so represented in a dimensional space.

The MDS algorithm starts by determining the specified topics and initial coordinate matrix, followed by calculating the distances between the several entities in the matrix. The next step relies on optimizing the matrix scale. There are several ways of obtaining an initial configuration for getting optimal scaling. Curiously, it can be done using SVD, a technique that, as we have seen, is used in the LSI process.

Thus, it becomes necessary to evaluate how well a particular configuration is able to produce an optimal scaling, or in other words, to measure the goodness-of-fit. This measure is called *Stress*, with probably the most known algorithm being proposed by Kruskal [27]. After this evaluation the coordinates are updated and the scale is optimized once again. This process is repeated until the stress is completely minimized by reaching a threshold.

Figure 4.15 shows a representation of the topics dissimilarity using MDS. It shows clearly the most closer or distant topics, being possible to identify different clusters and so the different subjects represented by the topics. For instance and as in the previous tests, we can see that topics 27 and 70 are more unique, both having a close relation to another: topic 27 is very close to 25 and topic 70 to 72.

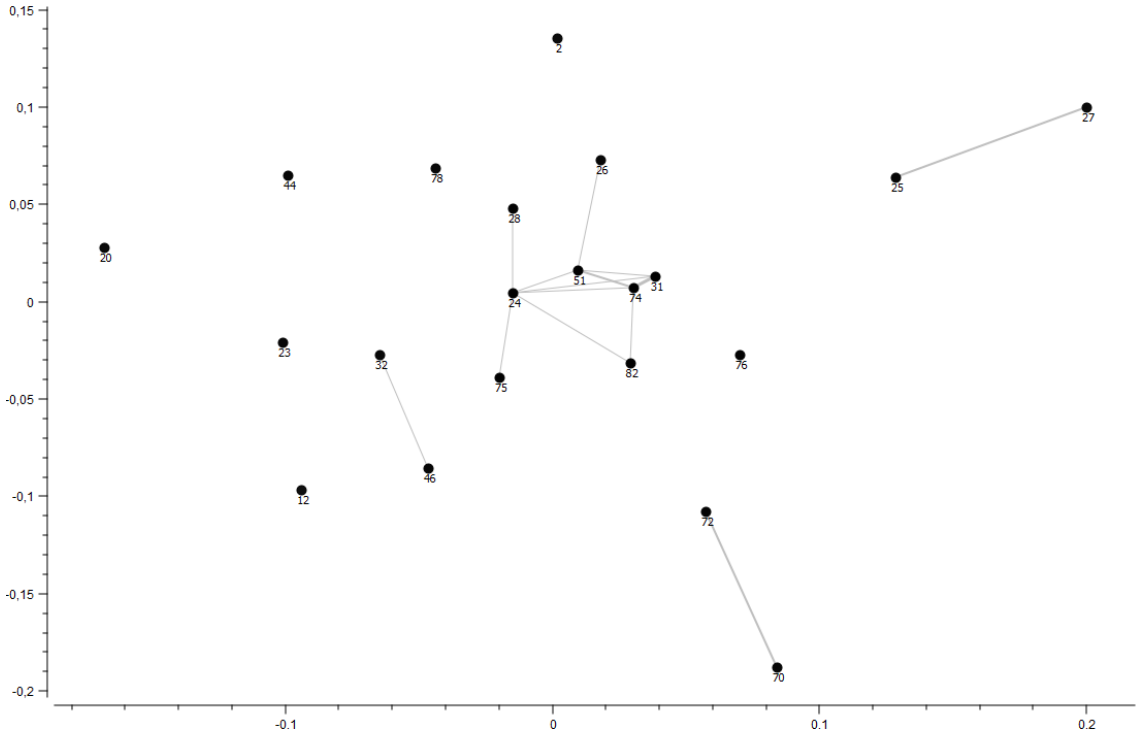


Figure 4.15 Experiment 3.4 represented by a scatterplot using MDS. Lines show the most similar pairs of topics. The MDS computation and the plot creation were performed with the Orange framework.

4.5.3. MDS using clusters of documents

Still using MDS, we tried another way of presenting our results by showing each topic as a cluster of documents. Our idea is to show that we can distinguish and identify the different topics by the dissimilarity between their relevant documents instead of using the topics themselves.

To achieve this, we collected the 20 most relevant documents of each topic and calculated the distance between them with MDS, using their LSI values. We then discriminate the ones which are relevant for only one specific topic with one distinct color, using different colors for the several topics. Documents which are relevant for more than one topic were marked with black.

As we can see in Figure 4.16, some distinct clusters of documents can be identified, marked with the same color, therefore corresponding to the same topic. Some topics, on the other hand are not so evident, with few exclusive documents. The black dots represent documents which are relevant for more than one topic. As we can see they are mostly located in the middle since their distance tends to be equal for the several topics. Nevertheless, some of those documents marked with black are only relevant to two or three different topics and not

necessarily to all of them or to the same ones. It would be interesting to discriminate those documents as well, but the graph would become too messy, being harder to interpret.

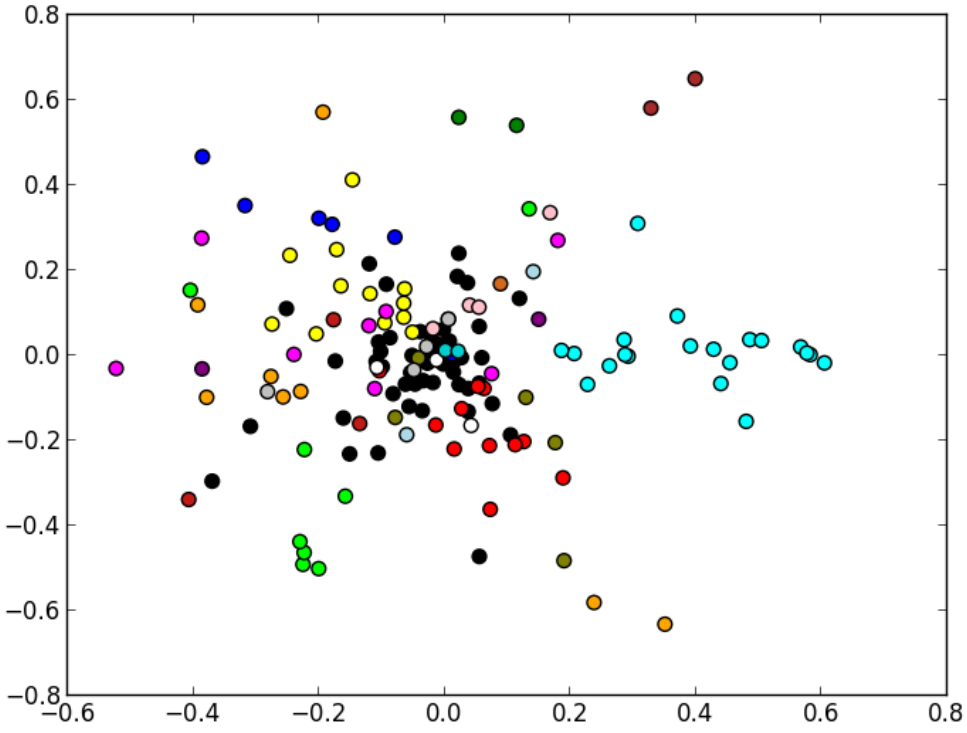


Figure 4.16 Representation of different clusters of documents using MDS. The different colors represent the several topics, with the black dots being documents relevant to more than one topic. The MDS computation was performed with the Orange Framework.

4.5.4. MDS using relevant terms

In the two previous solutions the topics are identified by a label (Figure 4.15) or by the documents exclusively associated to them (Figure 4.16). However, we still don't know which subject each topic refers to. We know that are dissimilar topics that have different sets of documents associated to them, but we don't know what each topic is about.

Ideally, each topic should be labeled by a set of terms that can identify it being easier to know which subject the topic is about. Regarding this, besides collecting the 20 most relevant documents for each topic for calculating MDS, we also collected the 5 most relevant terms, according to the LSI process. This way, besides having the distances between the most relevant documents for all the topics, we also have the distances between the most relevant terms.

Figure 4.17 shows the result of this process. The documents that are exclusively relevant for one specific topic are labeled with that topic number and documents that are relevant for more than one topic are labeled with "*D-1*". The plot is a bit messy due to the labels overlapping, being hard to identify the documents and terms that are closer to each other. Nevertheless, we can see some terms that appears to be closer to some clusters of documents. For instance the cluster formed by documents exclusively associated to the topic 20, has a few terms closely around such as "cholesterol", "nursing" or "dbs". However, we can see that a great part of the terms are distant from most of the documents, indicating that those terms are relevant for all of those documents in a similar proportion and so have the same distance.

As we mentioned, the plot is hard to interpret due to the proximity of the labels. Yet, zooming in the plot allows having a closer look at some parts of it, being clearer the position and distance of every point (Figure 4.18). For instance we can see that the term "dopamine", which matches the query we used, is in the middle of the plot surrounded mostly with documents that are relevant to more than one topic. This means that the term "dopamine" is probably one of the most relevant words for all the documents and has a stronger relation to all other terms, being equally distant to all the other point of the plot.

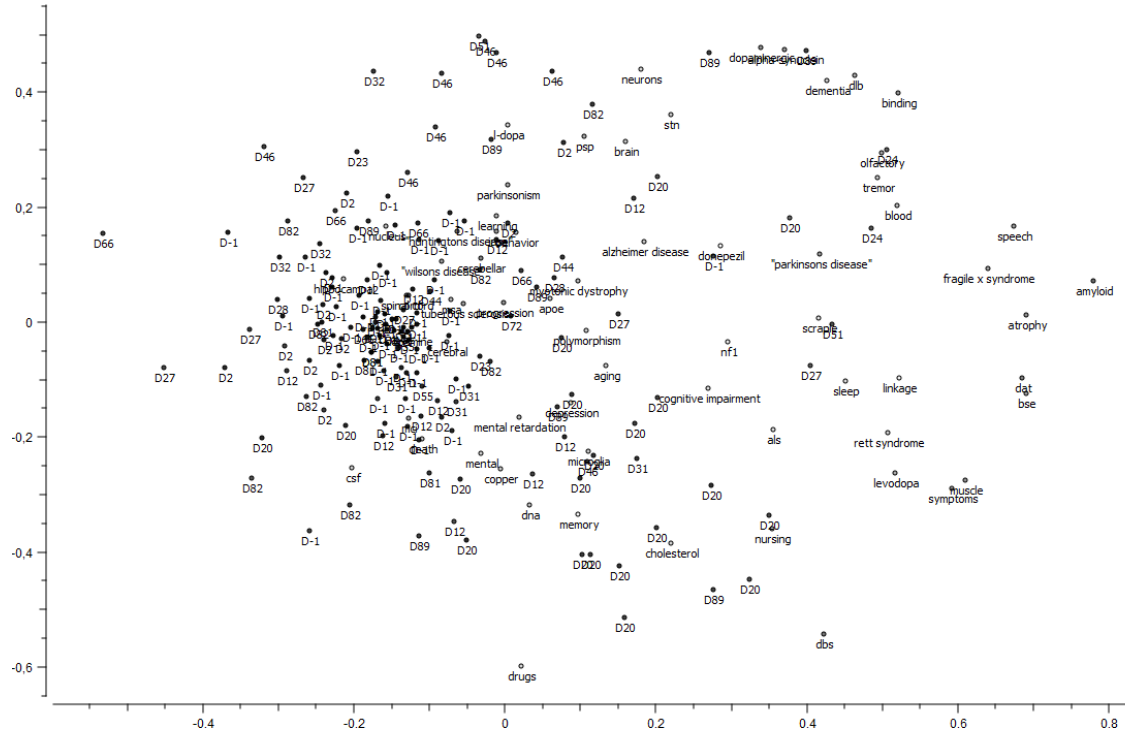


Figure 4.17 Representation of relevant documents and terms using MDS. Documents that are exclusively relevant to specific topics are labeled with that topic number. The others are labeled with "-1".

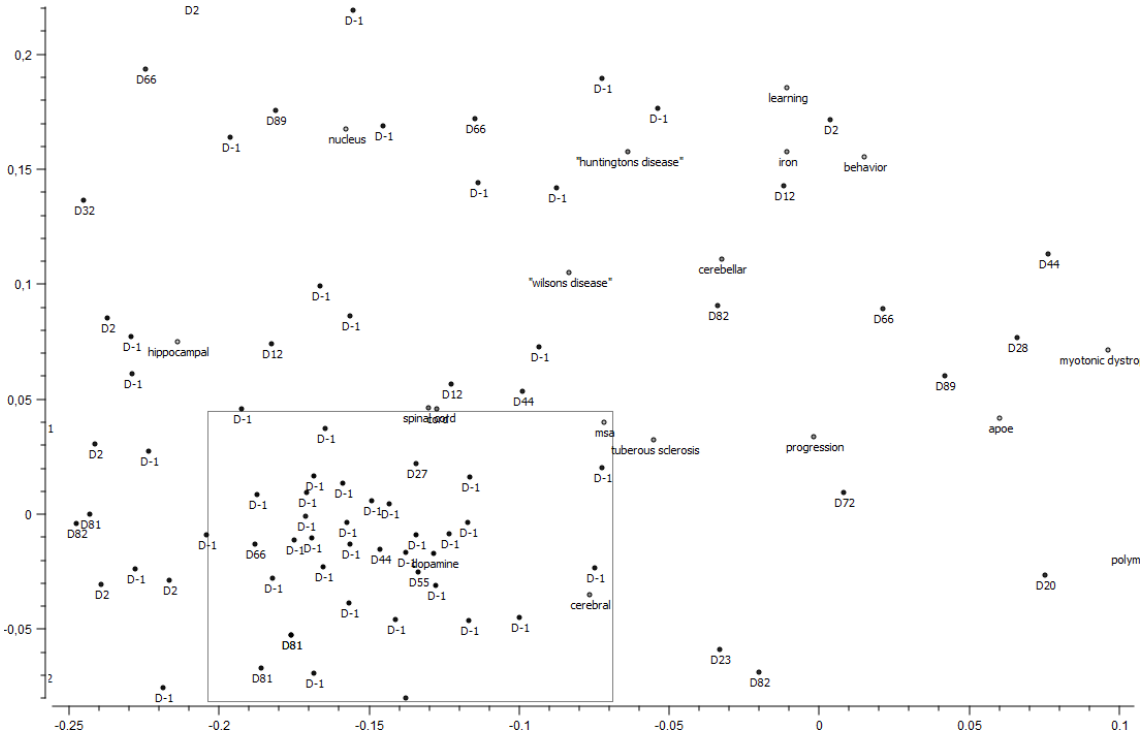


Figure 4.18 Zoomed area of the previous figure highlighting the term “dopamine” which match our query.

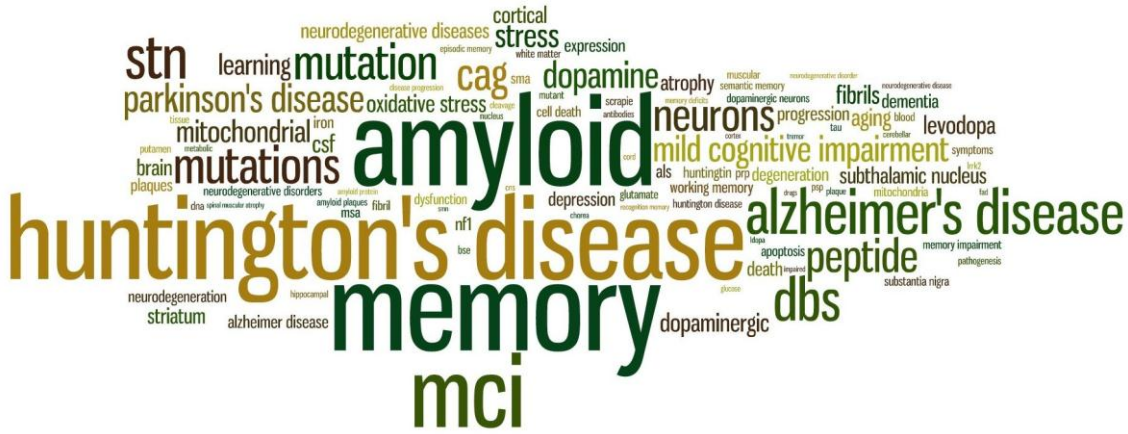
4.5.5. Word Clouds

The previous technique shows the most relevant terms for a topic by proximity and similarity to that topic’s most relevant documents. In some cases however, the results may be inconclusive, since some of the documents or even clusters are too isolated being hard to identify the terms that are more related to them.

In order to have another way of visually showing the terms which are more related to a topic and can be used to describe and identify it, we used word clouds. Word clouds can be used as a visual representation for text data, with each word differing in size, color, position or font, depending on a weight, usually defined by its frequency in a document.

In our case, we collected the top 100 most relevant terms for a specific topic according to LSI and used the LSI value for that topic as score or weight of each term. Figure 4.19 shows an example of word clouds for two specific topics. The more relevant the terms are for the topic, the bigger they are. This way it is possible to know which set of terms can be used to describe a topic and among them, which ones describe it better.

Topic 12



Topic 25

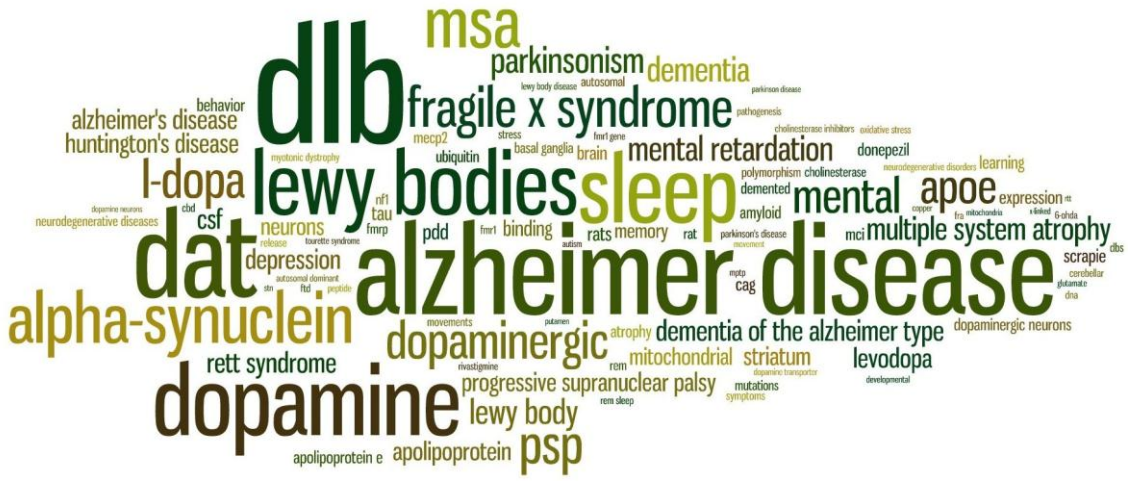


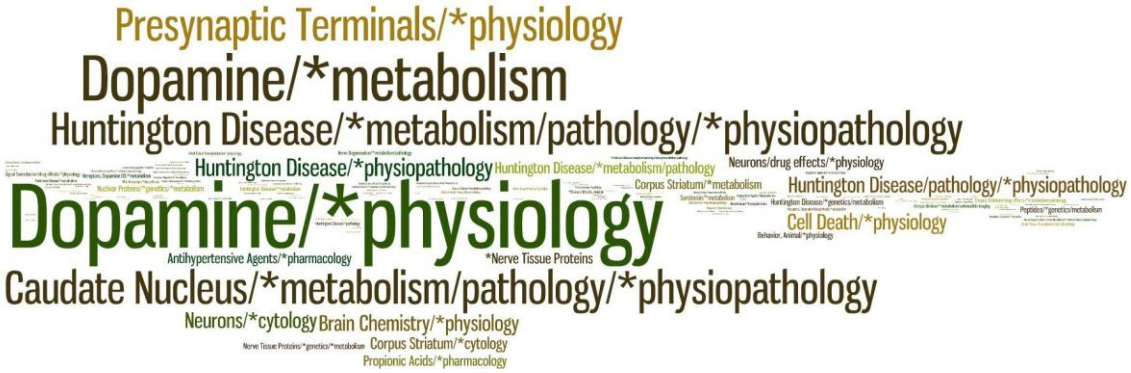
Figure 4.19 Example of two topics representation using Word Clouds. The bigger sized terms are the most relevant ones for that topic. These word clouds were made recurring to the Wordle website[11].

It is important to note that these terms and their scores are selected after the LSI process and before the cosine similarity calculation, which means that the selected terms are independent from the searched terms. Still, we can see that the term “dopamine” (which match with the searched query for this example) appears as one of the most relevant ones, especially in topic 25, which may be an indicator of the reliability of our method for selecting the most relevant topics for our query.

Besides creating the word clouds using the most relevant terms from the LSI process, we can also use the MeSH terms previously used for assessing our methods. As we can see in Figure 4.20, the logic is the same with the most relevant terms having a bigger size. Some users may find more useful to interpret the topics through MeSH terms if, for instance they are familiar with MEDLINE and used to work with it. Thus, we present the two alternatives, being possible to use the one that suits better for the topic interpretation.

[11] More information about Wordle at www.wordle.net

Topic 12



Topic 25

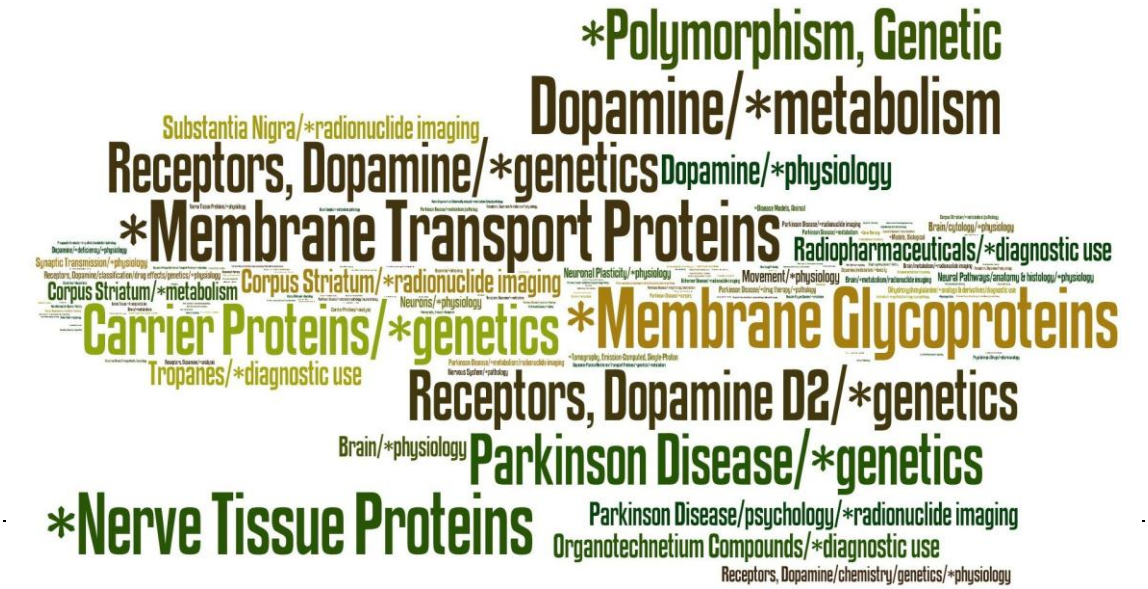


Figure 4.20 Example of two topics representation using Word Clouds, using MeSH terms. The bigger sized terms are the most relevant ones for that topic. The clouds were made, once more, recurring to the Wordle website.

4.6. Discussion

In this chapter, we show another way of how LSI can be used in the biomedical domain. As a first stage, our purpose is to extract a set of distinct topics from the LSI and Cosine Similarity results, in order to have for the same search, different results for different themes. To accomplish this, we developed some methods to find the LSI results for each of its dimensions, or topics and select the most relevant ones based on the documents retrieved for a specific query.

We concluded that this separation and selection of topics must consider not only the relevance of each document for each topic and the relevance of each topic for the query, but also the relevance of each document for the query given by the Cosine Similarity calculation. We also concluded that it is important to consider the documents ranking in the document set for each topic.

In a second stage, we propose several solutions to display the results from the first stage in a visual and understandable way. Using heat maps or MDS it is possible to show and distinguish the selected topics by the dissimilarity between them. Still with MDS or using Word Clouds, it is possible to show and identify each topic with the terms that better describe it.

It is important to note that we didn't develop an information retrieval system that visually shows the LSI results. We are simply presenting several different ways to do that, depending on what needs to be displayed.

Another important aspect is the fact that our evaluation of the topics selection is based on the dissimilarity presented between them. We expect the selected dimensions to be distinct from each other and the documents assigned to each one to be different and focused on different themes. However, we don't know for sure if the themes associated with each selected dimension and the documents or terms considered relevant to each one are correct and make any sense. Yet, there are some methods that can be used to measure the semantic meaning of topics of a topic model. These methods are based on human evaluation tasks to "explicitly evaluate both the quality of the topics inferred by the model and how well the model assigns topics to documents" [28]. Although those methods are usually used for topic models, they could be applied to our solution in order to, through human judgments, examine our selected topics and evaluate their coherence.

5. Conclusions

Our main goal with this work was to study and explore the Latent Semantic Indexing technique in order to find useful applications for it, especially in the biomedical domain.

After studying the advantages and limitations of LSI, we searched and compared several implementations to know which one would suits better our purpose and could be used in the following work. Then we tested the LSI capabilities for two main tasks: documents annotation and structuring the results of information retrieval system.

For the first task we conducted an experiment to use LSI for automatically predict and assign MeSH terms to biomedical articles. This experiment was useful to test LSI performance in tasks other than usual and to test its reliability as an information retrieval technique. The comparison with other studies reveals LSI as a good alternative for using in MeSH terms prediction, with its results being very similar to other techniques.

For the second task, using LSI as a retrieval technique, we managed to retrieve several different sets of documents as results, each one associated to a different topic or theme. We then presented several methods of visualizing and describing those topics and the documents and terms associated to them. This experiment shows some potentialities of LSI, poorly explored until now. Using this technique is possible to improve the results of an information retrieval system by structuring the retrieved documents in a more useful and understandable way. In addition, the several visualization solutions help exploring this new structure, the way documents are related to each other and to the different topics.

5.1. Future Work

The improvements that can be done are mostly related to the second experiment. As we mentioned in the last chapter (Section 4.6), we didn't developed a system to visualize the LSI results. We used other approaches, provided by the Orange framework or wordle.net, to show several solutions for visualization. Nevertheless, it would be interesting to have all these visualization solutions integrated in a complete information retrieval system, capable of retrieving the documents according to our approach and showing those results according to one of the visualization solutions depending on the user's choice.

Regarding the MeSH terms recommendation, as we mentioned in Chapter 3, more experiments should be conducted in order to verify and try to improve the presented results.

References

1. Christopher, D.M., R. Prabhakar, and S. Hinrich, *Introduction to Information Retrieval*2008: Cambridge University Press. 496.
2. Deerwester, S., et al., *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science, 1990. **41**(6): p. 391-407.
3. Jinxi, X. and W.B. Croft, *Query expansion using local and global document analysis*, in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval %@ 0-89791-792-8*1996, ACM: Zurich, Switzerland. p. 4-11.
4. Landauer, T.K. and S.T. Dumais, *A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.* Psychological Review, 1997. **104**(2): p. 211-240.
5. Landauer, T.K., P.W. Foltz, and D. Laham, *An introduction to latent semantic analysis.* Discourse Processes, 1998. **25**(2-3): p. 259-284.
6. Garcia, E. *SVD and LSI Tutorial 4: Latent Semantic Indexing (LSI) How-to Calculations*. 2006 October 2012]; Available from: http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-4-lsi-how-to-calculations.html.
7. Garcia, E. *SVD and LSI Tutorial 3: Computing the Full SVD of a Matrix*. 2006 October 2012]; Available from: http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-3-full-svd.html.
8. Klema, V.C. and A.J. Laub, *The Singular Value Decomposition - Its Computation and Some Applications.* Ieee Transactions on Automatic Control, 1980. **25**(2): p. 164-176.
9. Berry, M.W., S.T. Dumais, and G.W. OBrien, *Using linear algebra for intelligent information retrieval.* Siam Review, 1995. **37**(4): p. 573-595.
10. Dumais, S.T., *Improving the Retrieval of Information from External Sources.* Behavior Research Methods Instruments & Computers, 1991. **23**(2): p. 229-236.
11. Roger, B.B., *An empirical study of required dimensionality for large-scale latent semantic indexing applications*, in *Proceedings of the 17th ACM conference on Information and knowledge management %@ 978-1-59593-991-3*2008, ACM: Napa Valley, California, USA. p. 153-162.
12. ŘEHŮŘEK, R. and P. SOJKA, *Software Framework for Topic Modelling with Large Corpora*, in *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*2010, ELRA: Valletta, Malta: University of Malta. p. 46-50.
13. Brand, M., *Fast low-rank modifications of the thin singular value decomposition.* Linear Algebra and Its Applications, 2006. **415**(1): p. 20-30.
14. ŘEHŮŘEK, R., *Subspace Tracking for Latent Semantic Analysis*, in *Proceedings of the 33rd European Conference on Information Retrieval (ECIR)*, P. Clough, et al., Editors. 2010, Springer: Heidelberg. p. 289-300.
15. Blei, D. and J. Lafferty, *Topic models.* Text Mining: Theory and Applications, 2009.
16. Blei, D.M., A.Y. Ng, and M.I. Jordan, *Latent Dirichlet allocation.* Journal of Machine Learning Research, 2003. **3**(4-5): p. 993-1022.
17. *Medical Subject Headings (MeSH) and Keyword Searching*. Available from: dical Subject Headings (MeSH) and Keyword Searching.
18. Huang, M., A. Neveol, and Z. Lu, *Recommending MeSH terms for annotating biomedical articles.* J Am Med Inform Assoc, 2011. **18**(5): p. 660-7.
19. Zhu, S., J. Zeng, and H. Mamitsuka, *Enhancing MEDLINE document clustering by incorporating MeSH semantic similarity.* Bioinformatics, 2009. **25**(15): p. 1944-51.

20. Djebbari, A., et al., *MeSHer: identifying biological concepts in microarray assays based on PubMed references and MeSH terms.* Bioinformatics, 2005. **21**(15): p. 3324-6.
21. Aronson, A.R., et al., *The NLM Indexing Initiative.* Proc AMIA Symp, 2000: p. 17-21.
22. Aronson, A.R., et al., *The NLM Indexing Initiative's Medical Text Indexer.* Stud Health Technol Inform, 2004. **107**(Pt 1): p. 268-72.
23. Vasuki, V. and T. Cohen, *Reflective random indexing for semi-automatic indexing of the biomedical literature.* J Biomed Inform, 2010. **43**(5): p. 694-700.
24. Kim, W., A.R. Aronson, and W.J. Wilbur, *Automatic MeSH term assignment and quality assessment.* Proc AMIA Symp, 2001: p. 319-23.
25. Davies, D.L. and D.W. Bouldin, *A cluster separation measure.* IEEE Trans Pattern Anal Mach Intell, 1979. **1**(2): p. 224-7.
26. Van Deun, K., W.J. Heiser, and L. Delbeke, *Multidimensional unfolding by nonmetric multidimensional scaling of Spearman distances in the extended permutation polytope.* Multivariate Behavioral Research, 2007. **42**(1): p. 103-132.
27. Kruskal, J.B., *Multidimensional-Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis.* Psychometrika, 1964. **29**(1): p. 1-27.
28. Boyd-Graber, J., et al., *Reading Tea Leaves: How Humans Interpret Topic Models*, in *Neural Information Processing Systems (NIPS)*2009. p. 288--296.

Appendix A

Table A.1 Obtained results for Precision with different approaches of our solution for different number of topics: values refer to mean and standard deviation.

Precision						
Topics	LSI				LDA	
	Sim. Threshold = 0.4		Sim. Threshold = 0.2			
	Terms (%)	Tokens (%)	Terms (%)	Tokens (%)	Terms (%)	Tokens (%)
100	30.35±0.09	32.12±0.06	28.67±0.06	25.45±0.11	13.45±0.19	25.92±0.32
200	34.98±0.08	36.15±0.07	33.23±0.07	31.93±0.10	13.53±0.15	28.81±0.38
300	35.82±0.07	36.64±0.03	34.31±0.04	33.65±0.08	12.47±0.30	30.01±0.40
400	36.21±0.06	36.99±0.03	34.87±0.05	34.48±0.08	12.20±0.22	31.30±0.20
500	36.48±0.05	37.03±0.03	35.25±0.06	34.90±0.09	12.09±0.27	31.97±0.25
600	36.47±0.05	37.10±0.03	35.40±0.05	35.14±0.07	-	-
700	36.33±0.06	37.18±0.04	35.57±0.03	33.33±0.06	-	-
800	36.29±0.07	37.16±0.04	35.75±0.07	35.40±0.08	-	-
900	36.25±0.06	37.21±0.04	35.87±0.06	35.59±0.05	-	-
1000	36.16±0.06	37.20±0.04	35.91±0.04	36.87±0.03	-	-

Table A.2 Obtained results for Recall with different approaches of our solution for different number of topics: values refer to mean and standard deviation.

Recall						
Topics	LSI				LDA	
	Sim. Threshold = 0.4		Sim. Threshold = 0.2			
	Terms (%)	Tokens (%)	Terms (%)	Tokens (%)	Terms (%)	Tokens (%)
100	54.89±0.17	58.67±0.11	51.85±0.11	46.49±0.16	25.05±0.35	47.34±0.59
200	63.26±0.15	66.04±0.13	60.10±0.13	58.33±0.14	24.47±0.27	52.63±0.69
300	64.77±0.12	66.93±0.05	62.04±0.08	61.47±0.11	21.97±0.54	54.82±0.73
400	65.49±0.11	67.58±0.06	63.07±0.10	62.99±0.12	21.44±0.36	57.18±0.36
500	65.97±0.09	67.64±0.05	63.75±0.10	63.75±0.12	21.20±0.46	58.41±0.46
600	65.96±0.09	67.77±0.05	64.03±0.05	64.20±0.10	-	-
700	65.69±0.13	67.92±0.07	64.33±0.05	64.54±0.08	-	-
800	65.34±0.10	67.88±0.07	64.65±0.12	64.67±0.11	-	-
900	65.23±0.10	67.98±0.08	64.88±0.10	65.01±0.07	-	-
1000	65.07±0.12	67.97±0.08	64.95±0.08	67.35±0.06	-	-

Table A.3 Obtained results for F-Score with different approaches of our solution for different number of topics: values refer to mean and standard deviation.

F-score						
Topics	**LSI**				**LDA**	
	Sim. Threshold = 0.4		**Sim. Threshold = 0.2**			
	Terms (%)	**Tokens (%)**	**Terms (%)**	**Tokens (%)**	**Terms (%)**	**Tokens (%)**
100	39.09±0.12	41.51±0.07	36.92±0.08	32.89±0.11	17.84±0.25	33.50±0.42
200	45.05±0.10	46.72±0.09	42.80±0.09	41.27±0.10	17.43±0.19	37.24±0.49
300	46.13±0.09	47.36±0.04	44.18±0.06	43.49±0.08	15.91±0.39	38.79±0.52
400	46.64±0.08	47.81±0.04	44.91±0.07	44.57±0.08	15.55±0.27	40.45±0.26
500	46.98±0.06	47.86±0.04	45.40±0.07	45.11±0.09	15.40±0.34	41.33±0.32
600	46.97±0.06	47.95±0.04	45.60±0.07	45.32±0.07	-	-
700	46.79±0.08	48.05±0.05	45.81±0.03	45.66±0.06	-	-
800	46.66±0.08	48.03±0.05	46.04±0.09	45.75±0.08	-	-
900	46.60±0.07	48.09±0.06	46.20±0.07	46.00±0.05	-	-
1000	46.49±0.08	48.09±0.05	46.25±0.05	47.65±0.04	-	-

Table A.4 Obtained results for MAP with different approaches of our solution for different number of topics: values refer to mean and standard deviation.

MAP						
Topics	**LSI**				**LDA**	
	Sim. Threshold = 0.4		**Sim. Threshold = 0.2**			
	Terms (%)	**Tokens (%)**	**Terms (%)**	**Tokens (%)**	**Terms (%)**	**Tokens (%)**
100	40.24±0.13	44.09±0.07	36.51±0.09	38.45±0.07	16.90±0.23	31.18±0.33
200	49.63±0.06	53.39±0.06	44.95±0.07	49.42±0.06	16.41±0.24	35.71±0.46
300	51.61±0.06	54.75±0.03	47.45±0.04	51.51±0.03	14.51±0.45	37.96±0.58
400	52.29±0.08	55.27±0.05	48.63±0.07	52.54±0.05	14.02±0.40	40.04±0.28
500	52.71±0.08	55.44±0.04	49.47±0.06	53.06±0.04	13.91±0.32	41.37±0.26
600	52.90±0.05	55.60±0.04	50.00±0.05	53.50±0.04	-	-
700	52.65±0.08	55.76±0.05	50.44±0.05	53.74±0.05	-	-
800	52.29±0.10	55.77±0.04	50.94±0.06	53.98±0.04	-	-
900	52.14±0.06	55.76±0.05	51.12±0.04	54.18±0.05	-	-
1000	51.93±0.09	55.71±0.05	51.20±0.04	54.36±0.05	-	-

Table A.5 Obtained results for the time taken to perform with different approaches of our solution for different number of topics: values expressed in seconds.

Computing Time								
Topics	**LSI**				**LDA**			
	Terms (s)		**Tokens (s)**		**Terms (s)**		**Tokens (s)**	
	all process	**only LSI**	**all process**	**only LSI**	**all process**	**only LDA**	**all process**	**only LDA**
100	35 sec	10 sec	40 sec	15 sec	60 sec	40 sec	140 sec	90 sec
200	45 sec	15 sec	60 sec	25 sec	100 sec	50 sec	180 sec	120 sec
300	60 sec	20 sec	85 sec	35 sec	130 sec	110 sec	240 sec	150 sec
400	75 sec	25 sec	95 sec	40 sec	180 sec	160 sec	420 sec	270 sec
500	90 sec	32 sec	120 sec	50 sec	220 sec	185 sec	540 sec	300 sec
600	115 sec	38 sec	140 sec	60 sec	-	-	-	-
700	140 sec	52 sec	160 sec	70 sec	-	-	-	-
800	150 sec	54 sec	180 sec	85 sec	-	-	-	-
900	170 sec	63 sec	200 sec	95 sec	-	-	-	-
1000	190 sec	71 sec	220 sec	105 sec	-	-	-	-

Appendix B

Table B.1 Example of matrix with the distances between topics.

Topic	2	12	20	23	24	25	26	27	28	31	32	43	44	46	51	55	71	72	74	77	78	80
2	X	0,177	0,192	0,215	0,114	0,194	0,158	0,251	0,153	0,145	0,197	0,167	0,175	0,186	0,19	0,193	0,192	0,124	0,132	0,168	0,156	0,134
12	0,177	X	0,213	0,192	0,124	0,222	0,139	0,293	0,154	0,162	0,171	0,162	0,105	0,172	0,186	0,216	0,186	0,137	0,151	0,187	0,169	0,147
20	0,192	0,213	X	0,123	0,067	0,266	0,239	0,319	0,207	0,195	0,140	0,117	0,114	0,237	0,236	0,234	0,102	0,179	0,205	0,206	0,200	0,083
23	0,215	0,192	0,123	X	0,065	0,212	0,162	0,281	0,125	0,148	0,102	0,117	0,145	0,124	0,135	0,170	0,106	0,135	0,159	0,180	0,143	0,123
24	0,114	0,124	0,067	0,065	X	0,149	0,077	0,181	0,000	0,042	0,000	0,112	0,102	0,085	0,088	0,073	0,087	0,036	0,045	0,086	0,074	0,049
25	0,194	0,222	0,266	0,212	0,149	X	0,170	0,024	0,155	0,125	0,193	0,214	0,201	0,190	0,182	0,153	0,200	0,131	0,115	0,183	0,162	0,121
26	0,158	0,139	0,239	0,162	0,077	0,170	X	0,266	0,081	0,103	0,163	0,184	0,134	0,106	0,130	0,179	0,145	0,085	0,096	0,118	0,127	0,103
27	0,251	0,293	0,319	0,281	0,181	0,024	0,266	X	0,236	0,208	0,250	0,264	0,277	0,269	0,261	0,183	0,273	0,218	0,203	0,254	0,237	0,208
28	0,153	0,154	0,207	0,125	0,000	0,155	0,081	0,236	X	0,090	0,096	0,152	0,127	0,105	0,062	0,153	0,092	0,086	0,095	0,000	0,103	0,103
31	0,145	0,162	0,195	0,148	0,042	0,125	0,103	0,208	0,090	X	0,123	0,143	0,119	0,080	0,096	0,097	0,119	0,034	0,027	0,098	0,092	0,031
32	0,197	0,171	0,140	0,102	0,000	0,193	0,163	0,250	0,096	0,123	X	0,201	0,177	0,122	0,051	0,145	0,120	0,118	0,129	0,000	0,113	0,133
43	0,167	0,162	0,117	0,117	0,112	0,214	0,184	0,264	0,152	0,143	0,201	X	0,162	0,201	0,190	0,119	0,118	0,100	0,144	0,163	0,129	0,136
44	0,175	0,105	0,114	0,145	0,102	0,201	0,134	0,277	0,127	0,119	0,177	0,164	X	0,133	0,164	0,156	0,095	0,117	0,121	0,143	0,118	0,111
46	0,186	0,172	0,237	0,124	0,085	0,190	0,106	0,269	0,105	0,080	0,122	0,201	0,133	X	0,078	0,184	0,129	0,108	0,128	0,000	0,128	0,126
51	0,190	0,186	0,236	0,135	0,088	0,182	0,130	0,261	0,062	0,096	0,051	0,190	0,164	0,078	X	0,174	0,108	0,108	0,118	0,149	0,118	0,124
55	0,1933	0,216	0,234	0,170	0,073	0,153	0,179	0,183	0,153	0,097	0,145	0,119	0,156	0,184	0,174	X	0,184	0,110	0,08	0,170	0,149	0,099
71	0,1925	0,186	0,102	0,106	0,087	0,200	0,145	0,273	0,092	0,119	0,120	0,118	0,095	0,129	0,108	0,184	X	0,102	0,114	0,147	0,111	0,104
72	0,1248	0,137	0,179	0,135	0,036	0,131	0,085	0,218	0,086	0,034	0,118	0,100	0,117	0,108	0,108	0,110	0,102	X	0,019	0,098	0,087	0,017
74	0,132	0,151	0,205	0,159	0,045	0,115	0,096	0,203	0,095	0,027	0,129	0,144	0,121	0,128	0,118	0,088	0,114	0,019	X	0,094	0,082	0,007
77	0,1689	0,187	0,206	0,180	0,086	0,183	0,118	0,254	0.000	0,098	0,000	0,163	0,143	0,000	0,149	0,170	0,147	0,098	0,094	X	0,120	0,089
78	0,156	0,169	0,200	0,143	0,074	0,162	0,127	0,237	0,103	0,092	0,113	0,129	0,1185	0,128	0,118	0,111	0,111	0,087	0,082	0,120	X	0,088
80	0,134	0,147	0,083	0,123	0,049	0,125	0,103	0,208	0,103	0,031	0,133	0,136	0,111	0,126	0,124	0,104	0,104	0,017	0,007	0,089	0,088	X

Appendix C

Table C.1 Number of documents considered relevant by topic for each method.

Topic	Method 1	Method 2	Method 3
2	5818	968	2176
11	-	173	-
12	7694	1	569
20	127954	106	325
23	132211	117	301
24	131978	574	533
25	131971	1938	1267
26	131929	2135	1379
27	131794	342	527
28	134751	204	896
31	135168	725	940
32	133592	-	303
44	134406	39	469
46	133375	46	326
51	133094	142	585
70	135065	248	255
72	133337	-	205
74	135780	1718	1060
75	131013	-	134
76	130825	131	355
78	130479	74	320
82	130365	120	222

Appendix D

Calculating the Davies-Bouldin Index

The first step relies in calculating the measure of scatter within the cluster, which is nothing more than the Euclidean distance between the centroid of the cluster and each document:

$$S_i = \sqrt{\frac{1}{T_i}\sum_{j=1}^{T_i}\left|X_j - A_i\right|^2} \tag{D.1}$$

Where A_i is the centroid of cluster C_i, X_j the value of document j and T_i the size of cluster C_i, which means the number of documents of that cluster. S_i is the measure of scatter within the cluster C_i.

Then we measure the separation between each pair of clusters, C_i and C_j as follows:

$$M_{i,j} = \left\|A_i - A_j\right\| = \sqrt{\sum_{k=1}^{N}\left|a_{k,i} - a_{k,j}\right|^2} \tag{D.2}$$

To measure the quality of the clustering scheme, we calculate the ratio of S_i and $M_{i,j}$:

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}} \tag{D.3}$$

Since we want the clusters to be as separated as possible, the value of $M_{i,j}$ should be as higher as possible, making $R_{i,j}$ to be lower. The Davies-Bouldin value for the cluster C_i will be the worst case among the ratio between C_i and all the others, which means the most similar cluster to cluster C_i. Davies-Bouldin Index will be the mean of the chosen values for each cluster:

$$\begin{aligned} D_i &\equiv \max R_{i,j} \\ DB &= \frac{1}{N}\sum_{i=1}^{N} D_i \end{aligned} \tag{D.4}$$