FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

On Using Statistical Semantic on Domain Specific Information Retrieval

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Navid Rekabsaz
Matrikelnummer 1129057

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Dr. Allan Hanbury
Mitwirkung: Dr. Mihai Lupu

Wien, TT.MM.JJJJ

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

On Using Statistical Semantic on Domain Specific Information Retrieval

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Navid Rekabsaz
Registration Number 1129057

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Dr. Allan Hanbury
Assistance: Dr. Mihai Lupu

Vienna, TT.MM.JJJJ

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Navid Rekabsaz
Favoritenstrasse 9/11 HE0434, 1040 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Note of Thanks

With respect and love to my great parents.

Acknowledgements

This work is supported by MUCKE (FWF number 213457) project under student grant and later master student research position.

Abstract

Information retrieval must move from a pure surface-based point-of-view to a conceptual point-of-view that matches the contents on a semantic level. Exploring the opportunities offered by statistical semantics, we revisit text-based retrieval on two very different domains (social image as well as patent retrieval) in order to provide a comparative analysis of the efficiency and effectiveness of the analyzed methods. Our semantic-based retrieval approach consists of two elements: first, the methods to create the semantic representations of the terms and second, the approaches to measure the conceptual-based similarity of the texts. For term representations, we use Word2Vec, a state-of-the-art approach based on deep learning, as well as Random Indexing, a more straightforward but effective count-based method. Reviewing the literature, we also select two text similarity methods: one directly measuring similarity at document level (SimAgg) and the other considering the similarity of two documents as a linear combination of the relatedness of their terms (SimGreedy).

We assess the performance and limitations of the mentioned methods, by comparing them to the state-of-the-art text search engines. On both the domains, our semantic retrieval methods show a statistically significant improvement in comparison to a best practice term-frequency-based search engine, at the expense of a significant increase in processing time. To address the time-complexity problem of semantic-based methods, we also focus on optimization to enable larger and more real-world style applications.

Kurzfassung

Information Retrieval (dt. Informationsrückgewinnung) muss sich von einer rein Oberflächebasierten Sicht zu einer begrifflichen Sicht, die den Inhalt auf einer semantischen Ebene übereinstimmt, bewegen. Während wir die Möglichkeiten der statistischen Semantik erforschen, nehmen wir das textbasierte Information Retrieval Methoden in zwei sehr unterschiedliche Bereiche (soziale Medien sowie Patent Retrieval) wieder auf, um eine vergleichende Analyse der Effizienz und Effektivität der untersuchten Methoden zu liefern. Unser semantischbasiertes Vorgehen für Informationsrückgewinnung besteht aus zwei Elementen: auf einer Seite finden wir Methoden die semantische Begfriffdarstellungen erstellen, und auf der anderen Seite die Methoden um die begriffbasierte Ähnlichkeit der Texte zu messen. Für Begriffschilderungen, verwenden wir Word2Vec, eine auf dem neuesten Stand der Technik, auf tiefes Lernen basierte Methode, und Zufalls Indexierung, eine einfache aber effektive Zählbasierte Methode. Nach der Überprüfung der Literatur, wählen wir auch zwei Methoden die Textähnlichkeiten messen: eine Methode die die Textähnlichkeint direkt auf Dokumentebene misst (SimAgg), und eine Methode die Ähnlichkeit zweier Dokumente als Linearkombination der Begriffverwandtschaft ihrer Begriffe misst (SimGreedy).

Wir bewerten die Leistung und die Grenzen der oben genannten Verfahren durch einen Vergleich mit den modernsten Suchmaschinen. Für beide Domänen zeigt unsere semantische Suchverfahren eine statistisch signifikante Verbesserung im Vergleich zu einem bewährten, frequenzbasierten Suchverfahren, auf Kosten einer wesentlichen Erhöhung der Bearbeitungszeit. Um die Zeit-Komplexität der semantisch basierten Methoden anzugehen, konzentrieren wir uns auf Optimierungen die größere und erfahrungspraktischere Anwendungen ermöglichen.

Contents

CHAPTER 1

Introduction

In this work, we shed additional light on the use of the meaning of the words in domain-specific Information Retrieval. We study the state-of-the-art statistical semantic methods and explore their performance and limitations by testing them on two challenging Information Retrieval datasets. In addition, we focus on optimizing the execution time without decreasing the overall performance. In this chapter, we first explain the motivation and specify the domain of the study. Then, we describe the objective of the work, followed by the methodology to achieve it. Finally, the structure of the work is described.

1.1 Motivation

Information Retrieval is a highly experimental, often heuristic research area where 'discovery' is more frequent than 'invention'. In a standard model of information retrieval, searching the information consists of two phases: First, the system analyses the document and the query to create a clear and determined representation of each. This is a quite intuitive phase and in many ways the achieved representation hides the complexity of human language, while it can be addressed using formal statistical methods. In the second phase, the representations are matched and the documents with the best matches are retrieved as potential information resources [62].

Text Retrieval, as a branch of Information Retrieval, traditionally represents the documents as a set of terms (words) and finds the text similarity by matching the identical terms between the query and the documents. In these methods, the measuring the text similarity is essentially based on the number of occurrence of the matching terms between the documents (*term-frequency-based*). Despite their successes, they fail when two texts use a disjunct vocabulary to describe the same fact or situation. Measuring the degree of how well two texts relate semantically (semantic similarity) can be seen as the natural succession of text similarity.

In semantic similarity, each text unit (term, sentence, document and etc.) is represented by a set of *concepts*. For our purpose, we shall denote by *concept*, vectorial representations obtained by several terms, which do not necessarily have a correspondence in natural language.

Considering a defined set of concepts, in this thesis, the term *semantics* refers to a transformation from terms to a vector in a vector space where proximity is indicative of conceptual similarity. This numeric vector indicating the meaning of a term is called *semantic representation* of the term. In contrast to text similarity, semantic similarity goes beyond the surface of the text, tries to find conceptually similar aspects of the documents.

In order to generate the semantic representation of a term, some make use of *knowledge* databases e.g., WordNet [16] or, OpenCyc [1]. These databases encode term meanings from a priori and explicitly human-entered knowledge. WordNet in particular is a manually constructed semantic network for English inspired by human semantic memory. The database groups the terms into sets of synonyms called synsets, each representing a concept. The synsets are interlinked based on semantic and lexical relations. Although these methods are effective in representing the underlying relation of the terms, they cannot be applied to less supported languages or to more specific domains without pre-existing knowledge.

As an alternative to knowledge-based systems, *corpus-based* semantic methods model the co-occurrence of terms to represent their meaning. The idea behind these methods is that the meaning of terms must depend on the use and the context in which they appear [62]. In fact, if two terms occur often in the same context, they are more likely to share a similar meaning. In contrast to knowledge-based, corpus-based methods do not need any pre-existing knowledge and exclusively derive the information from large corpora.

Regardless of the applied approach (knowledge- or corpus-based), both eventually provide a representation for each term which can be used to identify the degree of conceptual similarity between the terms. As the succession of term-to-term similarity, text-to-text similarity tries to measure the semantically relatedness of two texts. The method is generally divided into short text (sentence, phrase, sequence of terms, etc.) and long text (paragraph or document) similarity. Semantic similarity of short texts has been investigated in a number of different tasks, e.g., paraphrase recognition [14], image retrieval by caption [9], Twitter tweets search [55] by applying the techniques from related fields, including natural language processing and artificial intelligence [58]. Despite a generally strong interest on term-to-term or sentence-to-sentence similarity [20], research on any text-to-text and specifically document retrieval level remains limited.

Beside similarity measures, the domain of the data plays an important role in retrieval approaches. We put the focus of this study on two specific domains: *Social Image Retrieval* based on textual features and *Patent Retrieval*. Both domains potentially have an intriguing possibility of being explored by semantic similarity methods. In addition, since each domain requires specific considerations, evaluating the methods on very different domains provides a better understanding on the capabilities of the semantic methods.

The first domain is Social Image Retrieval which uses image as well as textual features to retrieve the most similar documents. Despite the recent success of image deep learning methods [23], text remains an important source of information in many cases where tags, descriptions, or other textual contexts are associated with the image. Nevertheless, and understandably, the focus resided on image processing and, so far, the methods used for text similarity for the purpose of image retrieval are fairly mainstream [56]. Such methods fail when two texts use a disjunct

[1] `http://www.cyc.com/platform/opencyc`

vocabulary to describe the same fact or situation. As semantic similarity discovers the interlinks beyond the surface of the terms, therefore it is interesting to explore the effect of these methods on image metadata document retrieval level.

The second studied domain is Patent Retrieval. Search on patent data has attracted researchers' attention as early as 1977. As a specific characteristic, the language used in patents is precise while very sophisticated such that even an educated, native speaker cannot fully understand the language in which the documents were written. Another characteristic of the domain is the distribution of the term and document frequencies such that they are statistically different from other natural language domains. These characteristics make addressing these tasks much more challenging than the usual use cases, such as web search [28]. Considering these facts, a patent search system aims to ensure the novelty of a patent such that there is no significant overlap between the *essence* of the patent application and an earlier invention or publication [47]. It is therefore sensible to explore the opportunities offered by semantic text similarity methods in the context of patent retrieval.

1.2 Research Objectives

In this research, we focus on text-to-text semantic similarity based on statistical corpus features. We test state-of-the-art term representations in conjunction with semantic text-to-text similarity methods. The objective of the study is to evaluate the performance and limitations of the methods and term representations and provide a comparative study on two very different domains: Social Image Retrieval and Patent Retrieval. We provide experiments incorporating state-of-the-art test collections in both domains. Using corpus-based features sets our approach apart from the ones that use knowledge-based databases or Natural Language Processing (NLP). The corpus-based semantic approach can be applied to other, less supported languages as well as more specific domains without pre-existing knowledge. In addition, we examine possible optimization techniques.

Therefore, in this study we address three questions:

1. Can statistical semantics outperform state-of-the-art text search engines in terms of text-based retrieval effectiveness?

2. Among statistical semantic methods, do newer approaches (i.e., deep learning) outperform older ones (i.e., random indexing)?

3. To which extent do statistical semantics methods perform similarly/differently in different domains?

1.3 Methodology

As shown in Figure 1.1, our approach to explore the goal of the study consists of four main steps: *Semantic Retrieval*, *Optimization*, *Baseline*, and *Evaluation*. In the following, we explain each step in detail.

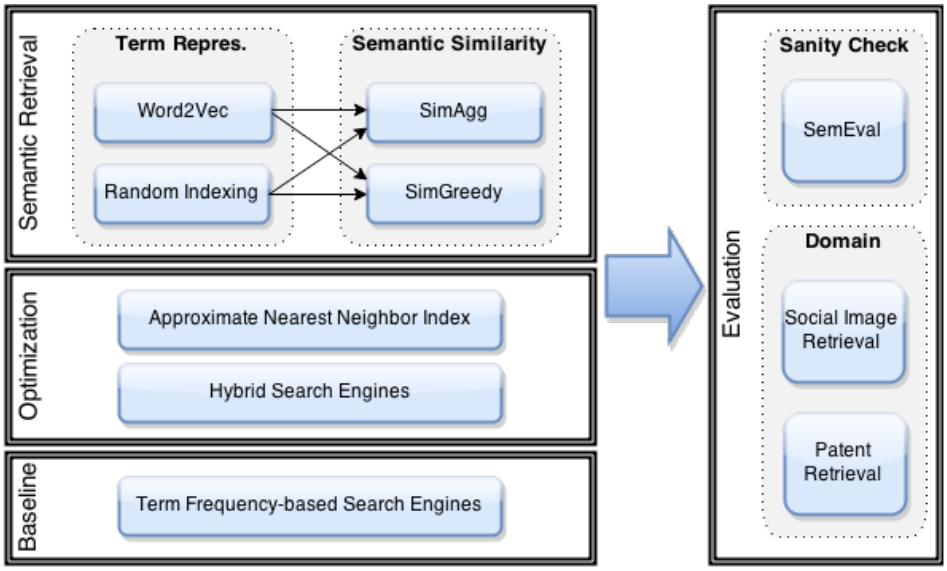

Figure 1.1: Overview of the essential steps of the study

Semantic Retrieval

This step focuses on applying semantic-based methods to retrieve related text documents. It consists of two sets of approaches: *Term Representations* and *Semantic Similarities*. As mentioned, term representation methods create the models, representing the terms in the vector space which are used to achieve a conceptual representation of the documents. The characteristics of term representations are influenced by many factors such as the training data set, the pre-processing phase, the training methods and their parameters tunings as well as the choice of the number of dimensions of the vector to represent the term. We focus on two semantic representation methods, namely *Word2Vec* and *Random Indexing*. Word2Vec is a state-of-the-art approach based on neural network and deep learning algorithms. Random Indexing introduces a more straight-forward but effective method which refines randomly generated vectors by the effect of the occurrence of the terms in their contexts. Given the vector representations of two sets of terms (two documents), semantic similarity methods measure the relatedness of these documents. We adopt two text-to-text semantic similarity measures introduced in the literature. The first is *SimAgg* which directly calculates the similarity at document level and the second is *SimGreedy* which considers the similarity of two documents as a linear combination of the relatedness of their terms.

By having two methods for each set (term representation and semantic similarity), we then examine the one-to-one combinations of the elements of the sets on textual document retrieval. For this purpose, we develop an information retrieval system called *TextSim*[2] which provides a framework for combining the different similarity, representation and preprocessing methods as well as running the experiments.

[2] `https://github.com/neds/textsim`

Optimization

In the optimization part, we focus on reducing the execution time by studying the effect of two methods: *Approximate Nearest Neighbor Index (ANN-Index)* and *Hybrid Search Engines*. ANN-Index uses nearest neighbor search techniques to create an index which later is used to enhance the document retrieval. The Hybrid approach combines two retrieval methods with the aim of exploiting the capabilities of both. The detail of the method is described in Section 4.3. A common characteristic of the mentioned methods is that they both use approximation techniques to select a portion of the data for retrieval instead of all of it. Obviously, this technique may affect the performance of the system by the case that the related documents has been filtered out in the approximation step. The goal is to optimize the execution time in a way that the performance remains in the range of significantly indifferent in comparison to not-optimized methods.

Baseline

In contrast to semantic-based methods, as mentioned bofore, the term-frequency-based methods use the number of occurrence of the terms to explore related documents. Since many state-of-the-art search engines are based on these methods, we define them as the baseline of the experiments. In other words in each experiment, the result of the semantic-based methods are assessed regarding to a best-practice of a term-frequency-based search engine.

Evaluation

As the last step, we evaluate the mentioned methods on Social Image and Patent Retrieval domains. For each domain, we select a state-of-the-art test collection to address specific information access task. The first test data set is the *MediaEval Retrieving Diverse Social Images Task 2013/2014 [21, 22]* which addresses result relevance and diversification in the social image retrieval domain. The second is *CLEF-IP Claims to Passage Task 2013 [47]* which targets the patent retrieval domain by providing a large data set in three languages (English, German, French). Before running the experiments on the mentioned domains, we first check the sanity of the methods using the SemEval data set [2] which is specifically designed for semantic evaluation. In the following, we briefly explain each test collection and the size of its data:

- *SemEval 2014 Task 10*: The goal of the task is to measure the semantic similarity of two sentences and express it as a similarity score. The data set covers different topics such as news, image description, discussion forums and tweet comments, comprising an overall of 4500 sentence pairs. Participating systems are compared by their correlation between the system output and the human-annotated gold standard. Although the task is in Paraphrasing field and not Information Retrieval, we examine the sanity of our systems by understanding its performance in comparison to other participants.

- *MediaEval Retrieving Diverse Social Images Task 2013/2014*: As mentioned, the task focuses on result relevance and diversification in social image retrieval. The data set of both the 2013 and 2014 editions consists of about 110k photos of 600 famous world locations (e.g. the Eiffel tower). Each location is provided with a ranked list of photos,

a representative text, Flickr's metadata, a Wikipedia article of the location, and only for 2014 version user tagging credibility estimation.

- *CLEF-IP Claims to Passage Task 2013*: The task provides a large, clean data set for experimentation of patent data in English, German, and French languages. The purpose of the passage retrieval task is to find passages relevant to a given (set of) patent claim(s). The test collection contains almost 1.5 million patents, stored into approximately 3.5 million Xml documents.

1.4 Structure of the Work

The remainder of this study is organized as follows: Chapter 2 provides the background on text processing, semantic term representation and statistical measures as well as approximation techniques for finding the nearest neighbors. Chapter 3 studies the related work and state of the art of semantic text similarity and also retrieval techniques in the mentioned domains. Then, Chapter 4 describes the architecture of the framework as well as text processing and similarity algorithms, followed by suggesting two optimization techniques. The outline of the experiments as well as the results are discussed in Chapter 5. We conclude the study and propose future research directions in Chapter 6.

CHAPTER 2

Background

In this chapter, we provide a comprehensive background on the methods and tools used in this work. First, we explain main text preprocessing techniques for cleaning and extracting data. Then, a variety of statistical measures for similarity and significance, as well as their usage in Information Retrieval is discussed. Afterwards, we thoroughly study a range of semantic representation approaches, comparing them based on the existing literature. Finally, we explain nearest neighbor search—a method to find the closest nodes in an n-dimensional space—as well as two of its optimization approaches.

2.1 Text Preprocessing

Preprocessing is an important step in Data Mining and Information Retrieval, which aims to remove irrelevant and redundant data and prepare it for the analyzing step. The phrase 'garbage in, garbage out' points out the fact that a lack of effective preprocessing can produce misleading results. In particular, Text Mining—as a branch of Data Mining—introduces specific preprocessing techniques in order to obtain the key features from text documents. The objective of these methods is to select the significant keywords that carry the meaning of the text and discard the terms that do not contribute to distinguishing between the documents [49]. In the following, we explain five text preprocessing steps: Tokenization, removing Stop Words, Normalization, Stemming and Lemmatization, and finally Part-Of-Speech (POS) tagging. The techniques are broadly used in Text Mining, Information Retrieval and Natural Language Processing (NLP).

Tokenization

Tokenization is the process of breaking a text into meaningful elements (tokens). A token is in fact a sequence of characters grouped together, representing a useful unit for processing. Tokens can be individual terms, phrases or even whole sentences but generally they are taken to be terms. The trivial strategy to achieve tokens is splitting the stream of text on whitespace. While it generally tokenizes most sections of the text, many cases still need specific considerations. For

Table 2.1: Standard stop words used in Lucene

a	an	and	are	as	at	for	if	in	into
is	it	no	not	of	on	or	such	that	the
their	then	there	these	they	this	to	was	will	with
be	but	by							

example, the apostrophe used in the verb *isn't* should be processed differently rather than the one in the name *O'Briens*. In addition, instead of separating, sometimes the tokenizer should merge the terms to represent them as one token (e.g., *New York*, or *Technical University of Vienna*). While the discussed cases are generally related to English, each new language also presents new issues. For example de-compounding nouns in German (*Suchmaschine* 'Search Engine'), or two-sectioned verbs in Farsi (جستجو کردن 'to search') [35].

Removing Stop Words

Stop words are terms which are assumed as non- or less-informative and need to be excluded from the vocabulary entirely. The list of stop words depends on the domain of the text, the objective of the system, and the language. Nevertheless, the general strategy for determining a stop list is to sort the terms by collection frequency and select the most frequent ones. Table 2.1 shows a list of common stop words for English used in Lucene[1]. Removing stop words significantly reduces the processing time when analyzing the data. In addition, it normally increases the overall performance of the system by dropping the terms that have little value in helping to select documents [35].

Normalization

Text Normalization is the process of transforming the source text into a standard, less variable form. This step requires awareness about the type of the text as well as the needs of the next steps. Here is a brief description about some important Normalization techniques:

- *Homogeneous Case:* It transforms the terms to capital or small case in order to prevent ambiguity between the same terms written in different letter cases.
- *Abbreviation Expansion:* This process reconstructs full terms or sentences from an abbreviated term using a dataset of abbreviations. Since an abbreviation may also have many expanded forms, each in different topics, the expansion process also needs an understanding of the context of the text.
- *Markup and Punctuation Removal:* Markups, often called 'tags', are sequence of characters or other symbols which describe the document's logical structure (e.g., XML, or

[1]Lucene is an open source popular Information Retrieval library, also known as a search engine library, available on `http://lucene.apache.org`

Table 2.2: Result of applying Stanford NLP POS tagger.

Original Text	Exerting yourself to the fullest within your individual limits: that is the essence of running, and a metaphor for life and for me. Haruki Murakami
Stanford NLP POS tagger	Exerting/VBG yourself/PRP to/TO the/DT fullest/JJS within/IN your/PRP$ individual/JJ limits/NNS :/: that/DT is/VBZ the/DT essence/NN of/IN running/VBG ,/, and/CC a/DT metaphor/NN for/IN life/NN and/CC for/IN me/PRP ./. Haruki/NNP Murakami/NNP

Table 2.3: A set of part of speech tags and their abbreviated forms

CC	Coordinating conjunction
DT	Determiner
IN	Preposition or subordinating conjunction
JJ	Adjective
JJS	Adjective, superlative
NN	Noun, singular or mass
NNP	Proper noun, singular
NNS	Noun, plural
PRP	Personal pronoun
PRP$	Possessive pronoun
TO	to
VBG	Verb, gerund or present participle
VBZ	Verb, 3rd person singular present

HTML tags). Markups as well as unnecessary punctuation are removed in this phase, usually done by regular expression.

Part of Speech Tagging

Part Of Speech (POS) tag refers to the label of a term in its narrow context which gives relevant information about its grammatical category such as noun, verb, adverb, adjective, pronoun, conjunction, etc.. Part of speech tagging is the process of identifying the POS of a term in a text, based on both its definition and context. In other words, POS tagging attempts to resolve the ambiguity of part of speech tags of a term when the term can belong to more than one part of speech (e.g., *run* is both noun and verb).

Table 2.3 shows a set of POS tags and their abbreviated forms. As an example, we run Stanford NLP POS tagger [57] on an arbitrary example text and show the results in Table 2.2.

There are mainly two type of taggers: rule-based and stochastic. Rule-based taggers use

hand-written rules based on the grammar of a language. In contrary, stochastic taggers use statistical methods with no use of pre-existing conventions. They generally choose the tag by maximizing the probability of occurring of term and tag sequences based on training data [7]. In practice, taggers either definitely identify the tag for the given term or make the best guess based on the available information, which can cause occasional errors. In addition, POS tagging is expensive and time consuming in comparison to other discussed text processing methods.

Stemming and Lemmatization

For grammatical reasons, a term can appear in documents in different forms (e.g., *make*, *makes*, and *making*). Additionally, some terms, derived from the same roots, have similar meanings (e.g., *bureaucracy* and *bureaucratic*). Since these are different forms of the terms with the same root, it is then useful from an IR system to search for one common form instead of considering them as different terms. Stemming and lemmatization aim to tackle this issue by reducing inflectional forms and sometimes derivationally related forms of a term to a common base form.

Lemmatization methods try to return the base or dictionary form of a term by use of a vocabulary and morphological analysis of terms. For example, passing the term *geese* to NLTK WordNet lemmatizer [60], it fetches the correct singular form (*goose*) from its internal dictionary. These methods also takes into account the part of speech tag of the term such that for example, *loving* as verb is lemmatized to *love*, while the lemmatizer returns *loving* when the input is considered as noun.

Stemming refers to an algorithm, written for a specific language, that heuristically cuts off suffixes and prefixes. These algorithms generally consist of different rules and conventions to reduce the terms step by step. Unlike lemmatizers, stemmers operate on a single term without considering its position in the sentence, and therefore cannot discriminate between terms which have different meanings depending on parts of speech. Revisiting the mentioned examples, NLTK Porter stemmer [60] returns *gees* as an incorrect singular form of *geese* and *love* regardless of the part of speech of *loving*. However, stemmers are typically easier to implement and also faster to run, but as the examples show, they have less accuracy in comparison to lemmatization. There exist a variety of English stemmers, such as *Porter stemmer*, *Lancaster Stemmer*, and *Snowball Stemmer* [35].

Table 2.4 shows the output of the NLTK Porter and Lancaster stemmer as well as WordNet lemmatizer. The Lancaster stemmer applies a more aggressive approach in comparison to the NLTK Porter such that the terms *open*, *operation* and *operate* all result to *op*. Among the results, the lemmatizer performs better as it also considers the part of speech of *operation* and discriminate it from the verb *operate*.

2.2 Statistical Measures

Statistical measures are of key importance in IR evaluation and retrieval. In this section, we explain two main categories: Similarity and Significant tests. Similarity measures interpret the position of a data point in comparison to another one. Statistical significance tests examine

Table 2.4: Result of applying NLTK Porter, Lancaster stemmer, and WordNet lemmatizer [60]

Original Text	an open market operation is an activity by a central bank to operate on the market
NLTK Porter Stemmer	an open market oper is an activ by a central bank to oper on the market
NLTK Lancaster Stemmer	an op market op is an act by a cent bank to op on the market
NLTK WordNet lemmatizer	an open market operation is an activity by a central bank to operate on the market

whether the difference between two sets of data most likely reflects a 'real' difference in the population from which the data were sampled.

In the following formula of this section, the arbitrary data point P is defined as follow:
$P = (p_1, p_2, \ldots, p_n) \in \mathbb{R}^n$

Cosine Similarity

The Cosine measure is a very popular measure in text mining and IR, which measures the cosine of the angle between two vectors using dot product:

$$Cos(P, Q) = \frac{P \cdot Q}{\|P\|\|Q\|} = \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n} (p_i)^2} \times \sqrt{\sum_{i=1}^{n} (q_i)^2}} \tag{2.1}$$

by normalizing the vectors ($|P| = |Q| = 1$), we also have:

$$Cos(P, Q) = \sum_{i=1}^{n} p_i \times q_i \tag{2.2}$$

An interesting characteristic of this metric is that it measures the orientation and not magnitude so that it considers the angle between the documents instead of the value of each dimension. In fact, it can be seen as a comparison between vectors on a normalized space. Another reason for the popularity of Cosine similarity is that it is very efficient, especially for sparse vectors, as only the non-zero dimensions need to be considered.

Euclidean Distance

Euclidean distance is another popular measure which has it roots in geometry as the 'ordinary' distance between two points in Euclidean space. The Euclidean distance between points P and

Q is the length of the line segment connecting them, defined as follows:

$$D_E(P,Q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}. \tag{2.3}$$

Cosine similarity is closely related to Euclidean distance so that each can be converted to the other as follow:

$$D_E(P,Q) = \sqrt{|P|^2 + |Q|^2 - 2|P||Q|Cos(P,Q)}. \tag{2.4}$$

after normalizing the vectors ($|P| = |Q| = 1$), we have:

$$D_E(P,Q) = \sqrt{2 - 2Cos(P,Q)}. \tag{2.5}$$

Minkowski Distance

The Minkowski distance is a generalization of the Euclidean distance defined as follows:

$$\left(\sum_{i=1}^{n}|p_i - q_i|^x\right)^{1/x} \tag{2.6}$$

As it is clear from the formula above, Euclidean distance can be achieved by setting $x = 2$. Manhattan distance is also a specific case of Minkowski where $x = 1$.

Mahalanobis Distance

Mahalanobis distance provides a powerful method of measuring how similar some set of conditions is to an 'ideal' set of conditions. It is based on both the mean and variance of the variables and takes advantage of the covariance matrix among all the variables. The Mahalanobis distance of an observation P from a group of observations with mean $\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_N)$ and covariance matrix S is defined as:

$$D_M(P) = \sqrt{(P-\mu)^T S^{-1}(P-\mu)} \tag{2.7}$$

The metric addresses several of the limitations of the Euclidean distance such that:

- It automatically accounts for the scaling of the coordinate axes.
- It corrects for correlation between the different features.
- It can provide curved as well as linear decision boundaries.

However, the covariance matrices can be hard to determine accurately, and the memory and time requirements grow quadratically rather than linearly with the number of features. These problems can become quite serious specially when the number of features becomes large.

Kullback–Leibler Distance

Kullback–Leibler (KL) divergence is a non-symmetric measure of the different between two probability distributions over the same event space such that:

$$D_{KL}(P||Q) = \sum_{i=1}^{n} p_i log \frac{p_i}{q_i} \tag{2.8}$$

Kullback-Leibler Distance (KLD) generalizes KL divergence to a symmetric distance metric:

$$D_{KLD}(P||Q) = D_{KL}(P||Q) + D_{KL}(Q||P) \tag{2.9}$$

KLD has been used in many applications in Information Retrieval (e.g., query expansion, topic identification, etc.) as well as other domains like speech processing based on statistical language modeling.

Pearson Correlation

Pearson Correlation calculates the linear relationship between two sets of data and returns a value between -1 and $+1$ where $+1$ is total positive correlation, 0 is no correlation, and -1 is total negative correlation. Given two samples P and Q, the Pearson Correlation r is calculated as follows:

$$r = \frac{\sum_{i=1}^{n}(p_i - \bar{P})(q_i - \bar{Q})}{\sqrt{\sum_{i=1}^{n}(p_i - \bar{P})^2}\sqrt{\sum_{i=1}^{n}(q_i - \bar{Q})^2}} \tag{2.10}$$

where $\bar{P}$ and $\bar{Q}$ is the sample mean of the samples P and Q.

Paired Significance Test

Paired Significance Test examines the significance of the difference between two sets of data by calculating the difference of each pair of the data sets and then running another method called 'one-sample t-test'. One-sample t-test, as a common significance method, aims to reject the 'null hypothesis' defined as follow:

'The hypothesis that there is no significant difference between specified populations, any observed difference being due to sampling or experimental error.' [61]

Every null hypothesis is defined in a significance level (usually 0.10, 0.05, or 0.01) which is the probability of observing the given difference of the hypothesis is true. In order to examine one-sample test, in the first step, the z value is calculated as follows:

$$z = \frac{\bar{P} - \mu_0}{\sigma}\sqrt{n} \tag{2.11}$$

where $\bar{P}$ is the sample mean, σ is the sample standard deviation of the sample, n is the sample size and finally μ_0 is the hypothesized population mean. Once the z value is determined, another value called p is fetched from the Student's t-distribution table. If the calculated p-value is below the threshold chosen for statistical significance, then the null hypothesis is rejected.

2.3 Semantic Representation

As it is mentioned before, semantic representation methods provide a mapping of text elements (documents, paragraphs, terms, etc.) to a vector space in which each dimension indicates a proximity of concepts. Study on semantic representation has attracted researchers' attention as early as 1990 and it is still a challenging research area. In this section, we explain main semantic representation models and compare their characteristics as well as limitations.

Latent Semantic Analysis/Indexing

In the text retrieval community, text semantics started with Latent Semantic Analysis/Indexing (LSA/LSI), which initiated a new trend in surface text analysis. LSA represents the meaning of a term as a kind of average of the meaning of all the passages in which it appears. Given a text corpus, it uses matrix-based methods to model the patterns between the terms and the concepts of the text [12]. The phrase 'Latent Semantic' refers to the fact that the model finds semantic relations of the terms that are latent in the collection of text.

Calculating LSA is based on *Singular Value Decomposition (SVD)*, a mathematical matrix factorization technique broadly used in signal processing and statistics. SVD provides powerful mathematical analysis for LSA and sets it apart from the approaches which use co-occurrence counts or simple contiguity frequencies [25]. SVD is defined as follows:

Given an $M \times N$ matrix C, then there is SVD of C in the form of

$$C = U\Sigma V^T \tag{2.12}$$

where:

- U is an $M \times M$ unitary matrix (i.e., $U.U^T = U^T.U = I$, when I is the identity matrix)
- Σ is an $M \times N$ diagonal matrix, where $\Sigma_{ii} = \sigma_i = \sqrt{\lambda_i}$, with eigenvalues $\lambda_i \geq \lambda_{i+1}$ for $1 \leq i \leq r$, and zero otherwise.
- V^T is an $N \times N$ unitary matrix, which is the transpose of unitary matrix V

In order to create Latent Semantic Index, first we build term-document matrix C in which terms and documents are the rows and columns of the matrix. The cells of the matrix are the results of an arbitrary weighting function which, in the simplest case, is the existence of the respective term in the document. An example of six documents and five terms (obtained from Manning et al. [35]) is shown in Table 2.5. Considering the representation of the documents in matrix C, the cosine similarity between the document d_1 and d_2 is equal to 1 which is the same between d_1 and d_5. Looking in the data, we can observe close relation between documents d_1 and d_2, as they use the terms *ship* and *boat* which are conceptually related. In fact, while the document d_1 and d_2 share more common concepts, their similarity value is the same as that of d_1 and d_5.

The LSD model addresses this issue, by first applying SVD to the matrix C which results to three matrices: U known as *term matrix*, Σ as *singular values* shown in Table 2.7 and finally V^T as *document matrix* represented in Table 2.6. The matrix V^T, specifically, provides a new

Table 2.5: A simple example of term-document matrix C (binary) achieved from Manning et al. [35]

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

Table 2.6: Matrix V^T achieved by applying SVD on matrix C in Table 2.5 fetched from Manning et al. [35]

	d_1	d_2	d_3	d_4	d_5	d_6
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

representations of the documents where the conceptual relations of the terms are also considered. Using these representations, we repeat measuring the degree of documents similarity with the cosine function. We thus observe a more similar relation between the documents d_1 and d_2 (-0.0028) in comparison to d_1 and d_5 (-0.2718).

The matrix Σ (Table 2.7) quantifies the importance of each dimension (here term) in the text, sorted from the most to least important one. Using this matrix, we can reduce the noise by 'zeroing out' the less important dimensions. In fact, we keep the critical information by getting rid of the 'details'. Table 2.8 shows the result of dimension reduction by zeroing all but the two largest singular values of Σ [35]. In order to observe the effect of noise reduction, we again measure the documents similarities using the new representations. In this case, the cosine similarity between the document d_1 and d_2 is $+2.0236$ while d_1 and d_5 has the similarity of $+0.973$ which shows that how the documents with more conceptual relations become more similar in the vector space.

Regardless of the success of LSA, the method is limited in practice by efficiency and scalability issues caused by significant computational cost of SVD. In addition, the co-occurrences of new documents are not captured by other pre-built vectors which causes degrading in the quality of the representation until the model is recreated from scratch.

Explicit Semantic Analysis

Explicit Semantic Analysis (ESA) is one of the early alternatives of LSA, which exploits the knowledge of ontologies or encyclopedias to explicitly define the concepts of the text. ESA

Table 2.7: Matrix Σ achieved by applying SVD on matrix C in Table 2.5 fetched from Manning et al. [35]

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

Table 2.8: Result of dimension reduction of matrix C in Table 2.5 obtained from Manning et al. [35]

	d_1	d_2	d_3	d_4	d_5	d_6
1	-1.62	-0.60	-0.44	-0.97	-0.70	-0.26
2	-0.46	-0.84	-0.30	1.00	0.35	0.65

defines the terms as a weighted vector of 'natural' concepts defined by humans.

As a realization of the ESA model, Gabrilovich et al. [17] uses Wikipedia articles (e.g., Family, Electronic, or Norway) to indicate the manifest concepts grounded in human cognition. Figure A.1 obtained from Gabrilovich et al. [17] depicts the steps of the process. In the first step, an inverted index is built, which maps each term into a list of concepts (Wikipedia articles) in which it appears. This inverted index can be interpreted as a primary representation of the terms by explicitly defined concepts. In the next step, using the vector representation of the terms of the input text, the text is transformed to weighted vectors of concepts using machine learning and text categorization techniques. The meaning of a text fragment is thus interpreted in terms of Wikipedia concepts (articles).

Since ESA relies on co-occurrence of the terms, it uses much less computational resources in comparison to LSA. As an another advantage of ESA, the built weighted inverted index can be applied many times on the input documents with no need of semantically indexing them. However, unlike LSA, ESA relies on a pre-existing set of concepts, which may not always be available or related to the retrieval domain [26].

Random Indexing

Random Indexing (RI) [53] introduces a more efficient while still effective method in comparison to LSA and ESA. The method creates representation vector for each term with a much smaller dimensionality than the whole number of the documents by accumulating the occurrence of the terms in their contexts. Creating the Random Indexing model is described as follows:

- In the first step, a unique and randomly generated representation, called *index vector*, is assigned to each context. These index vectors are sparse so that just a small number of their elements are randomly set to +1 or -1, and the rest are 0.

- Then, while reading the text, everytime a given term occurs, the index vectors of the terms in its context (e.g. in a document, or within a sliding context window) are added to the *context vector* of the term. Terms are thus represented by context vectors that are effectively the sum of the terms' contexts.

Random Indexing has the benefit of being incremental such that the representation model can be enriched by new documents, with no need of re-building whole the vectors. In addition, it operates with significantly less resources in comparison to LSA/LSI. Furthermore, Random Indexing, unlike ESA, does not rely on any pre-existing knowledge and extracts the meanings of the terms from the context of the text.

In order to take advantage of the performance of LSA while addressing the issue of SVD tractability, Sellberg et al. [54] uses the RI vectors to create SVD matrix with lesser dimensionality. They introduce three alternatives: the standard LSA, the standard Random Indexing, and finally SVD on a reduced RI matrix. They evaluate these three approaches on a subset of MEDLINE-corpus [41]. As shown in Figure A.2 and Figure A.3 (obtained from the paper), while SVD on reduced RI matrix is about twice as fast as SVD on non-reduced matrix, the performance of their approach is only slightly worse than the pure LSA.

Word2Vec

Word2Vec [37] has gained a lot of attraction lately, by providing state-of-the-art term representation. The model is based on *Deep Learning*, an effective unsupervised Machine Learning method. Deep Learning models high-level abstractions in data by composing multiple non-linear transformations in hidden-layers of a neural network. Word2Vec is highly incremental and scalable and, in principle, allows exploiting the implicit knowledge within corpora. In practice, when trained on large datasets, it captures many linguistic subtleties, such as semantic and syntactic relations that allow basic arithmetic operations within the model. In the following, we explain the architecture of the model as well as its characteristics.

The Word2Vec model is initiated from the idea of *Recurrent Neural Network Language Model (RNNLM)* [38]. The main difference between RNNLM and the standard neural networks is that the model uses a recurrent matrix that provides time-delayed connections within the hidden layer. The architecture of RNNLM is illustrated in Figure 2.1 obtained from Mikolov et al. [39]. The architecture consists of an input layer w(t), a hidden layer at current time s(t), recurrent connection of hidden layer to previous time line s(t-1), plus the output layer y(t). The input vector represents input term, and the output layer represents a probability distribution over terms. The hidden layer provides a representation of the sentence maintained by an inter-linked memory (time-delayed hidden layer).

Since RRNLM is computationally expensive, Mikolov et al. [37] propose two alternative architectural models: *Continuous Bag-Of-Words (CBOW)* and *Skip-gram*. The models try to replace the non-linear hidden layer of the RRNLM with simpler but still effective substitutes. The approaches are depicted in Figure 2.2 fetched from Mikolov et al. [37]. CBOW predicts the current term based on its context, by averaging all the terms in the window. In fact, it projects the terms into the same position while using a shared weight matrix between the input and the projection layer for all the term positions. The Skip-gram approach is similar to CBOW, but

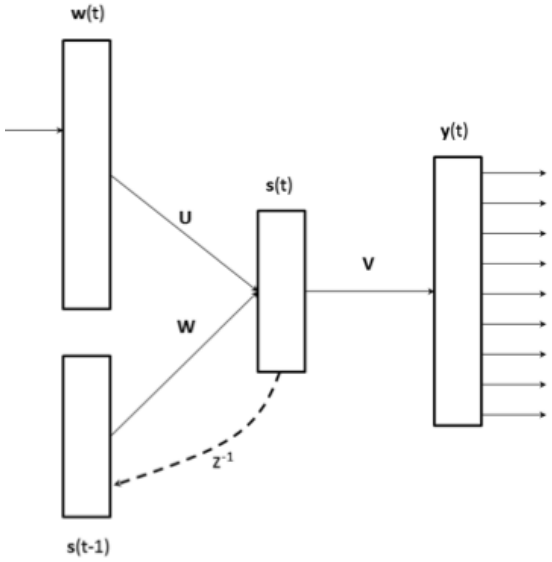

Figure 2.1: Recurrent Neural Network Language Model for term representation obtained from Mikolov et al. [39]

it tries to maximize classification of a term based on an another term in the same sentence. In other words, each term is an input to a log-linear classifier which aims to predict the terms in its window. In addition to the prediction result, a weight value is assigned to each connection based on the distance of the terms in the context window. In fact, using this weight value more distant terms influence to a smaller extent. Comparing to CBOW, Skip-gram shows better performance for infrequent terms while being computationally more expensive.

In addition to the architecture alternatives, Word2Vec introduces two additional parameters for discarding noise: The first is *rare-term pruning* which removes the terms that appear less than a minimum frequency in the input data (default for skip-gram is 10, and for CBOW 5). The second is *sub-sampling* which down-samples frequent terms in order to increase the effective window size. This noise-reduction feature is motivated by the idea that frequent terms are less informative. Therefore, Mikolov et al. [37] heuristically defines the sub-sampling feature such that it discards each term w_i in the training set with probability computed by the formula:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.13}$$

where $f(w_i)$ is the frequency of term w_i and t is an input parameter, typically between $1e^{-5}$ to $1e^{-3}$. As it is clear from the formula, the higher the value of t, the more aggressive is this feature for discarding frequent terms.

Although the model itself does not have any specific knowledge about the linguistic subtleties (i.e., syntax, morphology, or semantics), training the model on big corpus of text provides term representations with striking syntactic and semantic properties. For example, the relation

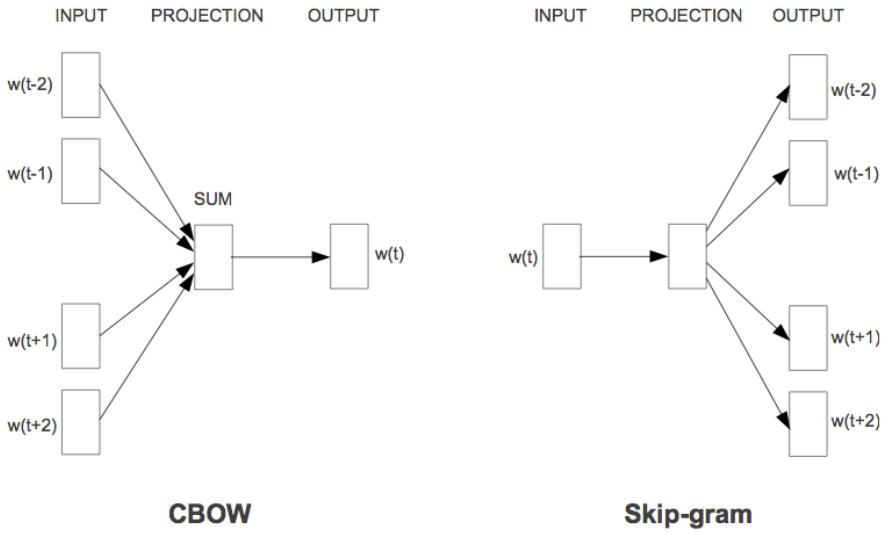

Figure 2.2: Two proposed architectures for the Word2Vec model obtained from Mikolov et al. [37]

between male and female is automatically learned such that an arithmetic operation on vector representations of '*King* - *Man* + *Woman*' results in a vector very close to *Queen* [39]. Figure A.4 obtained from Mikolov et al. [39] shows the similar relation between countries and their capitals projected in two dimensional space. In addition to semantic relations, the model also discovers the syntactical relations between terms. For instance, given the vector representations of the terms *biggest*, *big* and *small*, we can simply compute the vector '*biggest* - *big* + *small*' to which the representation of the term *smallest* is the closest node [37].

2.4 Nearest Neighbor Search

In this section, we study *nearest neighbor search* (also known as proximity search) as well as two of its optimization techniques, namely *KD-tree* and *Ball-tree*. Both the optimization methods apply tree-based data structures as well as approximation techniques to improve the query time. We also discuss the effective factors in the choose of the optimal nearest neighbor search approach.

Nearest Neighbor

The principle behind nearest neighbor method is to find a predefined number of samples closest in distance to a new point. The method can be also used as a classification algorithm where each node has defined labels and the method predicts the label of a new node based on an aggregation of the neighbors' labels. The number of selected neighbors can be a user-defined constant (k-nearest neighbor), or vary based on the local density of points (radius-based neighbor search).

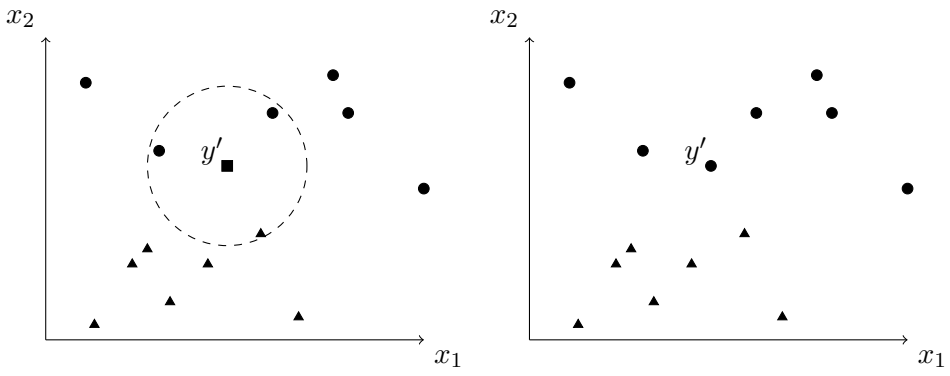

Figure 2.3: 3-NN classification of new data point y'

Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits or satellite image scenes.

Formally, given a set of data points $\{y_1, ..., y_n\}$ and a query point y', each with a set of features $\{x_1, ..., x_m\}$, k-Nearest Neighbor (k-NN) calculates the distance of y' to all the other nodes and returns the k nearest ones. The distance can, in general, be any metric measure defined in Section 2.2 while standard Euclidean distance is the most common choice.

Figure 2.3-left shows an example of k-NN classifier with data points in two-dimensional feature space ($m = 2$). The nodes are classified into categories *circle* and *triangle*. Given a new data point y', we want to classify it based on 3 nearest neighbors. As two out of three neighbors (a probability of $66.\dot{6}\%$) are *circle*, k-NN choose the label *circle* for y' shown in Figure 2.3-right [19]. This way of choosing the class is also referred to as majority voting.

Optimization Methods

Fast computation of nearest neighbors is an active area of research. The basic implementation of nearest neighbor search is the computation of the distance between the query to all the points in the dataset (Brute-Force). For N samples in D dimensions, the computation complexity of this approach is $O[DN^2]$. Although Brute-Force neighbors search performance is very competitive for small data samples, as the number of samples, the approach quickly becomes infeasible.

In order to address the computational inefficiencies of the Brute-Force method, we study two optimization approaches, both using tree-based data structures. In general, these approaches attempt to reduce the required number of distance calculations by efficiently encoding aggregate distance information for the sample. The basic idea is that if point A is very distant from point B, and point B is very close to point C, then we know that points A and C are very distant, without having to explicitly calculate their distance. In this way, the computational complexity of a search can be reduced to $O[DN \log(N)]$ or better which is a significant improvement over Brute-Force for large N [45].

The first approach is KD-tree data structure (short for K-Dimensional tree), which generalizes two-dimensional Quad-trees (every node has exactly four children) to an arbitrary number of dimensions. The KD-tree recursively partitions the space along the Cartesian axes which makes

its construction very fast as D-dimensional distances does not need to be computed. Once constructed, the nearest neighbor of a query point can be determined with only $O[\log(N)]$ distance computations. Although the KD-tree approach is very fast for low-dimensional data ($D < 20$), it becomes inefficient as D grows very large [5].

The second approach is Ball-tree data structure which addresses the inefficiencies of KD-trees in higher dimensions. Ball-tree partitions data in a series of hyper-spheres. This makes tree construction more costly than that of the KD-tree, but results in a data structure which can be very efficient on highly-structured data.

A Ball-tree recursively divides the data into hyper-spheres, defined by a centroid C and radius r, such that each point in the data lies within one of them. With this data structure, a single distance calculation between a test point and the centroid is sufficient to determine a lower and upper bound on the distance to all points within the hyper-sphere. Because of the spherical geometry of the Ball-tree nodes, it can out-perform a KD-tree in high dimensions, though the actual performance is highly dependent on the structure of the data [42].

Comparison

The optimal algorithm for finding the nearest neighbor is a complicated choice, and depends on a number of factors. We discuss the effect of three main factors: *Dimensionality*, *Number of Samples*, and *Number of Neighbors*.

The first studied factor is dimensionality. As mentioned before, Brute-Force time complexity is $O[DN]$ (D and N stand for the dimensionality and the number of samples) which clearly makes it very sensitive about increasing the dimensionality. KD-tree query time changes with D in a way that is difficult to precisely characterize. For $D \leq 20$, the cost is approximately $O[D\log(N)]$ which is the same as Ball-tree, while for larger D, it can become very inefficient so that the cost increases to nearly $O[DN]$. Considering the overhead due to the creating of the tree structure, it can lead to queries which are even slower than Brute-Force. The Ball-tree query time however grows as approximately $O[D\log(N)]$ and therefore the method is a better choice for high dimensional data.

The other important factor is the number of samples. For small data sets ($N \leq 30$), $\log(N)$ is comparable to N, and Brute-Force algorithms can be more efficient than a tree-based approach. In this special case of small size of samples, both KD-tree and Ball-tree address the efficiency problem through providing a *leaf size* parameter. Leaf size refers to the maximum number of nodes inside the hyper-spheres of the Ball-tree or the leaves of the KD-tree. Therefore, in this case, both the algorithms turn to an usual Brute-Force search although the overhead of creating the tree structures still makes them slower than the Brute-Force approach.

The last studied factor is the number of neighbors (k) requested by the query point. As Brute-Force computes the distance to all the nodes, its query time is completely unaffected by the value of k. However, Ball-tree and KD-tree will become slower as k increases. As k becomes large compared to N, the ability to prune branches in a tree-based structure is reduced so that the Brute-Force search can be even more efficient [45].

CHAPTER 3

Related Work

We review the related work and literature in three sections. First, we study the state-of-the-art research on the use of semantic representations and concept-based IR systems. We then study the current work in social media domain and specifically social image retrieval. Finally, we explain patent domain and review the research in patent retrieval.

3.1 Semantic Similarity

Wittgenstein [62] stated that the meaning of words must depend on the use and the context of these words. If two words occur often in the same context, then they are more likely to share a similar meaning. As mentioned in Chapter 2, based on this idea many semantic representation methods with different computation techniques (matrix-based, count-based, neural networks, etc.) have been introduced. Naturally, these methods provide different conceptual representations of the terms and hence they have different performance.

In order to compare the representation methods, Blacoe and Lapata [6] run experiments on terms similarity as well as paraphrase detection tasks. They introduce a new neural network based model and compare it with two count-based representations. They observe a better result for count-based methods in the phrase similarity task, whereas the representations show comparable performance in the paraphrase detection. As Mikolov et al. [39] introduce Word2Vec (Section 2.3), they also compare it with LSA using a 'semantic' as well as a newly created 'syntactic' test collection. Both test sets put forward analogy questions of the form '*a* to *b* is like *c* to *what*?'. The 'syntactic' test collection contains questions with the format of singular/plural, base/comparative/superlative, etc., while the 'semantic' provides pairs of the form *Class-Inclusion:Singular-Collective* (e.g., '*electronic device* to *tv* is like *clothing* to *shirt*'). They eventually find that Word2Vec is highly superior in all the tasks. However, they provide very little details about the created LSA representation models.

While the mentioned papers compare representation methods, their focus is on evaluation of the introduced models. In fact, they are still lacking providing a comprehensive comparison. In

order to address this problem, Baroni et al. [4] systematically compare a neural network based approach, namely Word2Vec, with classical count-based approaches. They create a set of models with different parameter settings and evaluate them across a wide range of lexical semantics tasks. The experiment consists of 5 tasks: The first is *Semantic Relatedness* which measures the topical (*family*/*planning*) or co-hyponymy (*king*/*queen*) relations. *Synonym Detection* for verbs with similar meanings and *Concept Categorization* (e.g., *dogs* and *giraffes* belong to *mammals* class) are introduced as the second and third task. The forth task is *Selection Preferences* which, in a verb-noun relation, assesses the typicality of the noun as a subject or object of the verb (e.g., *people* has higher score of being subject of *to eat* than being its object). Finally *Analogy* task uses the test collections introduced by Mikolov et al. [39] (see above). They observe an overall better performance of Word2Vec than the classic count-based methods in all the tasks. All of these tasks are however not directly applicable for an information retrieval scenario.

Exploiting the semantic representations, many information systems study text similarity techniques in order to facilitate a better text (document, sentence, word) retrieval. Rus et al. [51] provide a general-purpose semantic information retrieval system called SIMILAR. It implements a number of algorithms for assessing the semantic similarity between two texts. The algorithms combine the statistical semantic corpus-based methods with knowledge-based resources. As an example, they try to find an optimal assignment from words in the source text to words in the destination one by introducing an state-of-the-art algorithm based on the Quadratic Assignment Problem (QAP). Pilehvar et al. [46] focuses on aligning the words of a text to a set of semantically related words in the other text. They use POS tags to disambiguate the meaning of the words and finally introduce a unified approach to measure text-to-text similarity.

While the mentioned approaches use NLP or know-ledge based data sets, Kashyap et al. [24] introduce a similarity measure only based on corpus-based features. The method, called *Align-and-Penalize*, first measures the similarity by heuristically assigning words in the texts and then normalizes the result by introducing two penalizing factors: The first factor eliminates term alignments with naive similarity values, and the second penalizes the terms with syntactic contradictions. Similar to many semantic based information systems, it extracts the features of the texts and applies machine learning to tune the results. However, the method is only applied on paraphrasing tasks with short texts and is not tested on an information retrieval scenario.

3.2 Social Image Retrieval

In the social media domain, different modalities (image, text, tags, etc.) often occur together in the same document (scientific paper, website, blog, etc.). It is well known that combining information from multiple modalities assists in retrieval tasks. For instance, the results of the ImageCLEF campaign's photographic retrieval task have shown that combining image and text information results in better retrieval than text alone [44]. There are two fundamental approaches to fusing information from multiple modalities: early fusion and late fusion [13].

Late fusion is widely used, as it avoids working in a single fused feature space but, instead, fusing results by reordering them based on the scores from the individual systems. Clinchant et al. [8] propose and test a number of late fusion approaches involving the sum or product combination of weighted scores from text and image retrieval systems. Difficulties arise from

- weights that must be fixed in advance or that need to be learned from difficult to obtain training data
- modality weights that might be query dependent and
- weights that are sensitive to the IR system performance for the various modalities [13]

Separate queries are needed for each modality, so that for example to find a picture of a cat in a database of annotated images, one would need to provide a picture of a cat and text about the cat. There are ways of getting around this limitation, such as choosing the images for the top returned text documents as seeds in an image search [13], but these are generally ad-hoc.

With early fusion, a query would not have to contain elements from all modalities in the dataset. To continue the previous example, pictures of a cat could be found only with text input. Early fusion suffers from the problem that text tends to sparsely inhabit a large feature space, while non-text features have denser distributions in a small feature space. The simplest approach to early fusion is to simply concatenate the feature vectors from different modalities. However, concatenated feature vectors become less distinctive, due to the curse of dimensionality [13], making this approach rather ineffective. A solution proposed by Maes and Rüger [1] is to transform the feature vectors to reduce the dimension of the text feature vectors and increase the dimension of the image feature vectors using the minimum description length (MDL) principle.

As an alternative to the fusion problem, it is however possible to turn the document into one modality (generally text) by automatically obtaining the bags of 'visual words' [10] from images. These automatic-generated annotations are generally achieved by using machine learning and also clustering local image features. Magalhães and Rüger [30] study the accuracy of different annotations techniques. They split the manual tags into two subsets of amateur and professionals and evaluate them against automatic-generated ones on Flicker image documents. They find similar performance on using amateur-level manual annotations than automatic-generated ones.

Despite the effectiveness as well as limitations of using multiple modalities, many information systems has only used textual features as the main resource of retrieval. For instance, recently, Eskevich et al. [15] considered a wide range of text retrieval methods in the context of multimodal search for medical data, while Sabetghadam et al. [52] used text features in a graph-based model to retrieve images from Wikipedia. However, these works do not particularly exploit text semantics.

Approaching the text semantics, Liu et al. [27] introduced the Histogram for Textual Concepts (HTC) method to map tags to a concept dictionary. However, the method is reminiscent of ESA described above, and it was never evaluated for the purpose of text-based image retrieval.

3.3 Patent Retrieval

Before reviewing the state-of-the-art of the patent retrieval domain, it is worth to take a closer look at the patent documents. As an example of patent documents, Figure A.5 shows the first page of US patent application 836630 obtained from Google Patent[1]. As mentioned in Lupu and Hanbury [29], the application and granted patents consists of four sections:

[1] www.google.com/patents

- *Bibliographical Data:* Title, metadata, the inventors, assignees, domestic filing data, classification data, and relation to the other documents.
- *Abstract:* A brief summary of the invention.
- *Description:* A clear and concise explanation of the known existing work, novelty of the patent, as well as the related technologies.
- *Claims:* Legal definition of the subject. Each claim is a single sentence that defines an invention and its unique technical features, fully supported by the description section.

Patents are typically written in an intricate and hard language in the sense that even an educated, native-speaker can not easily understand them. Specifically, claims have a particular writing style, sometimes referred to as *patentese* [3], which resembles the language of legal contracts. Considering these facts, Lupu [28] studies the characteristics of the patent data from a statistical point of view and compare it with 'regular' English text. He observes considerable difference in the collection frequency of the terms so that the domain is rich in both high and low frequency, while poor in average frequency terms.

Given a patent application, patent examiners closely examine the claims against the existing documents in order to ensure the novelty of the application (*Prior-Art Search*). While in general, typical IR tasks such as web search aim to achieve high precision by retrieving more relevant documents, the Prior-Art search is a recall-oriented task where the objective is to find all possible documents.

In addition to complicated language, the length of the queries is also a difficulty in this domain. Since the queries are typically full patent applications, they are very long and do not have clear focus on information need. In order to address this problem, Mahdabi [34] proposes two approaches to extract essential terms and phrases from a query patent. The first approach extracts generic terms using KL-divergence (Section 2.2) between the terms in the query and the collection. The second approach focuses on extracting specific phrases by generating statistical language models through global analysis of the collection. In practice the methods show unstable performance regarding to different topics. Therefore, the authors provide a framework based on feature extraction and machine learning which decides the proper method for each topic.

Despite the long query, there is often a significant mismatch between the terms in the query and the relevant documents. By analyzing the CLEF-IP corpus[2], Magdy et al. [33] show that 12% of the relevant patents do not share any common terms with the topics after filtering stop words. Magdy and Jones [32] therefore attempt to match the relevant documents by finding synonym terms between them. They generate a database of possible synonym sets for patent data—called SynSet—by using the parallel translated documents of the CLEF-IP collection. Finally, they use the SynSet database to expand the query patents. They observe that this method is the most effective of standard query expansion approaches while it does not show an overall improvement in retrieval effectiveness. Recently, Wang et al. [59] targets improving query expansion in the domain by proposing a semantic-based approach which considers matching the

[2]CLEF-IP track launched in 2009 as a part of CLEF evaluation campaign. The track provides a large, clean data set of patent data in three languages (English, German, French), accessible under the webpage: `http://ifs.tuwien.ac.at/~clef-ip/`

domain of the query regarding to the patent data. In this approach, they first extract domain features using a modified version of TF-IDF scheme[3] as well as International Patent Classification (IPC) codes. In the next step, the input query is matched to a specific domain, based on which a list of expansion terms are generated. Furthermore, the candidate expansion terms are refined based on semantic similarity methods. This approach shows a competitive retrieval performance in comaprison to other state-of-the-art approaches.

While the mentioned studies focus on the content-based retrieval methods, Lupu [28] explores the usability of statistical semantic similarity on patent data. He particularly focuses on Random Indexing, and calculates the document similarity using the model's vector representations. The results of the paper show that the direct application of these methods to domain-specific retrieval challenges is not effective. Considering this, the current work provides additional insight into the use of statistical semantic methods in the patent data.

[3]Term Frequency-Inverse Document Frequency, is a measure to reflect how important a term is to a document

CHAPTER 4

Methodology and Design

In this chapter, we explain in detail our semantic-based approach to retrieve text documents. We start with explaining the architecture of a semantics-based Information Retrieval system, called *TextSim*[1], and its building components. Then, we describe two similarity algorithms introduced in literature, designed to measure the semantic relatedness of two text documents, followed by a comparison between the methods. Finally, we put forward two optimization techniques for the similarity algorithms and point out their characteristics as well as limitations.

4.1 System Architecture

The architecture of the TextSim system is depicted in Figure 4.1. The design of the system follows a similar model to the three-layer architecture model (*Presentation*, *Logic*, and *Data* layer) in software engineering, with focus on semantics text retrieval. We denote the layers as *Text Processing*, *Semantic Similarity*, and *Term Representation*. Analogous to the three-layer architecture, the layers are defined on top of each other. In addition, we define two cross-cutting aspects—*Logging* and *Performance Monitoring*—shared between all the layers. The Performance Monitoring aspect, in particular, measures the execution time of the critical processes in all the layers. In the following, we explain each layer and its main elements in detail.

Text Processing Layer

This layer provides the interface of the system for reading the documents, running the required processes and finally retrieving the results. In the first step, it reads the documents/topics and executes text preprocessing i.e., normalization, removing stop words, stemming, and tokenization (see Section 2.1). Then, the cleaned and tokenized texts are passed to the Semantic Similarity layer for measuring their relatedness. The layer repeats this process and accumulates the similarity relatedness of all the documents to the topics, achieved from the Semantic Similarity layer.

[1] `https://github.com/neds/textsim`

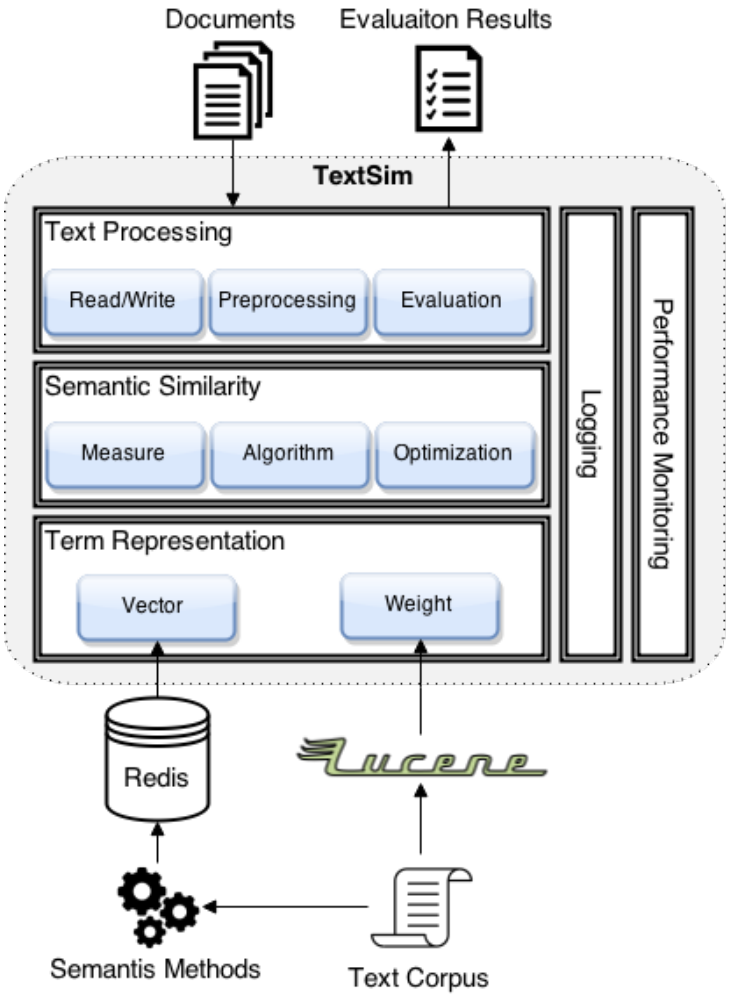

Figure 4.1: TextSim system architecture

Afterwards, this layer generates a ranked list which is generally in TREC[2] format. Finally, it evaluates the list based on the evaluation metric of the data collection and returns the results.

Semantic Similarity Layer

The Semantic Similarity layer consists of three main elements: *semantic similarity* algorithm, *similarity measure* between vector representations, and finally *time optimization* technique. This layer receives two sets of terms from the Text Processing layer and asks from the Term Representation layer for their corresponding vector representations as well as their weights (the importance of the term regarding to the collection). The text-to-text similarity is then calculated based on one or more of the available relatedness measure. Additionally, the layer improves the execution time by applying optimization techniques on the similarity algorithm.

[2] `http://trec.nist.gov`

Term Representation Layer

As mentioned before, this layer provides vector representations of the terms as well as their weights to the Semantic Similarity layer. Outside the platform of the TextSim system, the vector representations are generated from a text corpus using the semantics methods (e.g., Word2Vec, Random Indexing, etc.). The text corpus can be obtained from the input documents as well as other resources. These vector representations are then saved in a database as key-value records which later are fetched by the Term Representation layer. We use Redis[3], a fast key-value data store that allows multi-processing as well as distributed partitioning. The layer reads the term representations from the database and provides internal caching and vector normalization. The weight of the term, as the other element of this layer, is achieved from the Lucene[4] index of the text corpus.

4.2 Similarity Algorithms

In this section, we describe two document-to-document similarity measures, both based on the semantics of the terms in each document. In the first approach, denoted as *SimAgg*, the terms' vectors are aggregated in a document vector and then similarity between two such vectors is computed. The second approach (*SimGreedy*) identifies similar terms between documents and aggregates the similarity values of pairs of terms in the two documents. We compare the methods from the time complexity (efficiency) and performance (effectiveness) point of views.

SimAgg

The SimAgg [53] approach creates a representation vector for each document by aggregating the vectors of the terms in the document. We define the aggregation method as the weighted mean of the elements of the term vectors. The document vector is then obtained as follows:

$$V_A = \sum_{t \in A} idf(t) * V_t \tag{4.1}$$

where t is a term in the document A and V_A and V_t represent the vector representations of document A and the term t. $idf(t)$ as the weight of the term stands for Inverse Document Frequency (IDF) of the term t in the corpus. IDF is a measure whether the term is common or rare across all documents, defined by the following formula:

$$idf(t) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|} \tag{4.2}$$

where $|D|$ is the total number of the documents in the corpus and $|\{d \in D : t \in d\}|$ denotes the number of documents in which the term t appears.

[3] `http://redis.io`
[4] `https://lucene.apache.org`

Having the document vectors of the documents A and B, we calculate the similarity with the cosine function as defined in follows:

$$SimAgg(A, B) = Cos(V_A, V_B) \tag{4.3}$$

SimGreedy

The SimGreedy [36] approach is based on the function *SimGreedy(A,B)* which measures the similarity of the source document (A) to the target document (B). In this method, each term in the source document is aligned to a term in the target document to which it has the highest semantic similarity. Afterwards, the results are aggregated based on the weight of each term. The SimGreedy(A,B) function is defined as follows:

$$SimGreedy(A, B) = \frac{\sum_{t \in A} idf(t) * maxSim(t, B)}{\sum_{t \in A} idf(t)} \tag{4.4}$$

where the function *maxSim* calculates separately the cosine of the term t to each term in the document B and returns the highest value. During the search in the other document, if the maxSim function finds a term with the same surface (sequence of characters), it immediately returns 1 (the maximum similarity value regarding to the cosine function).

SimGreedy is then defined as the average of SimGreedy(A,B) and SimGreedy(B,A):

$$SimGreedy = \frac{1}{2}(SimGreedy(A, B) + SimGreedy(B, A)) \tag{4.5}$$

Comparison

From time complexity point of view, we can observe a significant difference between the methods such that if n and m are the number of terms in documents A and B respectively, the complexity of SimAgg is of order $n + m$ while SimGreedy is of order $n * m$. This difference can be a considerable limitation of SimGreedy in comparison to SimAgg especially when the documents become long (such as, for instance, in the case of patent documents).

Despite the fact that the performance of the methods is revealed by running experiments on different domains, we shed light on two factors that are probably effective in the performance of the methods. The first factor is related to the aggregation function of SimAgg. Since SimAgg aggregates over all the terms of a document, it is thus expected that when the documents become larger, the influence of the individual terms gets more indistinctive and thus the final aggregated vectors become more similar. In contrast, SimGreedy considers the representation of each term separately and then aggregates the results. The second factor is related to the tolerance of the methods against the situation that the terms do not have a semantic representation vector. This situation happens when the vectors are built based on a different corpus (usually a general domain, e.g. Wikipedia) or the terms has occurred so infrequent in the corpus that the semantic representation methods has filtered them out because of not having enough context information. In this case, while SimAgg simply ignore the terms, SimGreedy matches the terms when they occur in both documents. In fact, SimGreedy turns the semantic-based method to a simple term-frequency-based approach when it does not have enough information about the term's meaning.

4.3 Optimization

As mentioned in Section 4.2, the SimGreedy method can be highly inefficient when the documents become larger. In order to address this problem, in the section, we put forward two optimization techniques. Both methods attempt to reduce the execution time of the SimGreedy method without degrading its performance.

Hybrid Search Engines

In the first technique, we turn the procedure into a two-phase process [11]. In order to that, we choose an alternative algorithm with considerably lower execution time in comparison to SimGreedy such as SimAgg or term-frequency-based methods. Then, we apply the faster algorithm to obtain a first ranking of the results and afterwards, the top n percent of the results is re-ranked by applying SimGreedy. Therefore, the SimGreedy algorithm computes only on a portion of the data which is already filtered by the first (faster) one.

Considering the alternative algorithm has the execution time of t and is k time faster than SimGreedy, applying this approach takes $t + t \cdot k \cdot n/100$ where n is the percentage of the selected data. In fact, this approach is $k/(1 + k \cdot n/100)$ time faster than running the SimGreedy algorithm standalone. While achieving better execution time, the choice of the parameter n can reduce the performance. Finding the optimal value for n such that the performance remains in the range of significantly indifferent than non-optimized SimGreedy is a complicated choice. We show exploration result in Section 5.2

Approximate Nearest Neighborhood Index

As discussed in Section 2.4, nearest neighbor search attempts to find the closest neighbors in a vector space. Considering this method, we can adapt the maxSim function of the SimGreedy approach to a nearest neighbor search where it returns the closest node ($k = 1$) of a term. In this approach, we should first create an optimized nearest neighbor data structure for each document (semantic-based index) and then use the index to find the most similar terms.

The overhead time of creating the indexes depends on different factors such as vector's dimension, the number of terms in the document, and the selected data structure. Although this excessive time can influence the overall execution time, it can be especially effective when the indexes are used frequently by many queries. In Section 5.2, we show concrete results regarding this optimization method.

CHAPTER 5

Experiments and Results

In this chapter, we explain in detail the experiment procedure and report the achieved results. We start with examining the similarity methods and semantic representations on a paraphrasing task which is specifically tailored for semantic evaluation. A good result in this tasks would confirm the ability of the methods for identifying semantic similarity in general texts. In the next step, we focus on the use of the semantic methods on social image retrieval domain. After running a wide range of experiments, we thoroughly study the effect of optimization methods discussed in Section 4.3. Finally, we experiment the semantic retrieval methods on the patent domain and report the results.

5.1 Preliminary Experiments

As mentioned before, we check the sanity of the proposed semantic representations and similarity methods on SemEval 2014 Multilingual Semantic Textual Similarity - Task 10 [2], the English subtask. The goal of this task is to measure the semantic similarity of two sentences. Participating systems are compared by their mean Pearson correlation (Section 2.2) between the system output and a human-annotated gold standard. The dataset consists of sentence pairs from different resources: news headlines of Europe Media Monitor (EMM), sense definition pairs of WordNet [16], image descriptions, discussion forums and finally tweet comments. The gold standard is achieved by annotating each sentence pair with a score between 5 to 0 (5 completely equivalent, 0 completely dissimilar).

In order to test the methods, we used the English Wikipedia text corpus to train semantic representations based on Word2Vec and Random Indexing, each with 200 and 600 dimensions. We cleaned the corpus by removing HTML tags and non-alphabetic characters. We trained our Word2Vec word representation using Word2Vec toolkit[1] by applying CBOW architecture with context windows of 5 words and sub-sampling (down-sampling frequent words) at $t = 1e^{-5}$.

[1] `https://code.google.com/p/word2vec/`

Table 5.1: Mean Pearson correlation of SemEval 2014 Task 10 [2] using Word2Vec (W2V) [37] and and Random Indexing (RI) [53] word representations

Representation	Dim	SimAgg	SimGreedy
RI	600	0.691	0.706
RI	200	0.678	0.702
W2V	600	0.685	**0.715**
W2V	200	0.654	0.715

The Random Indexing word representations were trained using Semantic Vectors package[2]. We used the default parameter settings of the package which considers whole the document as context window. In both Word2Vec and Random Indexing we considered the words with frequency less than five as noise and filtered them out.

Table 5.1 shows the Mean Pearson correlations between the similarity methods and the gold standard. The most impressive result is that SimGreedy with Word2Vec achieved an average correlation of 0.71 as the best overall performance. This represents rank 11th out of the 38 submitted runs. However, all 10 runs above use a knowledge base and/or NLP which would not generalize to other domains or languages. Between similarity methods, SimGreedy shows better performance than SimAgg. The results also show that the similarity method has more effect than the number of dimensions or word representation. In what follows, we shall verify this observation for our domains of interest.

5.2 Social Image Retrieval

The evaluation is conducted using Flickr data, in particular in the framework of the MediaEval Retrieving Diverse Social Images Task 2013/2014 [21, 22]. The task addresses result relevance and diversification in social image retrieval. For a larger test collection, we merged the datasets of 2013 (Div400) [22] and 2014 (Div150Cred) [21] and denoted it as MediaEval. The resulting dataset consists of approximately 110k photos of 600 world landmark locations (e.g., museums, monuments, churches). For each of these 600 locations, the provided information include a ranked list of photos, a representative text, Flickr's metadata, a Wikipedia article of the location and for 2014 edition only user tagging credibility estimation.

For semantic text similarity, we focused on the relevance of the representative text of the photos namely the title, description, and tags. We cleaned the text by removing HTML tags and numbers. Then, we decompounded the tags by applying a greedy-based approach (see Algorithm B.1) using the dictionary obtained from the whole corpus. Decompounding was necessary even for English because in the data, the tags with many terms are concatenated with no space. As a sample result, the tag '*cityofwestminster*' is decompounded to '*city of west minster*'.

[2] `https://code.google.com/p/semanticvectors/`

Table 5.2: P@20 of MediaEval Retrieving Diverse Social Images Task 2013/2014 [21, 22]. Models are trained on Wikipedia. (Q,D) and (D,Q) are SimGreedy(Q,D) and SimGreedy(D,Q). Digits in brackets indicate the ID of each run

Repr.	Dim	SimAgg	SimGreedy	(Q,D)	(D,Q)
RI	200	0.774 (1)	†0.788 (5)	0.704	0.766
RI	600	0.766 (2)	†0.787 (6)	0.703	0.769
W2V	200	0.778 (3)	†**0.795** (7)	0.690	0.760
W2V	600	0.779 (4)	†0.793 (8)	0.693	0.757

As an evaluation metric, the MediaEval Retrieving Diverse Social Images task used P@20, and we therefore use this measure in our experiments as well. Precision at 20 is the proportion of relevant documents (images) in the first 20 images retrieved by a system. A standard Solr index was used as the baseline. Solr is a state-of-the-art search engine using the Lucene search library, commonly used in many commercial systems such as eBay and Instagram. The Solr search engine produces a P@20 of 0.76 on the test collection. For all experiments, statistical significance against the baseline is calculated using paired randomization test (Section 2.2). In all tables, † denotes statistical significant difference at p 0.05 or lower.

In the following, first we report the result of semantic text similarity experiments and then we study the effect of the optimization techniques.

Semantic Similarity

The results of evaluating the combination of methods and word representations are shown in Table 5.2. In addition to the similarity methods defined in Section 4.2, we also considered the asymmetric version of SimGreedy (SimGreedy(Q,D) and SimGreedy(D,Q)) to explore the directional nature of semantic similarity. Namely, whether semantic similarity is different from the perspective of the query or from that of the document. This in itself should not be surprising: such unbalances are quite common in nature: i.e. a member of a small group might find a member of a larger group closer than the other way around.

The experimental results show that SimGreedy outperforms SimAgg regardless of the training method while all runs with SimAgg have no significant difference from the baseline. Observing the asymetric versions of SimGreedy, SimGreedy(D,Q) shows better results than SimGreedy(Q,D) since documents are generally longer and more descriptive than queries. However SimGreedy outperforms both SimGreedy(Q,D) and SimGreedy(D,Q). We hypothesize that the (Q,D) version, which performs very poorly on its own, acts as a length normalization factor for the (D,Q) version, therefore contributing to the improved result.

For more insight on the differences between the runs, we additionally compared all our combinations by calculating their pairwise Pearson rank correlation (Figure 5.1). This complements the evaluation based on P@20, because it informs us on whether different methods tend to rank pairs of documents in the same way or in different ways even within the top 20. As it is shown

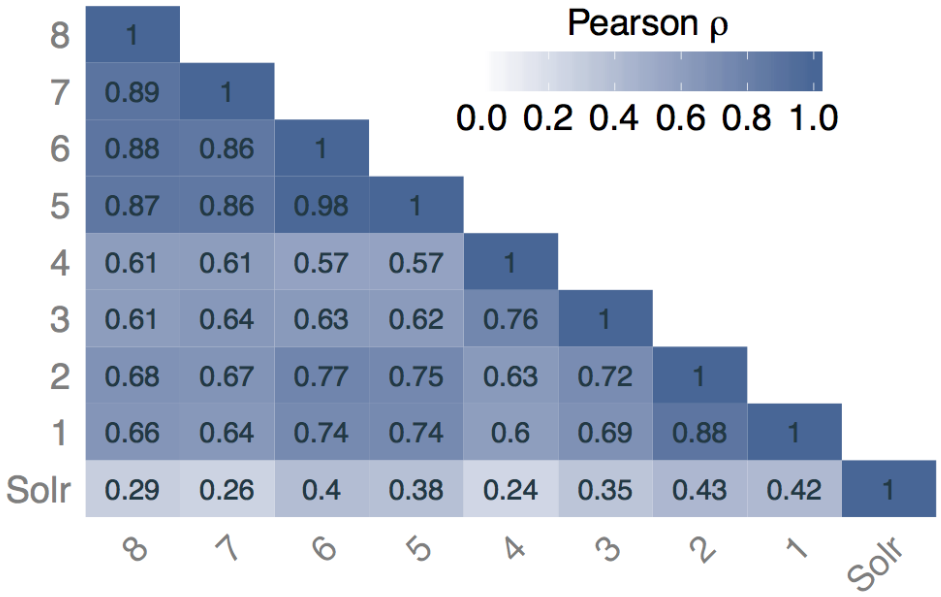

Figure 5.1: Pearson Correlations between all 8 combinations of approaches and the Solr baseline. The numbers refer to the ID of the runs in Table 5.2

in the results, the average correlation between runs using SimGreedy is larger than those that using SimAgg. This means that regardless of the training method and the number of dimensions for word representation, using SimGreedy produces more similar results. We also observed very high correlations between the same models with 200 and 600 dimensions which demonstrates that increasing the dimensionality does not affect results.

While Wikipedia provides a general knowledge about different topics, it is interesting to study the effectiveness of using the MediaEval corpus for representing the words instead of an external resource. Therefore, we trained the Word2Vec and Random Indexing models on the MediaEval corpus. As the previous experiments show the ineffectiveness of dimensionality, we trained the models only with 200 dimensions.

The results in Table 5.3 show an overall lower performance. Exceptionally, we observed a significantly better performance when using Random Indexing in combination with SimAgg—as good as the best result achieved by the previous experiment. As the default definition of Random Indexing uses the whole document as the context window, while Word2Vec uses a context window of 5, we also trained Random Indexing with the context window of 5 (RI-Window). As it is shown in Table 5.3, similar to Word2Vec, RI-Window does not improve the performance. It is therefore reasonable to assume that the difference is due to the amount of information considered in creating the vector of each term.

This observation leads to the hypothesis that, as additional information proved useful in the construction of the terms, it should also prove useful in the construction of the queries themselves. Therefore, in the following, we expand the topic names with the first sentence of their corresponding Wikipedia page. As it is shown in Table 5.4, in comparison to previous exper-

Table 5.3: Models are trained on MediaEval corpus

Representation	Dim	SimAgg	SimGreedy
RI	200	†**0.795**	0.776
W2V	200	0.767	0.758
RI-Window	200	0.753	0.759

Table 5.4: Results using query expansion

Corpus	Repres.	Dim	SimAgg	SimGreedy
Wiki	RI	200	0.768	†0.794
Wiki	W2V	200	0.756	†0.786
MediaEval	RI	200	†**0.795**	†0.788
MediaEval	W2V	200	†0.780	†0.792

Table 5.5: MediaEval2014 Results using query expansion

Corpus	Repres.	Dim	SimAgg	SimGreedy
Wiki	RI	200	0.795	0.833
Wiki	W2V	200	0.788	0.813
MediaEval	RI	200	0.840	0.820
MediaEval	W2V	200	0.831	**0.848**

iments, the performance does not change significantly except the results related to SimGreedy in combination to MediaEval corpus. We conclude that using SimGreedy with query expansion provides a stable method with good performance regardless of the word representation method or training corpus.

In order to compare the results with the participating systems in the task, we repeated the experiment on test dataset 2014. As it is shown in Table 5.5, using SimGreedy and Word2Vec trained on the MediaEval corpus, we achieve the state-of-the-art result of 0.848 for P@20 between 41 runs including even the ones which used image features but not external resources [18, 43].

Optimization

Although SimGreedy shows stable and better performance in comparison to SimAgg, based on the time complexity discussion in Section 4.2, it has a much longer execution time. We observed that SimGreedy is approximately 40 times slower than SimAgg so that SimGreedy generally has

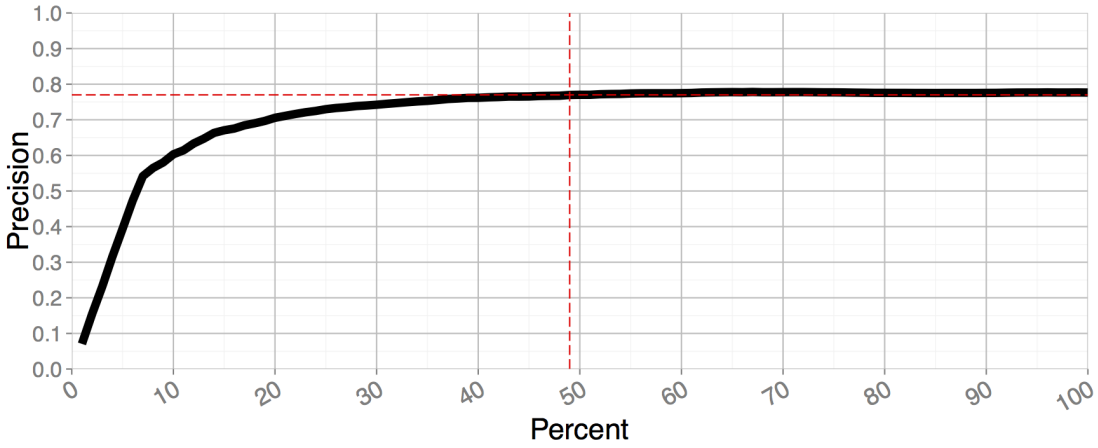

Figure 5.2: Average performance of the Hybrid Search Engines approach with best value at around 49%

the calculation time of about 110 to 130 minutes while it takes about 3 minutes for SimAgg for all the topics. We therefore apply the two techniques discussed in Section 4.3 in order to achieve better execution time.

Hybrid Search Engines

As discussed in Section 4.3, this approach consists of two phases with different retrieval algorithms. In the first phase, we applied the SimAgg method to obtain a first ranking of the results. As the second phase, we used n percent of the top documents ranked by the first phase and re-ranked them using SimGreedy. For each combination of different parameters (training data, dimensionality, training method), we found an extremely similar behavior for all the values of n from the 1 to 100, which we summarize in Figure 5.2. In order to find the best value for n as the cutting point, we identified the highest precision value that is not significantly different from the best one (i.e. when n is 100 percent). This corresponds to $n = 49$. Giving the second phase (SimGreedy) is about 40 times slower than the first (SimAgg), using this approach improves the execution time to almost two times (48 percent) while the performance remains the same.

Approximate Nearest Neighborhood Index

In this approach, we first created an Approximate Nearest Neighbor Index (ANN-Index) data structure—denoted as semantic index—for each document, using the scikit-learn library[3]. Because of the large dimension of the vectors (> 30), we chose Ball-Tree data structure (Section 2.4) with leaf size of 30. We then used the semantic indexes to calculate the SimGreedy algorithm as it is described in Section 4.3. We run the experiment using vector representations

[3] `http://scikit-learn.org/stable/`

Table 5.6: Execution time in hours for the standard, Hybrid, and Approximate Nearest Neighbor Index (ANN-Index) approaches of SimGreedy. Models are trained on Wikipedia corpus with 200 dimensions. There is no statistical significant difference between the achieved results of the evaluation metric (P@20).

Repres.	Algorithm	Indexing Time	I/O	Query Time	Overall	P@20
W2V	SimGreedy	-	0:16	1:50	2:06	0.795
	SimGreedy + Hybrid	-		0:50	1:06	0.772
	SimGreedy + ANN-Index	0:28		0:17	1:01	0.782
RI	SimGreedy	-	0:14	2:07	2:24	0.788
	SimGreedy + Hybrid	-		1:00	1:14	0.770
	SimGreedy + ANN-Index	0:21		0:19	0:54	0.782

with 200 dimensions trained on Wikipedia corpus using both Word2Vec and Random Indexing methods.

Table 5.6 shows the results compared with the original SimGreedy as well as the Hybrid algorithm. The I/O time consists of reading the documents, fetching the corresponding vector representations of the words and writing the final results which is common between all the approaches. Although the ANN-Index approach has the overhead of indexing time, its query time is significantly less than the original SimGreedy and also Hybrid approach. We therefore see an improvement of approximately two times in the overall execution time in comparison to the original SimGreedy method. In spite of the time optimization, there is no significant difference between the evaluation results of the methods.

It should also be noted that since in the MediaEval task, each topic has its own set of documents, the semantic index of each document is used only one time by its topic. Considering this fact, in particular, the ANN-Index method will benefit strongly from the index, as this is created once and used multiple times.

5.3 Patent Retrieval

The evaluation on patent retrieval domain is conducted using CLEF-IP Claims to Passage Task 2013 [47] test collection. The task models the *Prior-Art* search (Section 3.3) by evaluating the ability of an IR system in retrieving relevant documents and text passages regarding to a set of claims. The data set consists of patent documents published by European Patent Office (EPO) and World Intellectual Property Organization (WIPO). The patents are in English while a portion of the EPO documents are also provided in German and French. The collection contains almost 1.5 million patents published before 2002, stored into approximately 3.5 million XML files. The simplified structure of a sample XML patent document is shown below:

```
<patent-document>
  <bibliographic-data> ... </bibliographic-data>
```

```
  <abstract> ... </abstract>
  <description> ... </description>
  <claims> ... </claims>
</patent-document>
```

The focus of textual retrieval is on the abstract, description, and claims parts, while the bibliographic section mostly contains administrative data.

The task provides 150 training and 149 test topics. Each set consists of topics in English, German, and French languages (each language 50 topics). The task evaluates the results in two ways: *document-level* and *passage-level*. In document-level, the system should return documents which were considered relevant regarding to a set of claims, while passage-level evaluates the most relevant passages within these documents. The official evaluation metric for document-level is Patent Retrieval Evaluation Score (PRES) [31] at 20 and 100 cutoffs. PRES rewards systems that return relevant documents earlier in the retrieval list. For passage-level, an adoption of Mean Average Precision (MAP), annotated as MAP(D), is introduced by the organizers. The detail of the computation of the metric is described in Piroi et al [48].

We indexed the patent collection using Solr by assuming each passage as a Solr document. We removed HTML tags and tokenized the words using the StandardTokenizer. We then trained the Word2Vec model on the text corpus with 400 dimensions. Similar to the previous experiments, we used CBOW architecture with context windows of 5 words, subsampling at $t = 1e^{-5}$, and filtering the words with frequency less than five.

Since the patent documents are much longer than the previous experiments, fetching the vector representations as well as computing document-to-document similarity for SimGreedy method is a much more expensive and time-consuming process. By running the process for more than one month with 10 threads, we eventually obtained the results of 8 topics (topic number 1, 2, 5, 6, 10, 11, 12, and 13) out of overall 50 English topics. We therefore measured SimAgg for these 8 topics and defined baseline as the result of the Solr search engine. The evaluation results on document- and passage-level are shown in Table 5.7

The results show an overall better performance of SimAgg especially in passage-level task. As the results per topics are depicted in Figure 5.3, SimAgg outperforms the other methods in all the topics of passage-level (right) and also has a higher or similar result to Solr in document-level (left) task.

Despite the better performance of semantic-based methods by using SimAgg, SimGreedy shows much weaker results. We investigated the reason by focusing on the second topic in which SimGreedy has a bigger difference in comparison to the other two approaches. Table 5.8 shows the cleaned text of the second topic (top) as well as three high-ranked passages in SimGreedy results (middle). In all the examples, although the similarity between the topic and the passages (SimGreedy(Q,D)) is low, the value of the other direction (SimGreedy(D,Q)) is close to the 1 (maximum similarity value). Therefore, the final aggregation between these two values leads to a high similarity in comparison to the other passages. The common characteristic between the mentioned examples is that they are generally the title or keywords sections of the patents and therefore they are short and less informative. Also, these sections usually use general or buzz words of a domain which are repeated many times in the topic. Therefore, as SimGreedy finds all these terms of the passage in the topic, considers the passage as highly related. In order

Table 5.7: Evaluation results of 8/50 topics of CLEF-IP Claims to Passage Task 2013 [47]. The models are created by applying Word2Vec on CLEF-IP text corpus with 400 dimensions. Evaluation measures for document- and passage- level are PRES [31] and MAP(D) [48] respectively.

Topics	Document Level (PRES)			Passage Level (MAP(D))		
	Solr	SimAgg	SimGreedy	Solr	SimAgg	SimGreedy
PSG-1	0.499	0.481	0.000	0.062	0.479	0.000
PSG-2	0.183	0.619	0.000	0.333	0.347	0.000
PSG-5	0.910	0.989	0.496	0.034	0.271	0.063
PSG-6	0.951	0.985	0.973	0.088	0.267	0.189
PSG-10	0.000	0.000	0.000	0.000	0.000	0.000
PSG-11	0.478	0.489	0.250	0.104	0.162	0.151
PSG-12	0.281	0.000	0.000	0.017	0.000	0.000
PSG-13	0.652	0.656	0.333	0.140	0.219	0.208
Avg	0.494	**0.527**	0.257	0.097	**0.218**	0.076

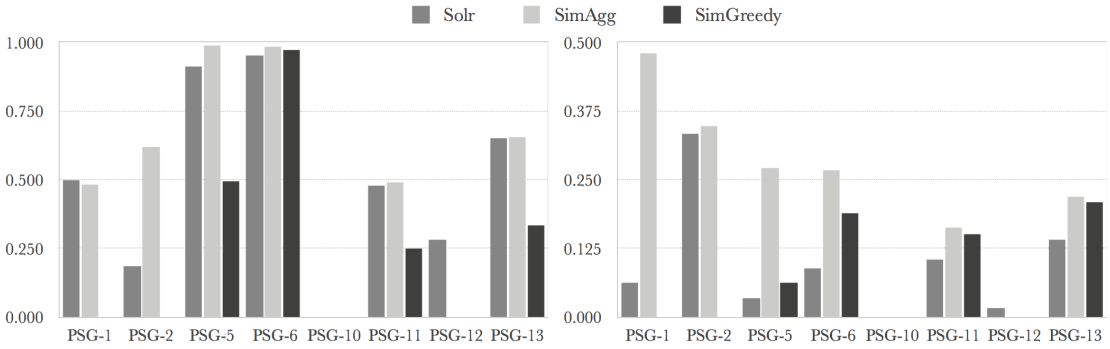

Figure 5.3: Comparison of evaluation results of Solr, SimAgg, and SimGreedy per topic for document-level (left) and passage-level (right)

to compare the algorithms, we also show the highest ranked passage of the SimAgg method in Table 5.8-bottom. The result of SimAgg seems much more related to the topic in comparison to SimGreedy while not in the set of passages and documents considered related by the task. The reason that the passage is not in the related documents could be due to the fact that in the CLEP-IP task, the provided relevant documents only cover a portion of all similar documents. In other words, there could be an apparently related document which is not suggested as relevant by the task.

We also repeated the experiment with SimGreedy(Q,D) and SimGreedy(D,Q) and observe much worse results. We therefore hypothesize that although SimGreedy acts as a normalization factor on SimGreedy(Q,D) and SimGreedy(D,Q), the aggregation still skews the results when one text is shorter and less informative than the other one.

Table 5.8: Cleaned text of the second topic (top), three examples of high ranked passages by the SimGreedy method (middle), as well as the top ranked passage of the SimAgg method (bottom). (Q,D) and (D,Q) stand for SimGreedy(Q,D) and SimGreedy(D,Q) respectively

Topic 2				
method regenerating oxidation catalyser exhaust gases internal combustion engine which method exhaust gases internal combustion engine directed through oxidation catalyser during normal operation engine which method oxidation catalyser regenerated desired moment bringing reducing gas contact catalyser characterized during regeneration amount engine exhaust gas flowing through catalyser reduced stopping engine method according claim characterized heat accumulated catalyser during normal use used regeneration method according claim characterized temperature oxidation catalyser measured 12 temperature data read control apparatus 11 under control which importing reducing gas exhaust gases terminated temperature oxidation catalyser falls certain set value method according claim characterized reducing gas produced separate arrangement producing reducing gas				
SimGreedy				
XPath in Doc.	Text	(Q,D)	(D,Q)	SimGreedy
/description/p[1]	method apparatus control combustion engine combustion engine	0.430	0.999	0.715
/description/p[2]	control method apparatus internal combustion	0.421	1.000	0.710
/description/p[1]	exhaust gas control device internal combustion	0.441	0.968	0.705
SimAgg				
fig 29 shows another embodiment invention recently exhaust regulation well fuel consumption regulation strengthened meet these regulations important achieve maximum limit cleaning performance catalyser 11 purpose therefore temperature sensor 61 mounted upstream side catalyser 11 engine turbocharger controlled gain maximum cleaning efficiency catalyser basis temperature exhaust gases flowing catalyser fig 30 shows relationship between inlet temperature cleaning efficiency catalyser 11 generally catalyser used optimum temperature range cleaning exhaust gases cleaning efficiency will lower temperature high lower optimum temperature configuration present embodiment necessary effectively transmit heat energy exhaust gases catalyser case small quantity exhaust gases low exhaust gas temperature when exhaust gases pass through turbine 32 heat exhaust gases deprived turbine housing impeller temperature therefore will drop before exhaust gases arrive catalyser bypass control valve 34 opened introduce exhaust gases catalyser passing through turbine 32 prevent wasteful heat dissipation turbine thereby enabling rise of catalyser temperature other hand exhaust gas temperature rises increase engine load possibility temperature catalyser 11 exceeds optimum temperature range bypass control valve closed exhaust gases pass through turbine 32 enabling absorption heat exhaust gases turbine thereby control catalyser temperature possible control amount exhaust gases which flow turbine 32 catalyser 11 controlling bypass valve previously stated also keep catalyser temperature within range high cleaning efficiency				

CHAPTER 6

Conclusion and Future Work

Regarding to the achieved results, in this chapter, we summarize the observations and conclude the study. We also explain some possible future research directions as well as ideas for further development of the studied methods.

6.1 Conclusion

We explored the use of semantic similarity in social as well as patent domains by applying Word2Vec and Random Indexing together with two similarity methods, denoted as SimGreedy and SimAgg. We first checked the sanity of the methods by the SemEval2014 Task 10 and then ran a wide range of experiments on the MediaEval Retrieving Diverse Social Images Task 2013/2014. Beside achieving state-of-the-art results on both data sets, we observe that the similarity method has more effect on the results rather than the number of dimensions or word representation training methods. Specifically, SimGreedy shows stable and also better performance on both the data sets.

We then focused on two optimization techniques: Hybrid Search Engines and Approximate Nearest Neighbor Index (ANN-Index). Both methods reduced in half the processing time of the SimGreedy method while keeping precision within the boundary of statistically insignificant difference. Although the mentioned techniques similarly optimize the processing time, they show different characteristics in practice. While the Hybrid approach needs pre-knowledge on the performance of the other search methods for setting the parameters, the ANN-Index method can easily be applied on new domains with no need of parameter tuning. In addition, in the ANN-Index approach, despite the overhead time of creating semantic-based data structures, the query time is significantly better which is a great benefit in real-time use-cases.

As the final experiment, we studied the effect of semantic methods on CLEF-IP Claims to Passage Task 2013. We obtained valuable insight about the limitations and effectiveness of the methods in this domain:

- SimAgg outperforms a best practice on a content-based search engine (Solr), specially on passage-level retrieval.

- Due to the strict aggregation method of the SimGreedy, it get skewed when one passage is short and consists of very common words of the domain.

We observe a significant difference between the performance of the semantic similarity methods in different domains. The studied domains are considerably different regarding to the content of the information provided by the terms in each document. In the social image retrieval domain, as each term provides notable amount of information, the SimGreedy method shows superior performance by finding relevant terms of the documents. In contrary, documents in patent domain consist of many general terms that are only meaningful when combined together. For instance, instead of 'camera', we would find statements like 'image shooting apparatus' [40]. These terms cause bias in SimGreedy, as it increases the similarity value when they are matched between the documents. In contrast to SimGreedy, SimAgg aggregates all the terms and therefore has a stronger normalizing effect. Although in social image retrieval domain, this approach does not show any particular benefit, it outperforms the baseline in patent retrieval domain. We hypothesize that this advantage is due to the conceptual representation of the documents rather than only use of term-frequencies, while it is also not biased by individual terms. We therefore conclude that while, similar to general methods in information retrieval, semantic-based similarity methods also strongly rely on the information content provided by the terms, selecting the appropriate similarity method can enhance the retrieval results.

6.2 Future Work

In future work, regarding to the social media domain, we want to integrate the semantics of the facets (e.g. text, image, etc.) based on its representing context into the payload feature of Lucene framework. The payload feature gives control of score-boosting on term level and is one of the latest additions to Lucene. Specifically, payloads were introduced to allow boosting individual terms based on additional meta information about these terms to further diversify the significance of these terms in the scoring process of a search engine. We will apply the payload feature semantically as part of the concept index. The concept index stores unique concept labels and associates them with documents and facets. For text facets, the payload will represent the probability of a term being conceptually similar to its context (e.g., document, window of the terms, etc.). For image facets, the payload will represent the probability of a visual concept that has been learned from an image (e.g. from a visual classifier).

As mentioned before, despite the effectiveness of SimGreedy in the social media domain, it shows very weak performance in the patent domain. Based on the observations in this work, one research path is the study of variant length normalization techniques in order to adapt it more to the information retrieval field. Another issue of SimGreedy is that, for each term in the source document, it only finds the highest similar term in the destination and ignores the others with less or an equal similarity value. Another research path is therefore the study of alternative similarity measures that match terms with groups of related terms.

Exploring new optimization techniques are key for semantic indexing to address the large-scale content resources. Studying heuristic parameter tuning methods as well as new approximation data structures can facilitate the retrieval process, especially on the patent domain. As a bottleneck for the introduced optimization methods, they both focus on optimization of the similarity calculation of query to one document. Inspiring from Vector Space Model, we work on new optimized semantic-based techniques which measure the position of a query regarding to all the documents and returns similarity values of all of them in one round.

In addition to the studied domains, exploring the use of semantic similarity in the area of Expert Retrieval is also highly interesting. The objective of this domain is retrieving the experts represented by a set of documents. Our initial experiments in this paper [50] shows that as the terms in each expertise overlap the same concepts, the search for experts can be enhanced by using the conceptual meanings of the representing terms.

APPENDIX A

Figures and Diagrams

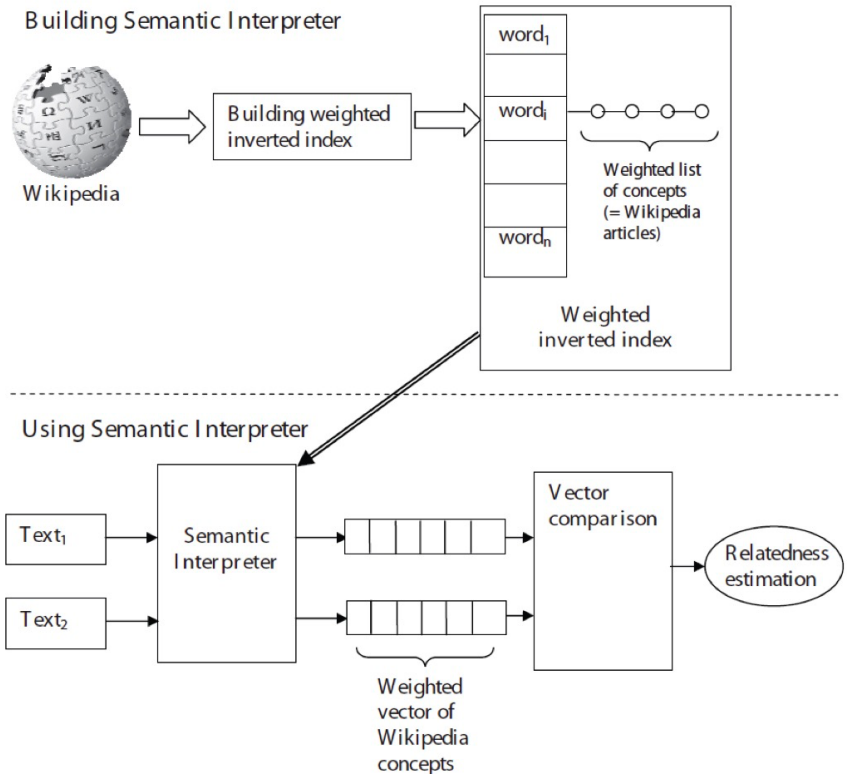

Figure A.1: Process of applying ESA-based semantic similarity using Wikipedia articles. The picture is obtained from Gabrilovich et al. [17]

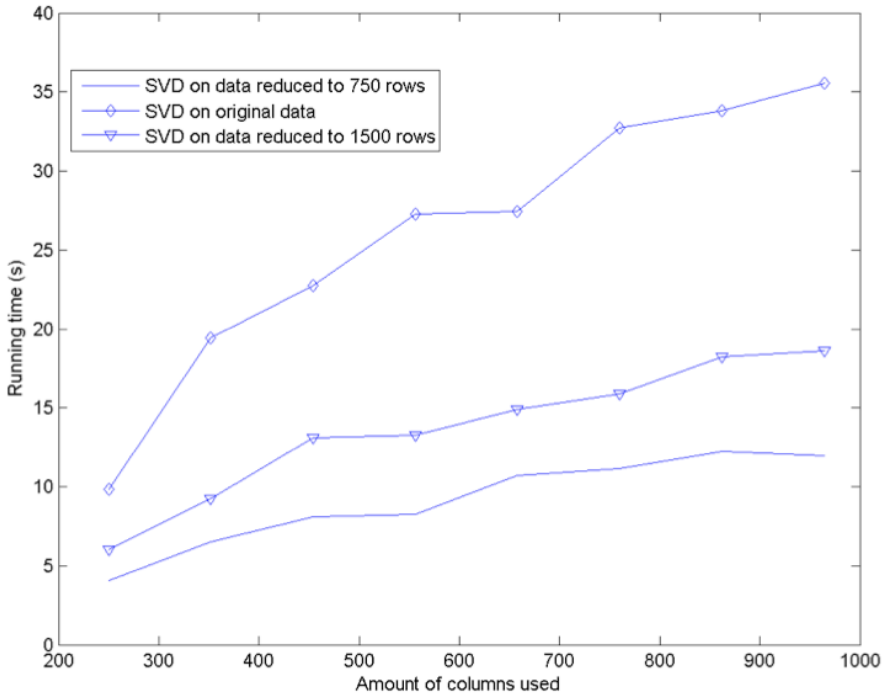

Figure A.2: Execution time of three semantic representation approaches: LSA, RI, and LSA with reduced RI matrix. The picture is obtained from Sellberg et al. [54]

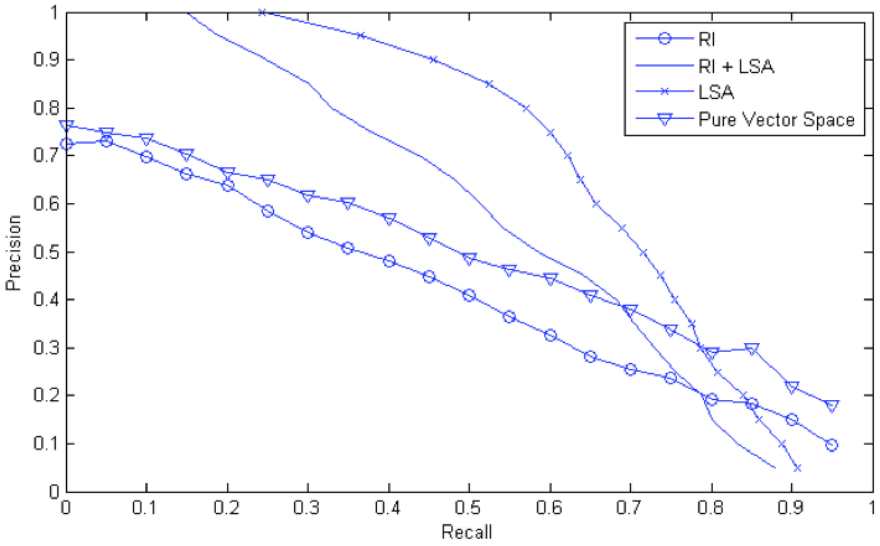

Figure A.3: Performance of three semantic representation approaches: LSA, RI, LSA with reduced RI matrix, as well as standard Vector Space Model. The picture is obtained from Sellberg et al. [54]

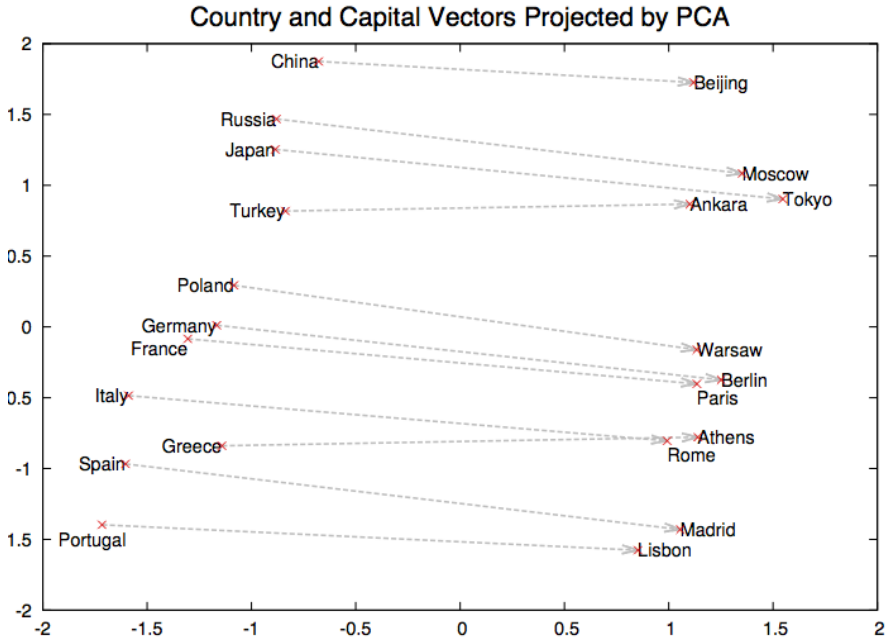

Figure A.4: Vector representation of countries and capitals using the Word2Vec model projected to two-dimensional space by PCA. The picture is obtained from Mikolov et al. [39]

United States Patent [19] [11] **4,131,276**
Judkins [45] **Dec. 26, 1978**

[54] **NON-INJURIOUS AMUSEMENT BALL AND METHOD OF MAKING SAME**

[76] Inventor: **Ren Judkins**, 2260 Belaire Dr., Salt Lake City, Utah 84109

[21] Appl. No.: **836,630**

[22] Filed: **Sep. 26, 1977**

[51] **Int. Cl.**[2] **A63B 43/02**; A63B 45/00; A63B 37/14

[52] **U.S. Cl.** ..... **273/58 C**; 273/58 K; 273/DIG. 20; 273/DIG. 8

[58] **Field of Search** ..... 273/199 R, 199 A, 58 K, 273/200 R, DIG. 20, DIG. 8; 15/244 C; 428/4, 5

[56] **References Cited**

U.S. PATENT DOCUMENTS

646,350	3/1900	Breinl	273/58 A
1,548,531	8/1925	Knight	273/199 R UX
1,611,076	12/1926	Rittner	273/200 R
2,450,474	10/1948	Grobner	273/199 R UX
2,789,305	4/1957	Weil	15/244 R
3,069,170	12/1962	Dillon	273/199 R

FOREIGN PATENT DOCUMENTS

483502	5/1952	Canada	273/58 K
963098	5/1957	Fed. Rep. of Germany	15/244 R
16231 of	1910	United Kingdom	273/199 A

Primary Examiner—George J. Marlo
Attorney, Agent, or Firm—H. Ross Workman; J. Winslow Young

[57] **ABSTRACT**

A lightweight, non-injurious amusement ball designed for use in a restricted area or where it is desirable to prevent injury to the surroundings and participants from impact with the ball. The ball is fabricated from resilient, cellular plastic foam strips of rectangular cross section compressed in a central core and extending radially into an essentially spherical periphery. The periphery is impact absorbent and wind resistant for non-injurious activity and limited flight. A core is used to tie and compress the strips in the central core, providing increased density for stable trajectory.

7 Claims, 3 Drawing Figures

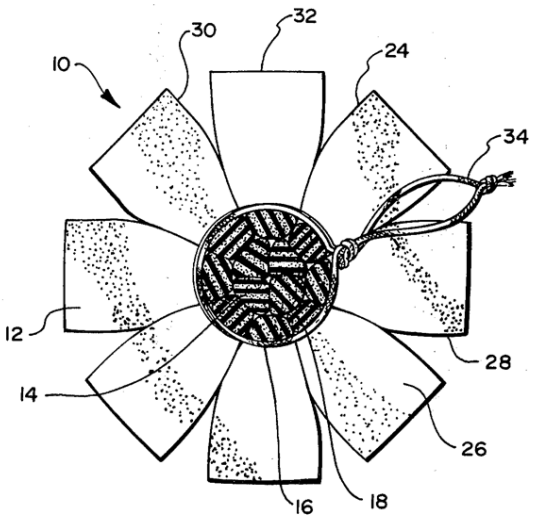

Figure A.5: Example of the first page of US patent application 836630 obtained from Google Patent data source

APPENDIX B

Programming Codes

input : A dictionary Dic of size d, a compound term $term$ of length l, minimum and maximum size of the terms $minSize$, $maxSize$
output: A list of decomposed terms $tokens$

```
1  i ← 0;
2  while i ≤ l − minSize do
3      longestMatchToken ← null;
4      for j ← minSize to maxSize do
5          if i + j > l then break;
6          if SUBSTRING(term ,i ,j) contains in Dic then
7              we only consider the longest match;
8              if longestMatchToken is not null then
9                  if LENGTH(longestMatchToken) < j then
10                     longestMatchToken ← SUBSTRING(term ,i, j);
11             else
12                 longestMatchToken ← SUBSTRING(term ,i, j);
13     if longestMatchToken is not null then
14         if LENGTH(longestMatchToken) ! = l then
15             tokens.ADD(longestMatchToken);
16         i ← i+ LENGTH(longestMatchToken);
17     else
18         i ← i + 1;
19 return longestMatchToken;
```

Algorithm B.1: Greedy-based decompounding algorithm

Bibliography

[1] Joao Magalh aes and Stefan Rüger. Information-theoretic semantic multimedia indexing. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 619–626. ACM, 2007.

[2] Eneko Agirrea, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirrea, Weiwei Guof, Rada Mihalceab, German Rigaua, and Janyce Wiebeg. Semeval-2014 task 10: Multilingual semantic textual similarity. *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*, 2014.

[3] Kristine H Atkinson. Toward a more rational patent search paradigm. In *Proceedings of the 1st ACM workshop on Patent information retrieval*, pages 37–40. ACM, 2008.

[4] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247, 2014.

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, pages 509–517, 1975.

[6] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics, 2012.

[7] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112–116. Association for Computational Linguistics, 1992.

[8] Stéphane Clinchant, Julien Ah-Pine, and Gabriela Csurka. Semantic combination of textual and visual information in multimedia retrieval. In *Proc. of the 1st ACM International Conference on Multimedia Retrieval*, 2011.

[9] Tatiana AS Coelho, Pável Pereira Calado, Lamarque Vieira Souza, Berthier Ribeiro-Neto, and Richard Muntz. Image retrieval using multiple evidence ranking. *Knowledge and Data Engineering, IEEE Transactions on*, 16(4):408–417, 2004.

[10] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cedric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision (at ECCV)*, 2004.

[11] Van Dang, Michael Bendersky, and W.Bruce Croft. Two-stage learning to rank for information retrieval. In *Proceeding of European Conference on Information Retrieval (ECIR)*, 2013.

[12] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science (JASIS)*, 1990.

[13] Adrien Depeursinge and Henning Müller. Fusion techniques for combining textual and visual information retrieval. In Henning Müller, Paul Clough, Thomas Deselaers, and Barbara Caputo, editors, *ImageCLEF*, volume 32, pages 95–114. Springer Berlin Heidelberg, 2010.

[14] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics, 2004.

[15] Maria Eskevich, Gareth J.F. Jones, Robin Aly, and et al. Multimedia information seeking through search and hyperlinking. In *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR)*, 2013.

[16] Christiane Fellbaum. ed. wordnet: an electronic lexical database. *MIT Press, Cambridge MA*, 1:998, 1998.

[17] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 1606–1611, 2007.

[18] Alexandru Lucian Ginsca, Adrian Popescu, and Navid Rekabsaz. Cea list's participation at the mediaeval 2014 retrieving diverse social images task. In *Working Notes Proceedings of the MediaEval Workshop, Barcelona, Catalunya, Spain*, 2014.

[19] GK Gupta. *Introduction to data mining with case studies*. PHI Learning Pvt. Ltd., 2014.

[20] Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *Joint Conference on Lexical and Computational Semantics*, 2013.

[21] Bogdan Ionescu, Adrian Popescu, Mihai Lupu, Alexandru Lucian Gînsca, Bogdan Boteanu, and Henning Müller. Div150cred: A social image retrieval result diversification with user tagging credibility dataset. *ACM Multimedia Systems Conference (MMSys)*, 2015.

[22] Bogdan Ionescu, Anca-Livia Radu, María Menéndez, Henning Müller, Adrian Popescu, and Babak Loni. Div400: a social image retrieval result diversification dataset. In *Proceedings of ACM Multimedia Systems 2015 Conference (MMSys)*, 2014.

[23] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[24] Abhay Kashyap, Lushan Han, Roberto Yus, Jennifer Sleeman, Taneeya Satyapanich, Sunil Gandhi, and Tim Finin. Meerkat mafia: Multilingual and cross-level semantic textual similarity systems. In *Proceedings of the International Workshop on Semantic Evaluation*, pages 416–423. Association for Computational Linguistics, 2014.

[25] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[26] Chao Liu and Yi-Min Wang. On the connections between explicit semantic analysis and latent semantic analysis. In *Proc.eeding of Conference on Information and Knowledge Management (CIKM)*, New York, NY, USA, 2012.

[27] Ningning Liu, Emmanuel Dellandréa, Liming Chen, Chao Zhu, Yu Zhang, Charles-Edmond Bichot, Stéphane Bres, and Bruno Tellez. Multimodal recognition of visual concepts using histograms of textual concepts and selective weighted late fusion scheme. *Computer Vision and Image Understanding (CVIU)*, 2013.

[28] Mihai Lupu. On the usability of random indexing in patent retrieval. In *Graph-Based Representation and Reasoning*, pages 202–216. Springer, 2014.

[29] Mihai Lupu and Allan Hanbury. Patent retrieval. *Foundations and Trends in Information Retrieval*, 7(1):1–97, 2013.

[30] João Magalhães and Stefan Rüger. Using manual and automated annotations to search images by semantic similarity. *Multimedia Tools Applications*, 2012.

[31] Walid Magdy and Gareth JF Jones. Pres: a score metric for evaluating recall-oriented information retrieval applications. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 611–618. ACM, 2010.

[32] Walid Magdy and Gareth JF Jones. A study on query expansion methods for patent retrieval. In *Proceedings of the 4th workshop on Patent information retrieval*, pages 19–24. ACM, 2011.

[33] Walid Magdy, Johannes Leveling, and Gareth JF Jones. Exploring structured documents and query formulation techniques for patent retrieval. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, pages 410–417. Springer, 2010.

[34] Parvaz Mahdabi, Linda Andersson, Mostafa Keikha, and Fabio Crestani. Automatic refinement of patent queries using concept importance predictors. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 505–514. ACM, 2012.

[35] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[36] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2006.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[38] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[39] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[40] Hidetsugu Nanba, Saori Mayumi, and Toshiyuki Takezawa. Automatic construction of a bilingual thesaurus using citation analysis. In *Proceedings of the 4th workshop on Patent information retrieval*, PaIR '11, pages 25–30, New York, NY, USA, 2011. ACM.

[41] University of Glagsow website. Medline corpus description. `http://www.dcs.gla.ac.uk/idom/ir`. Accessed: 2015-02-11.

[42] Stephen Malvern Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[43] João RM Palotti, Navid Rekabsaz, Mihai Lupu, and Allan Hanbury. Tuw@ retrieving diverse social images task 2014. In *Working Notes Proceedings of the MediaEval Workshop, Barcelona, Catalunya, Spain*, 2014.

[44] Monica Lestari Paramita and Michael Grubinger. Photographic image retrieval. In Henning Müller, Paul Clough, Thomas Deselaers, and Barbara Caputo, editors, *ImageCLEF: Experimental Evaluation in Visual Information Retrieval*, pages 141–162. Springer, 2010.

[45] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[46] Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *ACL (1)*, pages 1341–1351, 2013.

[47] Florina Piroi, Mihai Lupu, and Allan Hanbury. Overview of clef-ip 2013 lab. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 232–249. Springer, 2013.

[48] Florina Piroi, Mihai Lupu, Allan Hanbury, Walid Magdy, Alan P. Sexton, and Igor Filippov. Clef-ip 2012: Retrieval experiments in the intellectual property domain. In *Proceedings of the Conference and Labs of the Evaluation Forum (CLEF)*, 2012.

[49] C Ramasubramanian and R Ramya. Effective pre-processing activities in text mining using improved porter's stemming algorithm. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(12), 2013.

[50] Navid Rekabsaz and Mihai Lupu. A real-world framework for translator as expert retrieval. In *Information Access Evaluation. Multilinguality, Multimodality, and Interaction*, pages 141–152. Springer, 2014.

[51] V. Rus, M. Lintean, R. Banjade, N. Niraula, and D. Stefanescu. SEMILAR: The Semantic Similarity Toolkit. In *Proceedings of Association for Computational Linguistics (ACL)*, 2013.

[52] Serwah Sabetghadam, Mihai Lupu, Ralf Bierig, and Andreas Rauber:. A combined approach of structured and non-structured ir in multimodal domain. In *Proceeding of ACM International Conference on Multimedia Retrieval (ICMR)*, 2014.

[53] Magnus Sahlgren. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop in the Proceeding of conference of Terminology and Knowledge Engineering (TKE)*, 2005.

[54] Linus Sellberg and Arne Jönsson. Using random indexing to improve singular value decomposition for latent semantic analysis. In *The International Conference on Language Resources and Evaluation (LREC)*, pages 2335–2338, 2008.

[55] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010.

[56] B. Thomee and A. Popescu. Overview of the ImageCLEF 2012 Flickr Photo Annotation and Retrieval Task. In *Proceedings of Cross-Language Evaluation Forum (CLEF)*, 2012.

[57] Kristina Toutanova and Christopher D Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora:*

held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13, pages 63–70. Association for Computational Linguistics, 2000.

[58] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 2010.

[59] Feng Wang, Lanfen Lin, Shuai Yang, and Xiaowei Zhu. A semantic query expansion-based patent retrieval approach. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2013 10th International Conference on*, pages 572–577. IEEE, 2013.

[60] Wikipedia. Null hypothesis. `http://www.nltk.org/api/nltk.stem.html`. Accessed: 2015-02-11.

[61] Wikipedia. Null hypothesis. `http://en.wikipedia.org/wiki/Null_hypothesis`. Accessed: 2015-02-11.

[62] Ludwig Wittgenstein. *Philosophical investigations. Philosophische Untersuchungen.* Macmillan, 1953.