Masaryk University
Faculty of Informatics

Vector Space Representations in Information Retrieval

Master's Thesis

Vít Novotný

Brno, Fall 2017

Masaryk University
Faculty of Informatics

Vector Space Representations in Information Retrieval

Master's Thesis

Vít Novotný

Brno, Fall 2017

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Vít Novotný

Advisor: Doc. RNDr. Petr Sojka, Ph.D.

Acknowledgements

I wish to express my most sincere gratitude to my thesis supervisor, Petr Sojka, for sharing his impeccable expertise, valuable guidance, and for introducing me to the larger scientific community.

I would also like to thank Radim ehek who provided me with countless thought-provoking insights during my work on the thesis.

I would also like to thank my parents for their unceasing encouragement and support during my studies.

Lastly, I wish to express my sincere thanks to all those who have helped me, either directly or indirectly, in cultivating the ideas presented in this thesis.

Abstract

Modern text retrieval systems employ text segmentation during the indexing of documents. I show that, rather than returning the segments to the user, significant improvements are achieved on the semantic text similarity task by combining all segments from a single document into one result with an aggregate similarity score.

Standard text retrieval methods underestimate the semantic similarity between documents that use synonymous terms. Latent semantic indexing tackles the problem by clustering frequently co-occuring terms at the cost of the periodical reindexing of dynamic document collections and the suboptimality of co-occurences as a measure of synonymy. I develop a term similarity model that suffers neither of these flaws.

Keywords

document segmentation, synonymy, question answering, vector space model, text retrieval, information retrieval

Contents

List of Tables

List of Figures

1 Introduction

In 2012, a study conducted by the International Data Corporation (IDC) predicted that by the year 2020, the total amount of digital information resources will have reached the 40 zettabyte mark [1]. According to the rule formulated by Merrill Lynch, 80 to 90 % of these resources will be unstructured. [2] Despite the large amount of unstructured information, users expect an information retrieval (IR) system to be both fast and accurate with respect to their information need. Due to these demands, IR remains an active field of information science more than half a century after Calvin Mooers first formulated its principles [3].

As a researcher in the Math Information Retrieval (MIR) group[1] at the Masaryk University in Brno, Czech Republic, I participated in a state-funded research project in collaboration with a research team from RaRe Technologies[2]. The goal of the project was to develop a text retrieval system that would incorporate the state-of-the-art research in IR while remaining efficient. During the project, I was given an opportunity to attend the 55th Annual Meeting of the Association for Computational Linguistics (ACL) in Vancouver, Canada, where I presented our research [4] along with my thesis supervisor. In reaction to our research and other work presented at ACL 2017, I formulated and carried out two experiments that are the subject of this work.

In my experiments, I used the question answering (QA) datasets developed for the third task of the 10th and 11th International Workshop on Semantic Evaluation (SemEval). I describe the properties of these datasets in chapter 2.

The system we developed splits the stored documents into semantically coherent segments [5]. One of the major advantages of this approach is that short and highly relevant sections are retrieved rather than whole documents that often cover a multitude of topics irrelevant to the user. In practice, text IR systems rarely have this amount of leeway and are required to present the whole documents in the search results. In my first experiment described in chapter 3, I show that even

1. https://mir.fi.muni.cz/
2. https://rare-technologies.com/

such systems can improve their performance on the text similarity task by using segmentation internally.

Today's text IR systems are largely based on the standard vector space bag-of-words model first described by Salton et al. [6]. This model is simple and well-understood, but it makes the assumption that concepts described by different terms are unrelated. In reaction to the work of Charlet; Damnati [7] that was published and presented at ACL 2017, I developed an extension to the standard model that exploits term synonymy. In my second experiment described in chapter 4, I explore several properties of the extended model and show that the extension improves the performance of the standard model on the text similarity task.

Both chapters 3 and 4 are structured as self-contained full papers with their own introduction and conclusion sections. The experimental code in Python for both experiments is publicly available[3]. In the code, I make abundant use of the GNU Parallel [8], Gensim [9], and SciPy [10] software packages.

I conclude this work in chapter 5 by summarizing the results of both experiments, giving a perspective on how the ideas developed in chapters 3 and 4 can be used together in a single system, and suggesting future work.

3. https://github.com/witiko-masters-thesis

2 Datasets

In my experiments, I evaluated the systems on the SemEval-2016 and 2017 task 3 subtask B question answering (QA) datasets. These datasets consist of discussion threads from the Qatar Living[1] internet forum. Given an original question, and a set of ten candidate threads containing both a related question and the initial ten comments in the thread, the task is to rank the candidate threads by their relevance to the original question. [11, 12] The performance of a system is evaluated by its mean average precision (MAP) using relevance judgements.

The SemEval-2016 task 3 subtask B datasets consist of the training dataset (267 original questions, 1,790 threads), the dev dataset (50 original questions, 244 unique threads), and the test dataset (70 original questions, 327 unique threads). The winning SemEval-2016 task 3 subtask B submission was *UH-PRHLT-primary* with a MAP score of 76.7. SemEval-2017 task 3 subtask B uses the same training and dev datasets as SemEval-2016 with the provision that the SemEval-2016 test dataset can be used for training. A new test dataset (88 original questions, 293 unique threads) has also been added. The SemEval-2017 task 3 subtask B winning configuration was *SimBow-primary* with a MAP score of 47.22.

I also used the SemEval-2016 and 2017 task 3 subtask A datasets for statistical analysis. These datasets contain the same data (2,654 questions, 26,540 comments) as the subtask B training datasets, but now the relevance judgements assess how relevant a comment is to a question. For language modelling, the unannotated SemEval-2016 and 2017 task 3 subtask A datasets (189,941 questions, 1,894,456 comments) are available.

2.1 Relevance analysis

In 1991, the American attorney and author Mike Godwin formulated[2] a rule that "as a Usenet discussion grows longer, the probability of a comparison involving Nazis or Hitler approaches one." An immediate

1. http://www.qatarliving.com/forum
2. news:1991Aug18.215029.19421@eff.org

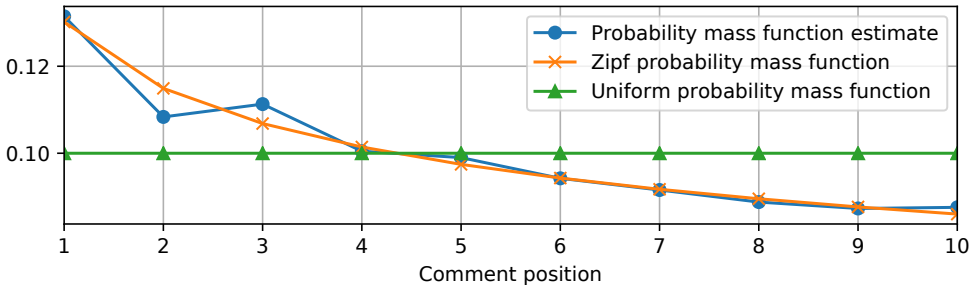

Figure 2.1: Probability mass function (PMF) estimate $\hat{P}(\text{at position } i \mid \text{relevant})$ plotted along the PMF of the Zipf distribution with parameters $n = 10$, and $s = 0.18$. If the position of a comment and its relevance were independent, we would expect the PMF estimate to be uniformly distributed.

corollary would be that as an online discussion grows longer, the probability of a relevant contribution approaches zero. I was curious whether the datasets would confirm these observations.

I used the subtask A relevance judgements to estimate the probability mass function $\hat{P}(\text{at position } i \mid \text{relevant})$ for $i = 1, 2, \ldots, 10$. Since there is a uniform number of comments at each position i, e.g. $\hat{P}(\text{at position } i) = 0.1$, we would expect $\hat{P}(\text{at position } i \mid \text{relevant})$ to be also uniform if the position of a comment and its relevance are independent. Figure 2.1 shows that this expectation is false, and that there seems to be an inverse relationship between the position of a comment and its relevance.

To see if this relationship was statistically significant, I modeled the number of relevant comments at each position i as a binomial random variable $X_i \sim \text{Bi}(n, \theta_i)$ with a known number of trials $n = 2{,}410$, and an unknown probability of success θ_i. I then used the one-tailed Fisher's exact test to reject the following system of null hypotheses at 5% significance:

$$H_0^{(ij)} : \theta_i = \theta_j, \text{ where } i, j = 1, 2, \ldots, 10, i < j$$

I rejected $H_0^{(ij)}$ for any $j - i > 3$. Namely, I failed to reject $H_0^{(ij)}$ for $(i, j) = (2, 3)$, $(4, 5)$, $(5, 6)$, $(6, 7)$, $(7, 8)$, $(7, 9)$, $(7, 10)$, $(8, 9)$, $(8, 10)$, and $(9, 10)$. The Benjamini-Hochberg procedure was used to control the false discovery rate due to multiple testing.

3 Segmented Retrieval

Modern text retrieval systems employ text segmentation during the indexing of documents. I show that, rather than returning the segments to the user, significant improvements are achieved on the semantic text similarity task by combining all segments from a single document into one result with an aggregate similarity score. Following an analysis of the SemEval-2016 and 2017 task 3 datasets, I design a segment decay weighting method that achieves state the art results on subtask B and can be readily implemented into existing inverted-index-based search engines.

3.1 Introduction

The standard bag-of-words vector space model [6] represents documents in terms of word frequencies as vectors in high-dimensional real vector spaces. The model disregards word order, which immediately limits its ability to capture the meaning of a document. Nevertheless, the model provides a notion of document similarity that is well-understood and scales to large datasets. As a result, it forms the basis of popular inverted-index-based search engines such as Apache Lucene [13], and any improvements to the model will have an immediate impact on the performance of a large body of text retrieval applications.

Long documents that cover a range of different topics provide a significant challenge for the standard model, since they are difficult to statically summarize, and deemed irrelevant to most queries. For that reason, we suggested [5] to segment the indexed documents into semantically coherent *nuggets*, and to retrieve these nuggets instead of the original documents. In this chapter, I focus on the frequent case, when the search engine is expected to retrieve full documents rather than just the nuggets relevant to a query. It might seem that, in this scenario, nuggets are useful for the summarization of results at best. Contrary to this intuition, I show that on our datasets, combining the evidence of similarity provided by the retrieved nuggets yields significant improvements on the text similarity task compared to

standard model on unsegmented documents. My results are fully reproducible.[1]

This chapter is structured as follows: In section 3.2, I review the related work. In section 3.3, I give an overview of the system without delving into the specifics of our datasets. In section 3.4 I describe the parameter estimation results, statistical observations, and techniques that I developed specifically for our datasets. Section 3.5 reports the achieved results. I conclude in section 3.6 by summarizing my results, and suggesting future work.

3.2 Related work

The notion of representing a document as a vector of term weights, and estimating the similarity between two documents by the cosine of the angle between their vectors was perhaps first explored by Salton [14] during his work on the SMART information retrieval system. Several competing methods for assigning term weights and normalizing document vectors were proposed in literature. In this work, I consider those presented by Salton; Buckley [15], Singhal et al. [16], and Murata et al. [17].

The probabilistic Okapi BM25 model is an alternative to Salton's vector space model and was developed by Robertson; Jones [18] and Robertson et al. [19, 20]. In this work, we consider the version of the model that Singhal [21] describes as the state of the art.

Assessing the similarity of two structured documents by combining the evidence of similarity provided by their *structural elements* (e.g. nuggets) has already been explored in the context of XML document retrieval. In this work, I draw inspiration from IBM Haifa's JuruXML system described by Mass et al. [22]. However, whereas XML documents have a tree structure, which makes it possible to compare segments on the basis of structural similarity, the system described in this work makes no assumptions about the structure of nuggets.

Improving unstructured text retrieval by removing or assigning different weights to *document zones* (e.g. nuggets) has been of interest to researchers in the fields of text summarization, feature selection,

1. `https://github.com/witiko-masters-thesis/segmentation`

and text classification. In this work, I consider the approaches of Kocz et al. [23] and Ko et al. [24].

3.3 System description

Our system takes as input a list of nuggets that form a single document, and preprocesses them. Given a preprocessed query document and a preprocessed result document, our system computes a single aggregate score that captures the similarity between the two documents. In the following subsections, I break the system into its individual components and describe each one in detail.

3.3.1 Nugget filtering

As a first preprocessing step, we take each nugget and make a prediction about its importance. If the predicted importance falls below a threshold, we remove the nugget. Under the hypothesis that only important nuggets contain (with a higher chance than random) important terms that describe the meaning of a document, this step extracts a summary with a high density of important terms.

In this work, I consider the following text summarization techniques proposed by Kocz et al. [23], which I modify to use nuggets rather than sentences and paragraphs as the base unit of text:

Title All nuggets other than the nugget corresponding to the title of a text document (title nugget) are removed.

FirstPara All nuggets other than the title nugget, and other than the first non-title nugget are removed.

FirstTwoPara All nuggets other than the title nugget, and other than the first two non-title nuggets are removed.

FirstLastPara All nuggets other than the title nugget, the first non-title nugget, and the last non-title nugget are removed.

ParaWithMostTitleWords All nuggets other than the title nugget, and other than the first non-title nugget that contains the highest number of tokens present in the title nugget (title tokens) are removed.

BestSentence$_l$ All nuggets other than the title nugget, and other than the nuggets that contain less than l title tokens are removed.

Note that the Title, ParaWithMostTitleWords, and BestSentence$_l$ techniques assume that a document contains a title, subject, abstract, or other form of a short summary. These techniques cannot be used without modification if this assumption is violated.

3.3.2 Scoring function

Indexing If we are indexing the nuggets, then for each nugget i that was not removed in the previous step, we record in the inverted index the term frequency vector $\mathbf{f}_i$, where f_{it} is the frequency of term t in nugget i, the number of tokens $\sum_t f_{it}$ in nugget i, and the vector $\mathbf{p}_i$, where p_{it} is the location where term t first occures in the document that contains nugget i.

We also update the collection-wide statistics that consist of the total number of nuggets in the collection N (collection size), the number of nuggets n_t that contain term t (nugget frequency), the average nugget length $\text{avg}_i(\sum_t f_{it})$, the average number of terms in a nugget $\text{avg}_i\, u_i$, and the average byte length of a nugget $\text{avg}_i\, b_i$.

Querying If the nuggets come from a query document, then for each nugget i that was not removed in the previous step, we traverse the inverted index, searching for candidate nuggets j that have at least one term in common with nugget i. For every i, j, we compute the similarity score $S(i, j)$.

With the standard model [6], we first map the nuggets i, j to nugget vectors $\mathbf{v}_i, \mathbf{v}_j$. Note that in this chapter, I will assume that all vectors are expressed in an orthonormal basis and I will treat vectors and their coordinates in this basis interchangeably. The formulas for the vectors depend on our weighting scheme. A naming convention for the various weighting schemes was proposed by Salton [14] during his work on the SMART information retrieval system and further extended in subsequent research [15, 16] (see table 3.1). Using this convention, every weighting scheme can be assigned a name in the form of $\mathbf{t}_j\mathbf{d}_j m_j \,.\, \mathbf{t}_i\mathbf{d}_i m_i$, where $\mathbf{t}_c, \mathbf{d}_c$, and $m_c, c = i, j$, stand for the term frequency, nugget frequency, and nugget length normalization factors, respectively. The nugget

vectors $\mathbf{v}_c$ then equal $\mathbf{t}_c \circ \mathbf{d}_c m_c^{-1}$, where $\circ$ denotes the entrywise (Hadamard) product. E.g. if we choose `bfx.nfx` as our weighting scheme, then $\mathbf{v}_j = \text{sign}(\mathbf{f}_j) \circ \ln \frac{N}{\mathbf{n}}$, $\mathbf{v}_i = \mathbf{f}_i \circ \ln \frac{N}{\mathbf{n}}$.

I also consider the term weighting method described by Murata et al. [17], which I will refer to as the *Murata* weighting method. The weight $K_{\text{location}}(i,t)$ of term t in a nugget i penalizes terms that appear near the end of a nugget and is defined as follows:

$$K_{\text{location}}(i,t) = \begin{cases} k_{\text{location},1} & \text{if } t \text{ occurs in a title nugget,} \\ 1 + k_{\text{location},2} \frac{\sum_t f_{it} - 2p_{it}}{\sum_t f_{it}} & \text{otherwise,} \end{cases}$$

where $k_{\text{location},1}$, and $k_{\text{location},2}$ are parameters that the authors set to $k_{\text{location},1} = 1.35$, and $k_{\text{location},2} = 0.125$ in their system A (I will refer to this method as Murata_A for ease of reference), and to $k_{\text{location},1} = 1.3$, and $k_{\text{location},2} = 0.15$ in their system B (I will refer to this method as Murata_B for ease of reference). If we let $\mathbf{e}$ denote the supplementary weight vector, such that $e_t = K_{\text{location}}(i,t)$. Then $\mathbf{v}_c = \mathbf{t}_c \circ \mathbf{e} \circ \mathbf{d}_c m_c^{-1}$, where the vector length normalization factor m_c corrects for the final weights $\mathbf{w}_c = \mathbf{t}_c \circ \mathbf{e} \circ \mathbf{d}_c$. Additional choices of the supplementary weight vector $\mathbf{e}$ were considered as a part of the dataset analysis in section 3.4.2.

Regardless of whether or not we use the supplementary weight vector $\mathbf{e}$, the value of the scoring function $S(i,j)$ equals $\mathbf{v}_i^\mathsf{T} \mathbf{v}_j$ in the standard model.

If instead of the standard model we use the Okapi BM25 model, then we define the scoring function $S(i,j)$ as follows [21]:

$$S(i,j) = \sum_{\text{term } t \in i,j} \ln \frac{N - n_k + 0.5}{n_k + 0.5} \cdot \frac{(k_1+1) f_{jt}}{k_1 \left((1-b) + b \frac{\sum_t f_{jt}}{\text{avg}_i (\sum_t f_{it})} \right) + f_{jt}} \cdot \frac{(k_3+1) f_{it}}{k_3 + f_{it}},$$

where $k_1 \in [1,2]$, $k_3 \in [0,1000]$, and $b \in [0,1]$ are parameters.

3.3.3 Result aggregation

If we are indexing a document, then the previous two steps (filtering and scoring) conclude our efforts. If, instead of indexing a document, we are processing a query document u, then we have just retrieved a number of candidate nuggets that are highly similar to at least one of

Term frequency		Nugget frequency		Normalization	
b	$\text{sign}(\mathbf{f}_i)$	x, n	$\mathbf{1}$	x, n	1
t, n	$\mathbf{f}_i$	f	$\mathbf{1} \cdot \ln \frac{N}{\mathbf{n}}$	c	$\sqrt{\mathbf{w}_i^\top \mathbf{w}_i}$
a	$\mathbf{0.5} + 0.5 \frac{\mathbf{f}_i}{\max_t f_{it}}$	t	$\mathbf{1} \cdot \ln \frac{N+1}{\mathbf{n}}$	u	$1 - s + s \frac{u_i}{\text{avg}_i u_i}$
l	$\mathbf{1} + \ln \mathbf{f}_i$	p	$\mathbf{1} \cdot \ln \frac{N-n_t}{\mathbf{n}}$	b	$1 - s + s \frac{b_i}{\text{avg}_i b_i}$
L	$\frac{\mathbf{1}+\ln \mathbf{f}_i}{1+\ln(\text{avg}_t f_{it})}$				
d	$\mathbf{1} + \ln(\mathbf{1} + \ln \mathbf{f}_i)$				

Table 3.1: The SMART notation for the weighting schemes used with the standard model, where f_{it} is the frequency of term t in a nugget i (term frequency), $\mathbf{f}_i$ is the vector of term frequencies in a nugget i, N is the total number of nuggets in the collection (collection size), n_t is the number of nuggets that contain term t (nugget frequency), $\mathbf{n}$ is the vector of nugget frequencies, u_i is the number of terms in a nugget i, b_i is the byte length of nugget i, $\mathbf{w}_i$ is the vector of term weights in a nugget i, and s is the slope in the context of pivoted document length normalization. Letters on the left uniquely identify each term frequency, nugget frequency, and nugget length normalization factor.

the query nuggets according to our scoring function S. Our task now is to use this information to return documents that satisfy the user's information need.

Unsegmented approach If we performed no segmentation, then a nugget corresponds directly to a document. In this scenario, we return to the user each candidate nugget j in decreasing order relative to the scoring function $S(i, j)$, where i is the single unsegmented query nugget.

If we performed segmentation, then for each document v (result document) having at least one nugget in the set of candidate nuggets, we retrieve the full set of v's nuggets (result nuggets) and we compute a similarity matrix $\mathbf{M}_{uv}$, where every row contains the scores between a single query nugget from u and all result nuggets from v and every column contains the scores between all query nuggets from u and a single result nugget from v. We are now interested in finding an aggregate scoring function $S'(u, v)$ that is expressed in terms of the

elements of $\mathbf{M}_{uv}$, such that the ordering of result documents induced by S' correlates with the relevance of the result documents v to the information need behind the query document u.

Should there be no upper bound on the number of nuggets in query and result documents, then the computation of $\mathbf{M}_{uv}$ may turn out to be prohibitively slow. One possible approach to speeding up the retrieval is to forgo the segmentation of query documents and to segment only the result documents instead. This is the standard approach in semi-structured XML retrieval [22], where the query constitutes only a single branch of an XML document tree. In my experiments, I did not evaluate this approach.

Machine learning approach It may happen that the indexed documents share the same structure, and the segmentation is predictable in that it always produces the same number of nuggets both per a query document and per a result document. If there is also annotated training data available, then we can build a classifier beforehand using the following procedure. For each query document u, result document v, and the corresponding relevance judgement, we concatenate the rows of $\mathbf{M}_{uv}$ into a feature vector $\mathbf{x}_{uv}$. We then perform logistic regression on the list of feature vectors and the corresponding relevance judgements to fit a classifier that assigns a posterior probability estimate $\hat{P}(\text{relevant} \mid \mathbf{x}_{uv})$ to any u and v. Our desired aggregate scoring function is then $S'(u, v) = \hat{P}(\text{relevant} \mid \mathbf{x}_{uv})$.

In practice, the segmentation is likely to produce a variable number of nuggets per document. In that scenario, we can treat matrix $\mathbf{M}_{uv}$ as a greyscale image $\mathbf{I}_{uv}$ (see figure 3.1), and set $S'(u, v) = \hat{P}(\text{relevant} \mid \mathbf{I}_{uv})$, where $\hat{P}(\text{relevant} \mid \mathbf{I}_{uv})$ is the posterior probability of an image classifier. In my experiments, I did not evaluate this approach.

Hand-crafted operators Having apriori knowledge about our documents allows us to construct rules that describe which query and result nuggets will contribute the most to the aggregate scoring function S'. Let $\ominus, \oplus$ be associative and commutative binary operators on $\mathbb{R}$ and let m_{ij} denote the value of a matrix $\mathbf{M}_{uv}$ in the row and column corresponding to the query and result nuggets i and j. Then we can

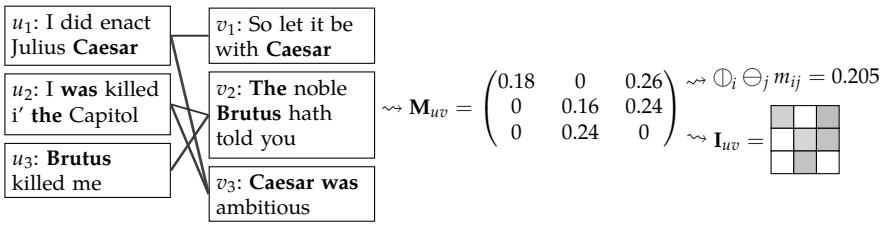

Figure 3.1: Given query and result documents u and v consisting of nuggets u_1, u_2, u_3, v_1, v_2, and v_3, we compute a similarity matrix $\mathbf{M}_{uv}$ using the bnc.bnc standard model weighting scheme. Using the handcrafted operators $\ominus = \oplus = \text{wavg}_{\text{length}'}$ we compute the aggregate score $S'_{\text{result-first}}$ (above), or we convert $\mathbf{M}_{uv}$ to a grayscale image $\mathbf{I}_{uv}$ that we pass to an image classifier (below).

express our aggregate scoring function as either

$$S'_{\text{query-first}}(u, v) = \bigominus_{\text{result nugget } j \in v} \bigoplus_{\text{query nugget } i \in u} m_{ij}, \text{ or}$$
$$S'_{\text{result-first}}(u, v) = \bigoplus_{\text{query nugget } i \in u} \bigominus_{\text{result nugget } j \in v} m_{ij} \text{ (see figure 3.1).}$$

In my experiments, I evaluated $\ominus, \oplus \in \{\text{min}, \text{max}, \text{avg}, \text{wavg}\}$ with various weighting methods for the wavg operator, such as $\text{wavg}_{\text{length}'}$ which assigns weights proportional to the number of tokens in a nugget, and $\text{wavg}_{\text{Koetal}}$ inspired by the work of Ko et al. [24], which assigns weights proportional to the score between a nugget and the title nugget of its document. Additional weighting methods were developed as a part of the dataset analysis (see section 3.4.2). Note that for $\ominus, \oplus \in \{\text{avg}, \text{wavg}\}$, we obtain $S'_{\text{query-first}} = S'_{\text{result-first}'}$ and that for $\ominus, \oplus \in \{\text{min}, \text{max}, \text{avg}\}$ such that $\ominus = \oplus$, the following holds:

$$S'_{\text{query-first}}(u, v) = S'_{\text{result-first}}(u, v) = \bigominus_{\text{result nugget } j \in v, \text{ query nugget } i \in u} m_{ij}.$$

3.4 Experimental setup

In this section, I give an overview of the experiments performed on the QA datasets described in chapter 2.

3.4.1 Language modeling and segmentation

The texts in the datasets were lower-cased, stripped of images and URLs, and tokenized on white spaces and punctuation. Tokens shorter than two characters or longer than 15 characters were removed. Using the existing structure of the datasets, every original question was split into two nuggets corresponding to the question subject and text, and every candidate thread was split into twelve nuggets corresponding to the related question subject and text, and the initial ten comments.

Since the annotated datasets did not contain enough text to build a proper language model, I used the unannotated subtask A datasets to obtain the collection-wide statistics required to compute the scoring function described in section 3.3.2.

3.4.2 Dataset analysis

Folowing the relevance analysis in section 2.1, I used the subtask B training dataset to fit a classifier using the machine learning approach described in section 3.3.3. In logistic regression, the posterior probability estimate $\hat{P}(\text{relevant} \mid \mathbf{x}_{uv})$ is modelled as follows:

$$\ln \frac{\hat{P}(\text{relevant} \mid \mathbf{x}_{uv})}{1 - \hat{P}(\text{relevant} \mid \mathbf{x}_{uv})} = \beta_0 + \boldsymbol{\beta}^\mathsf{T} \mathbf{x}_{uv}, \tag{3.1}$$

where $\beta_0, \boldsymbol{\beta}$ are the machine-learned coefficients. If we let $\beta_j, j = 1, 2, \ldots, 24$ denote the individual elements of $\boldsymbol{\beta}$, then $|\beta_{i+2}|$, and $|\beta_{i+14}|$ correspond to the discriminativeness of the elements $m_{1,i+2}$ and $m_{2,i+2}$ in a matrix $\mathbf{M}_{uv}$. These elements in turn correspond to the score between the original question text and a comment at position i, and the score between the original question subject and a comment at position i (recall that a thread consists of twelve nuggets, which means that $\mathbf{M}_{uv}$ consists of twelve columns). Plotting $|\beta_{i+2}|$ and $|\beta_{i+14}|$ against the comment position i in figure 3.2 shows that later comments are in general less discriminative than earlier comments. Note that the classifier discovers this relationship without access to the relevance judgements about comments.

These discoveries led us to design and evaluate the $\text{wavg}_{\text{Godwin}}$ operator, which assigns a weight proportional to i^{-1} to a nugget at position i in accordance to *Zipf's law*. This decreases the effect

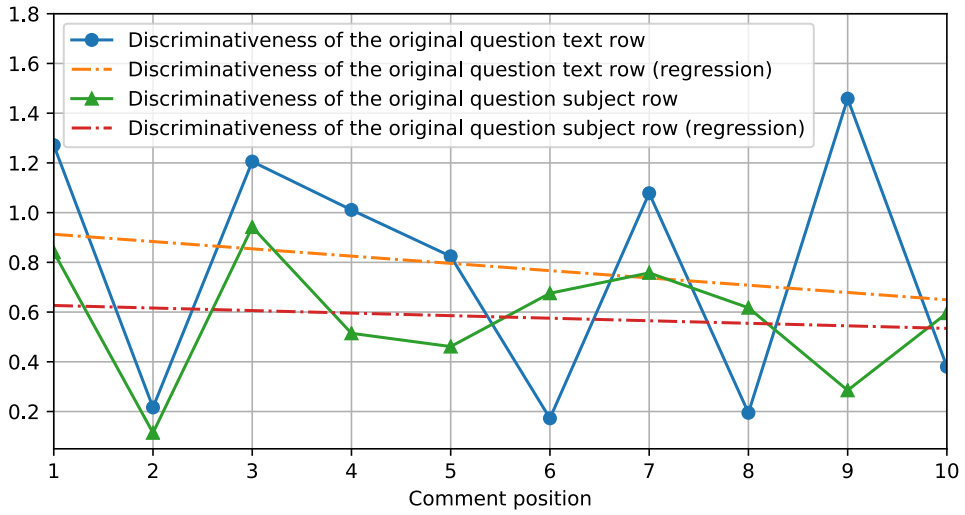

Figure 3.2: The discriminativeness of the individual elements of a matrix $\mathbf{M}_{uv}$ expressed by the absolute values of the machine-learned coefficients $\boldsymbol{\beta}$. The discriminativeness of the original question text row corresponds to $|\beta_{i+2}|, i = 1, 2. \ldots, 10$, and the discriminativeness of the original question subject row corresponds to $|\beta_{i+14}|, i = 1, 2, \ldots, 10$.

of comments that are likely to be irrelevant. Under the hypothesis that only relevant comments contain (with a higher chance than random) important terms that describe the meaning of a document, this weighting scheme pays attention to scores between those nuggets that are likely to contain important terms.

Since weighting terms is conceptually and computationally simpler than segmentation and result aggregation, I will experimentally verify that the segmentation is meaningful and that the relevance loss occurs at nugget boundaries rather than at term boundaries. For that reason, I designed the *Godwin* term weighting method for the standard model scoring function. For each term t at positions $i_1, i_2, \ldots, i_n$ in a nugget, the method assigns a weight proportional to $\sum_{j=1}^{n} i_j^{-1}$ to the supplementary weight vector element e_t described in section 3.3.2. It is easy to show that, given the right choice of term frequency factor (`t` in table 3.1) and vector length normalization factor (`n` in table 3.1), the standard model scoring function produces the same ordering on unsegmented threads as the aggregate scoring function defined in

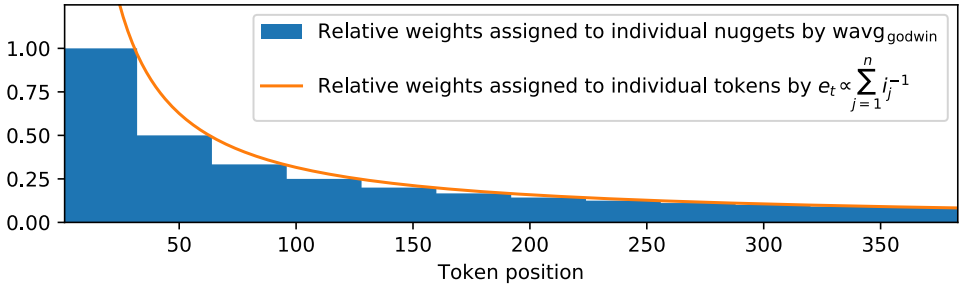

Figure 3.3: The relative weights assigned to the individual columns of a matrix $\mathbf{M}_{uv}$ by the $\text{wavg}_{\text{Godwin}}$ operator plotted along the relative weights assigned to the individual tokens by the Godwin term weighting method. The figure assumes the mean number of tokens per a thread in the subtask A unannotated datasets (383 tokens), and uniform nugget length in tokens.

terms of the $\ominus = \text{wavg}_{\text{Godwin}}, \oplus = \text{wavg}_{\text{length}}$ operators would on threads segmented to one nugget per a token (see figure 3.3).

3.4.3 Parameter estimation

Some parameters of the scoring functions described in section 3.3.2 require careful tuning to each particular dataset. For the standard model, this corresponds to the slope parameter s, which is used by the normalization factors u and b from table 3.1 as a part of pivoted document normalization [16]. For Okapi BM25, this corresponds to the parameters k_1, k_3, and b. Apart from the scoring functions, the parameter l of the BestSentence_l nugget filtering method introduced in section 3.3.1 also requires tuning.

To find the optimum values of s, we initially set $s = 0.3$ as suggested by Singhal et al. [16]. By evaluating all the considered configurations against the subtask B dev dataset, we obtained the top six configurations using the normalization factors u and b, and taking either the unsegmented, machine learning, or hand-crafted operator approach to result aggregation. I then ran a grid search to find the values of s in the interval $s \in [0, 1]$ maximizing the performance of these eight configurations. Figure 3.4 plots the achieved MAP scores against the choices of parameter s.

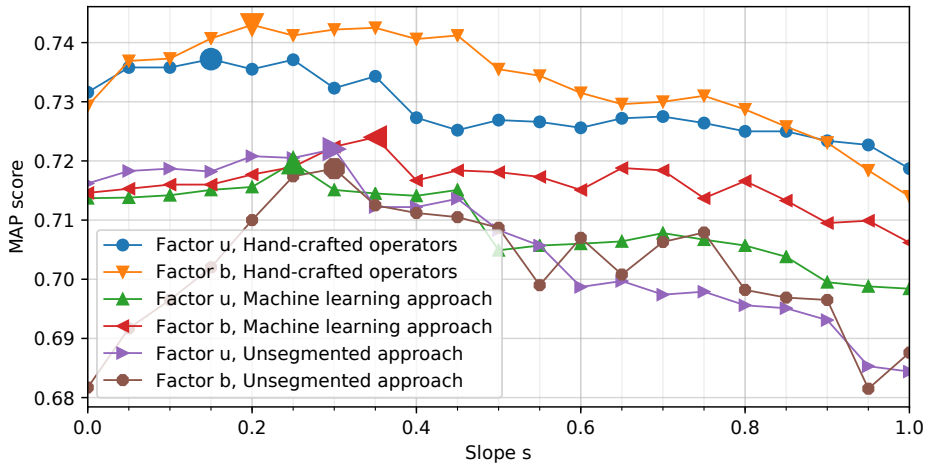

Figure 3.4: The MAP scores achieved with the pivoted normalization factors u and b for the various choices of the slope parameter s.

To find the optimum values of k_1, k_3, and b, we initially set $k_1 = 1.2, k_3 = 1000$, and $b = 0.75$ as suggested by Singhal [21]. Using the same procedure as above in the intervals $k_1 \in [1, 2], k_3 \in [0, 1000]$, and $b \in [0, 1]$, we found the optimal parameters to be $k_1 = 2, k_3 = 950$, and $b = 0$ for the unsegmented approach, $k_1 = 2, k_3 = 0$, and $b = 0.80$ for the machine learning approach, and $k_1 = 1.2, k_3 = 0$, and $b = 0.75$ for the hand-crafted operator approach to result aggregation.

To find the optimum values of l, we took the best-performing configuration using the hand-crafted operator approach on the subtask B dev dataset (note that BestSentence$_l$ is a nugget filtering method producing a variable number of nuggets, which makes it unsuitable for both the unsegmented and the machine learning approach). I then ran a grid search to find the values of l in the interval $l \in [0, 5]$ maximizing the performance of this configuration. Although Kocz et al. [23] suggest the value of $l = 3$ in the context of news articles, the typical question subjects and nuggets in our datasets are shorter than the typical news article titles and paragraphs. As a result, we found more success with less aggressive filtering using $l = 0$, and $l = 1$.

3.5 Results

All the 44,496 configurations (twelve nugget filtering methods, 17 preselected standard model weighting schemes, four supplementary word vectors, four sets of Okapi BM25 parameters, 36 pairs of handcrafted operators, and two aggregate scoring functions $S'_{\text{query-first}}$ and $S'_{\text{result-first}}$) were evaluated on the SemEval-2016 and 2017 task 3 subtask B QA datasets. Tables 3.2 and 3.3 show the results. Although various configurations achieved outstanding MAP scores on one or the other dataset, only the *primary* configuration (highlighted in bold and italics) consistently achieved scores higher than the winning SemEval submissions. This configuration uses the `bfx.nfx` standard model weighting scheme suggested by Salton; Buckley [15] for short and homogeneous documents, and the $\ominus = \text{wavg}_{\text{Godwin}}$ handcrafted operator developed in section 3.4.2. This shows that the $\text{wavg}_{\text{Godwin}}$ operator is well-suited to our datasets and hopefully to QA datasets in general.

Several *contrastive* configurations that use the unsegmented approach to result aggregation were derived from the primary configuration. The contrastive configuration using the Godwin term weighting (denoted ***`bfx.nfx`***, *Godwin* in the tables) performed consistently worse than the primary configuration, which shows that the segmentation to nuggets is meaningful and cannot be replaced with term weighting.

The two remaining contrastive configurations using no nugget filtering and the FirstTwoPara nugget filtering show that without segmentation, the standard model is unable to cope with the outlier terms introduced by irrelevant comments. Removing all nuggets other than the question subject and text improves the MAP score, but at the cost of losing important terms. The primary configuration achieves the highest MAP score by properly weighting the individual nuggets.

3.6 Conclusion and future work

Segmentation matters and so does careful weighting. By combining both, I was able to achieve state-of-the-art results on the SemEval-2016 and 2017 task 3 subtask B QA datasets using the standard bag-of-words vector space model without semantic modelling. Our method can

Rslt. aggregation	Nggt. filtering	Scoring function	Operator $\ominus$	Operator $\oplus$	S'	MAP
Handcrafted	FirstTwoPara	`dpc.ann`	max	$\text{wavg}_{\text{length}}$	$S'_{\text{query-first}}$	77.25
Handcrafted	—	`bfx.nfx`	$\text{wavg}_{\text{Godwin}}$	$\text{wavg}_{\text{Godwin}}$	$S'_{\text{result-first}}$	77.23
Handcrafted	FirstTwoPara	`dpc.ann`	max	$\text{wavg}_{\text{length}}$	$S'_{\text{result-first}}$	77.09
Handcrafted	FirstTwoPara	`nfc.dfc`, Murata_{B}	avg	min	$S'_{\text{result-first}}$	77.00
Handcrafted	FirstTwoPara	`Lpc.ann`	$\text{wavg}_{\text{length}}$	max	$S'_{\text{query-first}}$	76.96
Handcrafted	—	***`bfx.nfx`***	$\boldsymbol{wavg_{Godwin}}$	$\boldsymbol{wavg_{length}}$	$\boldsymbol{S'_{result\text{-}first}}$	**76.77**
SemEval-2016 task 3 subtask B winner (UH-PRHLT-primary)				—	—	**76.70**
Unsegmented	FirstTwoPara	`Lfb.bfc`, Murata_{B}	—	—	—	76.58
Machine learning	FirstPara	`nfc.lfc`	—	—	—	75.63
Unsegmented	*FirstTwoPara*	*`bfx.nfx`*	—	—	—	*75.21*
SemEval-2016 task 3 subtask B IR baseline			—	—	—	**74.75**
Unsegmented	—	*`bfx.nfx`*	—	—	—	*73.94*
Unsegmented	—	*`bfx.nfx`, Godwin*	—	—	—	*70.28*

Table 3.2: Results for the top five configurations on the 2016 subtask B test dataset, the primary configuration (highlighted in bold and italics) with its contrastive configurations (highlighted in italics), the top unsegmented and machine learning configurations. The IR baseline and the winning submission are highlighted in bold.

Rslt. aggregation	Nggt. filtering	Scoring function	Operator $\ominus$	Operator $\oplus$	S'	MAP
Handcrafted	BestSentence_0	Okapi BM25	avg	max	$S'_{\text{result-first}}$	49.20
Handcrafted	BestSentence_0	Okapi BM25	avg	$\text{wavg}_{\text{Godwin}}$	$S'_{\text{result-first}}$	48.87
Handcrafted	BestSentence_0	Okapi BM25	avg	max	$S'_{\text{query-first}}$	48.79
Handcrafted	BestSentence_0	`bfx.nfx`, Murata_A	max	$\text{wavg}_{\text{Godwin}}$	$S'_{\text{result-first}}$	48.76
Handcrafted	BestSentence_0	Okapi BM25	avg	$\text{wavg}_{\text{Godwin}}$	$S'_{\text{result-first}}$	48.76
Unsegmented	FirstTwoPara	`Lpu.Lpc`, Murata_A	—	—	—	48.75
Handcrafted	—	***bfx.nfx***	$\mathit{wavg}_{\mathit{Godwin}}$	$\mathit{wavg}_{\mathit{length}}$	$S'_{\mathit{result\text{-}first}}$	***47.45***
SemEval-2017 task 3 subtask B winner (SimBow-primary)				—	—	**47.22**
Machine learning	FirstTwoPara	`bfx.nfx`	—	—	—	46.90
Unsegmented	*FirstTwoPara*	***bfx.nfx***	—	—	—	*44.67*
SemEval-2017 task 3 subtask B IR baseline			—	—	—	**41.85**
Unsegmented	—	***bfx.nfx**, Godwin*	—	—	—	*37.18*
Unsegmented	—	***bfx.nfx***	—	—	—	*36.82*

Table 3.3: Results for the top five configurations on the 2017 subtask B test dataset, the primary configuration (highlighted in bold and italics) with its contrastive configurations (highlighted in italics), the top unsegmented and machine learning configurations. The IR baseline and the winning submission are highlighted in bold.

be readily implemented into existing inverted-index-based search engines.

We have recently shown [4] that arbitrary vector space models (e.g. LSA, LDA, Doc2vec, ...) can be used to represent documents in inverted-index-based search engines. Evaluating how the concepts of segmentation and result aggregation interact with semantic models that are not based on a bag-of-words representation is an intriguing direction for future research.

I have shown in section 3.4.2 that there is a statistically significant relationship between the position of a comment and its relevance in the SemEval-2016 and 2017 subtask A datasets. Investigating whether such a relationship exists in other QA datasets (and other datasets in general) will provide us with new insights to the dynamics of online discourse and lead to more effective retrieval systems.

4 Modeling Synonymy

Standard text retrieval methods underestimate the semantic similarity between documents that use synonymous terms. Latent semantic indexing (LSA) tackles the problem by clustering frequently co-occuring terms at the cost of the periodical reindexing of dynamic document collections and the suboptimality of co-occurences as a measure of synonymy. In this chapter, I develop a term similarity model that suffers neither of these flaws. I analyze the associated computational complexity, show how the model can be implemented into existing IR systems, and evaluate its performance on the semantic text similarity task.

4.1 Introduction

The standard bag-of-words vector space model [6] represents documents as vectors in high-dimensional real vector spaces. The documents are traditionally expressed in a basis where each basis vector corresponds to a single term, and each coordinate corresponds to the frequency of a term in our document. Consider the documents

$$d_1 = \text{“When Antony found Julius Caesar dead”, and}$$
$$d_2 = \text{“I did enact Julius Caesar: I was killed i’ the Capitol”}$$

represented in a basis $\{\boldsymbol{\alpha}_i\}_{i=1}^{14}$ of $\mathbb{R}^{14}$, where the basis vectors corresponds to the following terms: when, Anthony, found, Julius, Caesar, dead, I, did, enact, was, killed, i’, the, Capitol. Then the corresponding vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ would have the following coordinates in basis $\boldsymbol{\alpha}$:

$$\mathbf{v}_1^{\boldsymbol{\alpha}} = [1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0]^{\mathsf{T}}, \text{and}$$
$$\mathbf{v}_2^{\boldsymbol{\alpha}} = [0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1\,1\,1\,1]^{\mathsf{T}}.$$

Since rare words are more important for describing a document than frequent words, we generally do not assume that $\boldsymbol{\alpha}$ is orthonormal, and we cast the coordinates into an orthonormal basis $\boldsymbol{\delta}$ chosen in such a way that the frequencies of rare words are amplified. However, the assumption is usually made that $\boldsymbol{\alpha}$ is orthogonal. This has the

practical advantage that we can directly compute the inner product as a measure of similarity between any two vectors expressed in $\boldsymbol{\alpha}$. Assuming $\boldsymbol{\alpha}$ is orthonormal, we may take the inner product between the normalized $\mathbf{v}_1$ and $\mathbf{v}_2$ to measure the similarity between d_1 and d_2:

$$\langle \mathbf{v}_1/\|\mathbf{v}_1\|, \mathbf{v}_2/\|\mathbf{v}_2\| \rangle_{\mathbb{R}^{14}} = \left((\mathbf{v}_1^{\boldsymbol{\alpha}})^\mathsf{T}\mathbf{v}_2^{\boldsymbol{\alpha}}\right) \Big/ \left(\sqrt{(\mathbf{v}_1^{\boldsymbol{\alpha}})^\mathsf{T}\mathbf{v}_1^{\boldsymbol{\alpha}}}\sqrt{(\mathbf{v}_2^{\boldsymbol{\alpha}})^\mathsf{T}\mathbf{v}_2^{\boldsymbol{\alpha}}}\right) \approx 0.26.$$

Intuitively, this underestimates the true similarity between d_1 and d_2. If we assume $\boldsymbol{\alpha}$ is non-orthonormal, and that the terms Anthony, Julius, and Caesar are five times more important than the other terms, then we might construct a diagonal change-of-basis matrix[1] $\mathbf{W} = (w_{ij})$ from $\boldsymbol{\alpha}$ to an orthonormal basis $\boldsymbol{\delta}$, where

$$w_{ij} = \begin{cases} 5 & \text{if } i = j, i \in \{\text{Anthony}, \text{Julius}, \text{Caesar}\}, \\ 1 & \text{if } i = j, i \notin \{\text{Anthony}, \text{Julius}, \text{Caesar}\}, \text{and} \\ 0 & \text{if } i \neq j. \end{cases} \tag{4.1}$$

Then $\mathbf{v}_1$ and $\mathbf{v}_2$ would have the following coordinates in $\boldsymbol{\delta}$:

$$\mathbf{v}_1^{\boldsymbol{\delta}} = \mathbf{W}\mathbf{v}_1^{\boldsymbol{\alpha}} = [1\,5\,1\,5\,5\,1\,0\,0\,0\,0\,0\,0\,0\,0]^\mathsf{T}, \text{and}$$
$$\mathbf{v}_2^{\boldsymbol{\delta}} = \mathbf{W}\mathbf{v}_2^{\boldsymbol{\alpha}} = [0\,0\,0\,5\,5\,0\,1\,1\,1\,1\,1\,1\,1\,1]^\mathsf{T},$$

and our inner product would change to

$$\langle \mathbf{v}_1/\|\mathbf{v}_1\|, \mathbf{v}_2/\|\mathbf{v}_2\| \rangle_{\mathbb{R}^{14}} = \left((\mathbf{v}_1^{\boldsymbol{\delta}})^\mathsf{T}\mathbf{v}_2^{\boldsymbol{\delta}}\right) \Big/ \left(\sqrt{(\mathbf{v}_1^{\boldsymbol{\delta}})^\mathsf{T}\mathbf{v}_1^{\boldsymbol{\delta}}}\sqrt{(\mathbf{v}_2^{\boldsymbol{\delta}})^\mathsf{T}\mathbf{v}_2^{\boldsymbol{\delta}}}\right) =$$
$$= \left((\mathbf{W}\mathbf{v}_1^{\boldsymbol{\alpha}})^\mathsf{T}(\mathbf{W}\mathbf{v}_2^{\boldsymbol{\alpha}})\right) \Big/ \left(\sqrt{(\mathbf{W}\mathbf{v}_1^{\boldsymbol{\alpha}})^\mathsf{T}(\mathbf{W}\mathbf{v}_1^{\boldsymbol{\alpha}})}\sqrt{(\mathbf{W}\mathbf{v}_2^{\boldsymbol{\alpha}})^\mathsf{T}(\mathbf{W}\mathbf{v}_2^{\boldsymbol{\alpha}})}\right) \approx 0.89.$$

Intuitively, this comes closer to the true similarity between d_1 and d_2.

Although the assumption that $\boldsymbol{\alpha}$ is orthogonal has been an inherent part of the standard model since its inception, it makes modeling synonymy difficult. The terms dead and killed contribute nothing to the inner product of $\mathbf{v}_1$ and $\mathbf{v}_2$ despite the clear synonymy, since we assume $\langle \boldsymbol{\alpha}_{\text{dead}}, \boldsymbol{\alpha}_{\text{killed}} \rangle_{\mathbb{R}^{14}} = 0$. In general, the standard model will underestimate the true similarity between documents that carry the

1. The main diagonal of $\mathbf{W}$ corresponds to the nugget frequency factor in table 3.1.

same meaning, but use different terminology. In this chapter, I extend the standard model to non-orthogonal bases and experimentally evaluate several properties of the extended model.[2]

The chapter is structured as follows: In section 4.2, I review the previous research in the area of modeling synonymy in the standard model. I provide a mathematical definition of the extended standard model and place upper asymptotic complexity bounds on computing the similarity between two documents in sections 4.3 and 4.4. I then develop several term similarity matrices for the extended model in section 4.5. In section 4.6, I consider several approaches to implementing the extension into existing tools. I describe the experiments that I carried out on the dataset in section 4.7; the results are presented in section 4.8. I conclude in section 4.9 by summarizing the results, and suggesting future work.

4.2 Related work

The notion of modeling document similarity using a non-orthogonal coordinate system was perhaps first explored by Sidorov et al. [25] in the context of entrance exam question answering, where the basis vectors did not correspond directly to terms, but rather to n-grams constructed by following paths in syntactic trees. They derive the inner product between two basis vectors from the edit distance between the corresponding n-grams.

A notable contribution of Sidorov et al. [25] is a formula for computing the inner product between two vectors expressed in a non-orthogonal basis without performing an explicit change of basis. The formula was termed *soft cosine similarity*, perhaps because the inner product between two normalized vectors corresponds to the cosine of the angle between the vectors (often referred to as the *cosine similarity*), and the nonzero inner product between basis vectors induces soft clustering of the individual n-grams. Notably, they also develop two algorithms for casting the coordinates to an orthonormal basis. In section 4.4, I show that one of the algorithms is time-suboptimal.

Charlet; Damnati [7] won SemEval-2017 task 3 subtask B [12] by training a supervised classifier using soft cosine similarity between

2. https://github.com/witiko-masters-thesis/similarity

every possible combination of subparts of an original question and a thread. Unlike Sidorov et al. [25], Charlet; Damnati [7] already use basis vectors that correspond to terms. Notably, they derive the inner product between two basis vectors from the inner product between the corresponding terms embedded in a Word2vec [26] space. I further develop this notion in section 4.5.

4.3 Extended model

In this section, I state a formal definition of the extended standard model and describe key properties of the model.

Let $\mathbb{R}^n$ be the finite inner-product space over $\mathbb{R}$ from which we draw our feature vectors. Let $\langle \cdot, \cdot \rangle_{\mathbb{R}^n}$ be the bilinear inner product on $\mathbb{R}^n$ and $\{\boldsymbol{\beta}_i\}_{i=1}^n$ the basis in which our feature vectors are expressed. Assume that $\boldsymbol{\beta}$ is non-orthogonal, i.e. that $\exists i, j : i \neq j, \langle \boldsymbol{\beta}_i, \boldsymbol{\beta}_j \rangle_{\mathbb{R}^n} > 0$. Let $\mathbf{W} = (w_{ij})$ be a diagonal change-of-basis matrix from $\boldsymbol{\beta}$ to a non-orthogonal normalized basis $\{\boldsymbol{\gamma}_i\}_{i=1}^n$ of $\mathbb{R}^n$, i.e. $\forall i, j : \langle \boldsymbol{\gamma}_i, \boldsymbol{\gamma}_j \rangle_{\mathbb{R}^n} \in [-1, 1], \langle \boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i \rangle_{\mathbb{R}^n} = 1$. Let $\mathbf{S} = (s_{ij})$, where $s_{ij} = \langle \boldsymbol{\gamma}_i, \boldsymbol{\gamma}_j \rangle_{\mathbb{R}^n}$. Since $\mathbf{S}$ is a matrix of inner products between linearly independent vectors, it is Gramian and therefore positive definite and Hermitian. Let $\mathbf{E}$ denote the change-of-basis matrix from $\boldsymbol{\gamma}$ to some orthonormal basis $\boldsymbol{\delta}$ of $\mathbb{R}^n$.

Take any two feature vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and let $\mathbf{x}^{\boldsymbol{\beta}}, \mathbf{y}^{\boldsymbol{\beta}}$ denote the coordinates of $\mathbf{x}$ and $\mathbf{y}$ in the basis $\boldsymbol{\beta}$. By expressing $\mathbf{x}$ and $\mathbf{y}$ in the orthonormal basis $\boldsymbol{\delta}$, we can express the inner product between $\mathbf{x}$ and $\mathbf{y}$ in terms of an algebraic dot product, i.e.

$$\begin{aligned}
\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^n} &= (\mathbf{x}^{\boldsymbol{\delta}})^\mathsf{T} (\mathbf{y}^{\boldsymbol{\delta}}) = (\mathbf{E}\mathbf{x}^{\boldsymbol{\gamma}})^\mathsf{T} (\mathbf{E}\mathbf{y}^{\boldsymbol{\gamma}}) = (\mathbf{E}\mathbf{W}\mathbf{x}^{\boldsymbol{\beta}})^\mathsf{T} (\mathbf{E}\mathbf{W}\mathbf{y}^{\boldsymbol{\beta}}) = \\
&= \left(\sum_{i=1}^n \boldsymbol{\beta}_i^{\boldsymbol{\delta}} w_{ii} x_i^{\boldsymbol{\beta}} \right) \left(\sum_{j=1}^n \boldsymbol{\beta}_i^{\boldsymbol{\delta}} w_{ii} y_i^{\boldsymbol{\beta}} \right) = \sum_{i=1}^n \sum_{j=1}^n w_{ii} x_i^{\boldsymbol{\beta}} w_{jj} y_j^{\boldsymbol{\beta}} \langle \boldsymbol{\beta}_i, \boldsymbol{\beta}_j \rangle_{\mathbb{R}^n} = \\
&= \sum_{i=1}^n \sum_{j=1}^n w_{ii} x_i^{\boldsymbol{\beta}} w_{jj} y_j^{\boldsymbol{\beta}} s_{ij} = (\mathbf{W}\mathbf{x}^{\boldsymbol{\beta}})^\mathsf{T} \mathbf{S} \mathbf{W} \mathbf{y}^{\boldsymbol{\beta}} = (\mathbf{x}^{\boldsymbol{\gamma}})^\mathsf{T} \mathbf{S} \mathbf{y}^{\boldsymbol{\gamma}} \quad (4.2)
\end{aligned}$$

Therefore we only need to supply a positive-definite Hermitian matrix $\mathbf{S}$ and a diagonal matrix $\mathbf{W}$ to compute the inner product between $\mathbf{x}$ and $\mathbf{y}$; computing $\mathbf{E}$ is unnecessary. I give several examples of such matrices $\mathbf{S}$ in section 4.5. From here, we can directly derive the cosine

of the angle between $\mathbf{x}$ and $\mathbf{y}$, i.e. the soft cosine similarity formula

$$\begin{aligned}\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle_{\mathbb{R}^n} &= \frac{(\mathbf{x}^\gamma)^\mathsf{T} \mathbf{S} \mathbf{y}^\gamma}{\sqrt{((\mathbf{x}^\gamma)^\mathsf{T} \mathbf{S} \mathbf{x}^\gamma} \sqrt{(\mathbf{y}^\gamma)^\mathsf{T} \mathbf{S} \mathbf{y}^\gamma}} = \\ &= \frac{(\mathbf{W}\mathbf{x}^\beta)^\mathsf{T} \mathbf{S} \mathbf{W} \mathbf{y}^\beta}{\sqrt{((\mathbf{W}\mathbf{x}^\beta)^\mathsf{T} \mathbf{S} \mathbf{W} \mathbf{x}^\beta} \sqrt{(\mathbf{W}\mathbf{y}^\beta)^\mathsf{T} \mathbf{S} \mathbf{W} \mathbf{y}^\beta}}.\end{aligned} \tag{4.3}$$

4.4 Complexity analysis

In this section, I place upper worst-case asymptotic complexity bounds on computing the inner product and on casting vector coordinates to an orthonormal basis. Unlike previous work, I focus on bases that are mostly orthogonal, i.e. bases whose corresponding matrices $\mathbf{S}$ are very sparse outside the main diagonal, and prove lower complexity bounds for such bases.

Following the notation of Manning et al. [27], let $M_{\max}$ denote the maximum number of terms in a document, and let C be the maximum number of nonzero non-diagonal elements in a row of a matrix $\mathbf{S}$.

4.4.1 Inner product

Time complexity First, I show that a time-optimal algorithm for computing the inner product between two document vectors $\mathbf{x}$ and $\mathbf{y}$ will perform no more than $(C+1)M_{\max}$ steps. To see why, consider the expression

$$\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ii} x_i^\beta w_{jj} y_j^\beta s_{ij}$$

from equation 4.2. By our assumption, x_i^β is nonzero for at most $M_{\max}$ values of the variable i and for each fixed i, s_{ij} is nonzero for at most $C+1$ values of the variable j (the additional element corresponds to the ones on the main diagonal of $\mathbf{S}$). By using an appropriate data structure to represent the vectors $\mathbf{x}$ and $\mathbf{y}$, e.g. lists of nonzero coordinates in the basis $\boldsymbol{\beta}$, the matrix $\mathbf{W}$, e.g. an array corresponding to the main diagonal of $\mathbf{W}$, and the matrix $\mathbf{S}$, e.g. an array of rows represented as a list of nonzero non-diagonal elements, a time-optimal algorithm will perform at most $(C+1)M_{\max}$ steps.

Space complexity Next, I will show that a space-optimal algorithm for computing the inner product between two document vectors $\mathbf{x}$ and $\mathbf{y}$ will occupy no more than $2M_{\max} + (n+1)C$ units of space, where n is the number of basis vectors in the bases $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\delta}$ as well as the number of terms in our vocabulary. To see why, consider the data structures given above. The lists representing the vectors $\mathbf{x}$ and $\mathbf{y}$ will occupy at most $M_{\max}$ units of space each, the array representing the matrix $\mathbf{W}$ will occupy n units of space, and the sparse matrix $\mathbf{S}$ will occupy at most nC units of space.

Assuming that $M_{\max}$ and C are arbitrary, i.e. at most n, we get a worst-case time and space complexity of $\Theta(n^2)$. According to the *Heaps' law*, $n^2 \approx T$, where T is the size of the document collection in tokens. Given the size of today's document collections, which are routinely "sharded" between several computers since they cannot fit into the memory of a single device, such space complexity is practically intractable outside distributed computing.

Let us instead assume that $M_{\max}$ is a constant, i.e. that we can place an upper limit on how large the vocabulary of any single document can be, e.g. by limiting the maximum length of a document. Let us also assume that we can build the matrix $\mathbf{S}$ in such a way that C is a constant, which will be the topic of section 4.5. Under these assumptions, the worst-case time complexity of computing the inner product is $\Theta(1)$ and the worst-case space complexity is $\Theta(n)$.

4.4.2 Orthonormalization

One way to cast the coordinates of a vector from the basis $\boldsymbol{\beta}$ to an orthonormal basis is to compute the change-of-basis matrix $\mathbf{E}$, since for any document vector $\mathbf{x}$,

$$\mathbf{x}^{\boldsymbol{\delta}} = \mathbf{E}\mathbf{x}^{\boldsymbol{\gamma}} = \mathbf{E}\mathbf{W}\mathbf{x}^{\boldsymbol{\beta}}.$$

For ease of reference, I will use $\Phi_1(\mathbf{x}^{\boldsymbol{\beta}})$ to denote $\mathbf{E}\mathbf{W}\mathbf{x}^{\boldsymbol{\beta}}$. Another way to cast the coordinates of a vector from the basis $\boldsymbol{\beta}$ to an orthonormal basis is to express the vector in an orthonormal basis $\boldsymbol{\delta}'$ of $\mathbb{R}^{n^2}$. As

shown by Sidorov et al. [25, sec. 2.4], for any document vectors $\mathbf{x}, \mathbf{y}$,

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^n} = \langle \mathbf{x}', \mathbf{y}' \rangle_{\mathbb{R}^{n^2}}, \text{ where } x'^{\delta'}_{ij} = \sqrt{2c_{ij}} \frac{w_{ii} x_i^{\beta} + w_{jj} x_j^{\beta}}{2},$$
$$y'^{\delta'}_{ij} = \sqrt{2c_{ij}} \frac{w_{ii} y_i^{\beta} + w_{jj} y_j^{\beta}}{2}, \text{ and } c_{ij} = \begin{cases} \frac{1 - \sum_{k \neq i} s_{ik}}{2} & \text{if } i = j, \text{ and} \\ s_{ij} & \text{otherwise.} \end{cases} \quad (4.4)$$

For ease of reference, I will use $\Phi_2(\mathbf{x}^{\beta})$ to denote $\mathbf{x}'^{\delta'}$.

Time complexity First, I will show that a time-optimal algorithm for casting the coordinates of a document vector $\mathbf{x}$ to an orthonormal basis will perform no more than $M_{\max}^2$ steps. To see why, consider the expression

$$x'^{\delta'}_{ij} = \sqrt{2c_{ij}} \frac{w_{ii} x_i^{\beta} + w_{jj} x_j^{\beta}}{2}$$

that corresponds to a single coordinate of $\Phi_2(\mathbf{x}^{\beta})$ in equation 4.4. By the assumption from the beginning of the section, x_i^{β} and x_j^{β} are nonzero for at most $M_{\max}$ values of the variables i and j. By using an appropriate data structure to represent $\mathbf{x}$, e.g. a list of nonzero coordinates in the basis β, an optimal algorithm will compute the $M_{\max}^2$ nonzero coordinates of $\mathbf{x}'^{\delta'}$ using no more than $M_{\max}^2$ steps.

By the definition of $\mathbf{S}$, $\mathbf{E}\mathbf{E}^{\mathsf{T}} = \mathbf{S}$. Furthermore, Sidorov et al. [25, sec. 2.5] show that $\mathbf{E}$ is lower-diagonal. Therefore, by the positive-definiteness and the hermicity of $\mathbf{S}$, $\mathbf{E}$ is uniquely determined by the *Cholesky factorization* of $\mathbf{S}$. Using the well-known Cholesky algorithm, we can compute $\mathbf{E}$ in $\mathcal{O}(n^3)$ steps. This improves on the result of Sidorov et al. [25, sec. 2.5] who show that $\mathbf{E}$ can be computed from $\mathbf{S}$ in $\mathcal{O}(n^4)$ steps. For a document vector $\mathbf{x}$, a time-optimal algorithm can then compute $\Phi_1(\mathbf{x}^{\beta})$ in no more than $M_{\max} n$ steps. To see why, consider the expression

$$\sum_{i=1}^{n} \beta_i^{\delta} w_{ii} x_i^{\beta}$$

that corresponds to $\Phi_1(\mathbf{x}^{\beta})$ in equation 4.2. By our assumption, x_i^{β} is nonzero for at most $M_{\max}$ values of the variable i. Assuming nothing about the density of $\mathbf{E}$, β_i^{δ} has at most n nonzero values for each fixed i.

Space complexity Next, I will show that a space-optimal algorithm for casting the coordinates of a document vector $\mathbf{x}$ to an orthonormal basis will occupy no more than $M_{\max}^2 + M_{\max}(1+C) + n$ units of space. To see why, consider again that $\mathbf{x}'^{\delta'}$ has $M_{\max}^2$ nonzero coordinates. By using an appropriate data structure to represent $\mathbf{x}'$, e.g. a list of nonzero coordinates in the basis δ', the resulting vector will occupy precisely $M_{\max}^2$ units of space. The list representing the vector $\mathbf{x}$ will then occupy at most $M_{\max}$ units of space, the array representing the matrix $\mathbf{W}$ will occupy n units of space, and the sparse matrix $\mathbf{S}$ will occupy at most nC units of space.

If we assume again that nothing is known about the density of $\mathbf{E}$, then a space-optimal algorithm computing $\mathbf{E}$ occupies at most $n(C+n)$ units of space. To see why, consider that the Cholesky algorithm is computed in-place, the sparse matrix $\mathbf{S}$ occupies at most nC units of space and the resulting matrix $\mathbf{E}$ occupies at most n^2 units of space. For a document vector $\mathbf{x}$, a space-optimal algorithm computing $\Phi_1(\mathbf{x}^\beta)$ then occupies at most $n^2 + 2n$ units of space. To see why, consider that the matrix $\mathbf{E}$ will occupy at most n^2 units of space, the array representing the matrix $\mathbf{W}$ will occupy at most n units of space, and the resulting vector coordinates will occupy at most n units of space. Notice that although Φ_1 does not lead to an increase of dimensionality like Φ_2 does, it can instead lead to an arbitrary increase of density.

Assuming that $M_{\max}$ is arbitrary, i.e. at most n, we get a worst-case time and space complexity of $\Theta(n^2)$. As I argued in the previous section, such space complexity is practically intractable outside distributed computing. Let us assume again that $M_{\max}$ is a constant. Under this assumption, the worst-case time complexity of computing the inner product is $\Theta(1)$ and the worst-case space complexity is $\Theta(n)$.

As a remark, note that although we assumed in the above analysis that nothing is known about the density of $\mathbf{E}$, this does not mean that we cannot find a more compact representation of $\mathbf{E}$. Given a sparse matrix $\mathbf{S}$, we can generally find a permutation matrix $\mathbf{P}$, such that the Cholesky factor $\mathbf{L}$, where $\mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{P}\mathbf{S}\mathbf{P}^\mathsf{T}$, is also sparse. Using basic facts about matrix transposition, we can derive $\mathbf{E} = \mathbf{P}^{-1}\mathbf{L}$ as follows:

$$\mathbf{S} = \mathbf{P}^{-1}\mathbf{L}\mathbf{L}^\mathsf{T}(\mathbf{P}^\mathsf{T})^{-1} = \mathbf{P}^{-1}\mathbf{L}\mathbf{L}^\mathsf{T}(\mathbf{P}^{-1})^\mathsf{T} = \mathbf{E}\mathbf{E}^\mathsf{T}.$$

This allows us to store a permutation matrix $\mathbf{P}$ and a sparse matrix $\mathbf{L}$ instead of an arbitrarily dense matrix $\mathbf{E}$. However, the existence of

such a permutation matrix $\mathbf{P}$ is only guaranteed for matrices $\mathbf{S}$ of a specific structure, and even if such a $\mathbf{P}$ exists, the problem of finding it is NP-complete [28], so we will generally only find a suboptimal $\mathbf{P}$.

4.5 Similarity matrices

In this section, I construct several matrices $\mathbf{S}$ that model different facets of term similarity. Although it is possible to explicitly construct a change-of-basis matrix $\mathbf{E}$, it is perhaps easier to construct the matrix $\mathbf{S}$ of inner products between the basis vectors. Since we assume the basis vectors of basis γ are normalized, this is equivalent to constructing a matrix $\mathbf{S}$ of cosines of the angles between the basis vectors.

If we expect to be computing $\mathbf{E}$ from $\mathbf{S}$, then we require $\mathbf{S}$ to be both positive-definite and Hermitian. Hermicity comes down to simple symmetry in $\mathbb{R}^n$ and as such, it is easily satisfied; all the matrices shown in this section are Hermitian. One approach to satisfying positive-definiteness is to satisfy a stronger condition and build a strictly diagonally dominant matrix $\mathbf{S}$. In the context of section 4.4, this can be easily satisfied by placing only $C = 1$ nonzero non-diagonal elements into each row of $\mathbf{S}$. If a non-diagonal element has precisely the value -1 or 1, and would therefore violate the strict diagonal dominance of $\mathbf{S}$, we will retain only the diagonal element for the corresponding row. All the matrices $\mathbf{S}$ shown in this section can be sparsified to this form.

If we expect to be computing only the inner product between two normalized vectors (i.e. the soft cosine similarity), then we require only that $\mathbf{x} \neq \mathbf{0} \implies \|\mathbf{x}\| \in \mathbb{R}^+$, i.e. that $\mathbf{x} \neq \mathbf{0} \implies (\mathbf{x}^\gamma)^\mathsf{T} \mathbf{S} \mathbf{x}^\gamma > 0$. If we required this for any $\mathbf{x} \in \mathbb{R}^n$, then this would be equivalent to the requirement that $\mathbf{S}$ is positive definite. However, since we know that the coordinates $\mathbf{x}^\gamma$ correspond to term frequencies, which are non-negative, it is sufficient to require that $\mathbf{S}$ is non-negative as well, i.e. that $\forall i, j = 1, \ldots, n : s_{ij} \geq 0$. Like hermicity, this condition is easy to satisfy, and all the matrices $\mathbf{S}$ shown in this section are non-negative. Note that by using a matrix $\mathbf{S}$ that is not positive definite, we are violating the original assumption that γ is a basis. Nevertheless, these matrices may still be practically useful, as evidenced by their use in the work of Charlet; Damnati [7].

Lastly, if we expect to be computing only the inner product and only between two non-normalized vectors, then $\mathbf{S}$ can be arbitrary.

4.5.1 Spelling

One facet of term similarity that we might want to model is spelling. Intuitively, a two document that are identical except for a single misspelled term should be also considered nearly identical by our model. We may use edit distance to formalize this intuition. Charlet; Damnati [7] propose the following measure:

$$s_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } \theta_1 \left(1 - \frac{\text{edit}(i,j)}{\max(b_i,b_j)}\right)^{\theta_2} \leq \theta_3, \text{ and} \\ \theta_1 \left(1 - \frac{\text{edit}(i,j)}{\max(b_i,b_j)}\right)^{\theta_2} & \text{otherwise,} \end{cases} \tag{4.5}$$

where $\text{edit}(i, j)$ is the edit distance between the terms i and j, b_i and b_j are the byte lengths of the terms i, and $\boldsymbol{\theta}$ are parameters whose optimal values on the subtask B dev dataset are $\theta_1 = 1.8, \theta_2 = 5$, and $\theta_3 = 0$ according to the authors. I will refer to the resulting matrix as $\mathbf{S}_{\text{lev}}$ for ease of reference.

Additionally, I introduce a pruning parameter θ_4 and preemptively assign $s_{ij} = 0$ to all terms i and j such that $\max(b_i, b_j) / \min(b_i, b_j) > \theta_4$. Intuitively, terms with disproportionate lengths are unlikely to be misspellings of one another, so this initial pruning saves us the time that we would otherwise spend computing the edit distance between distant terms. Due to the time complexity of edit distance, this parameter was not exhaustively optimized; instead, the value $\theta_4 = 1.5$ was chosen.

In the context of section 4.4, this formula provides no efficient procedure to sparsity the matrix $\mathbf{S}_{\text{lev}}$. For each term i, we may compute the value s_{ij} for every other term j that survived the initial pruning and then assign $s_{ij} \leftarrow 0$ for all but the C highest values of s_{ij}. Charlet; Damnati [7] reported using only $C = 100$ highest values of s_{ij}.

4.5.2 Synonymy

Another facet of term similarity is synonymy. By building a term co-occurence model of a dataset, we can derive an unsupervised mea-

sure of synonymy by considering terms occuring in similar contexts more synonymous than terms occuring in different contexts. Charlet; Damnati [7] propose the following measure:

$$s_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } \langle \mathbf{v}_i / \|\mathbf{v}_i\|, \mathbf{v}_j / \|\mathbf{v}_j\| \rangle_{\mathbb{X}} \leq \theta_3, \text{and} \\ \langle \mathbf{v}_i / \|\mathbf{v}_i\|, \mathbf{v}_j / \|\mathbf{v}_j\| \rangle_{\mathbb{X}}^{\theta_5} & \text{otherwise,} \end{cases} \quad (4.6)$$

where $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{X}$ are the embeddings of the terms i and j in the inner-product space $\mathbb{X}$ produced by a term embedding model such as Word2vec [26], GloVe [29], or FastText [30], $\langle \cdot, \cdot \rangle_{\mathbb{X}}$ is the inner product on $\mathbb{X}$, and $\boldsymbol{\theta}$ are parameters whose optimal values on the subtask B dev dataset are $\theta_3 = 0$, and $\theta_5 = 2$ according to the authors. I will refer to the resulting matrix as $\mathbf{S}_{\text{rel}}$ for ease of reference.

In the context of section 4.4, this formula provides an opportunity to sparsify $\mathbf{S}_{\text{rel}}$ efficiently. For each term i, we assign $s_{ij} \leftarrow 0$ for all but the C nearest neighbors $\mathbf{v}_j$ of $\mathbf{v}_i$ in $\mathbb{X}$. Charlet; Damnati [7] reported retrieving only $C = 100$ nearest neighbors in their experiments.

Higher values of the parameter θ_3 decrease the density of $\mathbf{S}_{\text{rel}}$. We may also tweak the parameter `min_count` available in the C and Python implementations of Word2vec; the default value of this parameter is five. This parameter causes only those terms that occur sufficiently often to be embedded in $\mathbb{X}$. If we consider the inner product to be 0 for terms missing from $\mathbb{X}$, then higher values of this parameter decrease the density of $\mathbf{S}_{\text{rel}}$ as well. Note, however, that the decrease in density produced by these parameters depends on the dataset and no improved computational complexity is guaranteed.

4.5.3 Ensembles

Unsupervised We can combine a set of unsupervised measures $\{f_i\}_i$ modeling diferent facets of term similarity using an operator $\oplus$. Several choices of $\oplus$, such as the (weighted) average, min, max, or percentiles immediately spring to mind. Note that if each measure f_i produces at most C_i nonzero non-diagonal elements, the measure $s_{ij} = \oplus_k f_k(i, j)$ may produce at most $\sum_i C_i$ nonzero non-diagonal elements. Additional sparsification of the resulting matrix $\mathbf{S}$ may therefore be necessary to achieve the desired complexity. In my experiments, I evaluated the quality of the model using the matrix $\mathbf{S}_{\text{avg}} = \text{avg}(\mathbf{S}_{\text{rel}}, \mathbf{S}_{\text{lev}})$.

Supervised Using a lexical database such as WordNet, we may derive several ground truth measures of term relatedness [31, 32, 33, 34, 35]. Using unsupervised measures of term similarity, such as the edit distance and the inner product of the inner-product space $\mathbb{X}$, we may then fit a regressor for a ground truth measure of our choice using all the unsupervised measures as the training sample. The regressor then gives us a single supervised measure s_{ij} that will provide an estimate of the similarity of terms i and j outside WordNet.

4.6 System description

In this section, I describe several approaches to implementing the extended model into existing search engines. Considered are both inverted-index-based search engines and vector databases.

4.6.1 Inverted index

Inverted-index-based search engines such as Apache Lucene [13] and its distributed cousin ElasticSearch form the backbone of today's large-scale information retrieval systems. In this section, I consider changing both the search engines themselves and also the client applications when the search engines are off-limits.

Direct support Unlike vector databases, inverted-index-based search engines are built around a data structure called the *inverted index*, which maps each term in our vocabulary to a list of documents containing the term; these documents are sorted by a common criterion. In their most rudimentary form described by Manning et al. [27], these search engines split each incoming text query into terms, retrieve the sorted document lists corresponding to the individual query terms, and then traverse the lists, computing the similarity between the query and each individual candidate document until the lists have been exhausted, an allotted time quantum has been exceeded, a sufficient number of documents has been processed, etc.

Assuming we have access to the internals of our search engine, and the search engine uses the standard model for computing the similarity between a query and a document (and not a different model such as

the probabilistic Okapi BM25 model briefly mentioned in chapter 3), we may directly replace the inner product formula for orthogonal bases ($[\mathbf{x}^\delta]^\mathsf{T}\mathbf{y}^\delta$) with the inner product formula for non-orthogonal bases ($[\mathbf{x}^\gamma]^\mathsf{T}\mathbf{S}[\mathbf{y}^\gamma]$) from the equation 4.3.

A close examination reveals that after this straightforward change, the system will still only retrieve documents that have at least one term in common with the query, wasting much of the opportunity provided by the extended standard model. A better approach would be to first expand the query vector $\mathbf{x}$ by computing $(\mathbf{x}^\gamma)^\mathsf{T}\mathbf{S}$ and retrieving document lists for all terms corresponding to the nonzero coordinates in the result. As discussed in section 4.4, this will only lead to a constant increase in the number of document lists assuming that C and $M_{\max}$ are constants.

Query expansion If we don't have access to the internals of our search engine and all we can do is alter the text of the query before the client software submits it to the search engine, we then can use a *query expansion* technique.

In the client software, we will include a vocabulary of terms, and matrices $\mathbf{W}$ and $\mathbf{S}$ corresponding to the terms in the vocabulary. In the ideal case, the vocabulary would include all the terms indexed by the search engine, but in practice, we will probably use a vocabulary extracted from a general-purpose corpus such as the Brown Corpus, or Wikipedia, that best models our document collection in terms of the vocabulary. If we use the corpus not only to build a vocabulary, but also to construct the matrices $\mathbf{S}$ and $\mathbf{W}$, e.g. to construct $\mathbf{W}$ by counting the number of document in the corpus containing a term and to construct $\mathbf{S}_{\text{rel}}$ by building a term embedding model, then the term distribution in the corpus should match our document collection as well.

When the user submits a text query, we might use our vocabulary to transform the query to a vector $\mathbf{x}$, compute $\mathbf{W}^{-1}(\mathbf{W}\mathbf{x}^\beta)^\mathsf{T}\mathbf{S}$, and transform the nonzero coordinates in the result back to a text query. As an example, I will expand the example document

$$d_2 = \text{“I did enact Julius Caesar: I was killed i' the Capitol”},$$

using the three-million term vocabulary and the matrix $\mathbf{S}_{\text{rel}}$ extracted from the Google News Word2vec model distributed with the C

implementation of Word2vec, and the example matrix $\mathbf{W}$ given in equation 4.1. The parameters $\theta_3 = 0, \theta_4 = 2$, and $C = 100$ suggested by Charlet; Damnati [7] were used when building the matrix $\mathbf{S}$. By rounding the resulting coordinates down and transforming them back to a text query, I obtained the following result:

$d_3 =$ "Give␣unto␣Caesar Brutus␣Cassius choreographers␣Bosco Julius␣Caeser therefore␣unto␣Caesar Marcus␣Antonius Caesarion Gallic␣Wars Marcus␣Crassus Antoninus Catiline Seleucus Gaius␣Julius␣Caesar Theodoric Marcus␣Tullius␣Cicero unto␣Caesar emperor␣Nero Herod␣Antipas Titus␣Pullo Julius␣Ceasar Et␣tu␣Brute Tyrannus Render␣unto␣Caesar Crassus render␣unto␣Caesar Agrippina Commodus Antiochus Agrippa Caeser Roman␣emperor Ramses Agamemnon Ceasar Claudius Brutus Julius␣Caesar Pharaoh **Caesar** Napoleon Roman Nii␣Teiko R.␣Nasso Cleavon Zadock Carmy Chibuike Egharevba Aloysious Ajene Adonijah Ndubisi Luckson Chibuzo Lyonel Montay Okoya Shenika Lavel Arrie Bethuel Osbert Authur Olando Elija Geremy Gershom Alvan Darnel Leron Ekow Meshack Emmanual Antione Winfred Theophilus Cedrick Jerald Quintin Thaddeus Derick Wilfred Errol Kwame Deon Josiah Fredrick Simeon Cyril Horace Demetrius Reginald Virgil Marlon Reuben Hubert Willy Elijah Cedric Darius Kelvin **Julius** Anton Nathaniel Jeremiah Leroy Geoffrey Desmond Emmanuel Darryl Ernest Ernest Edwin Alfred Isaac Calvin Marvin Jerome Maurice Rodney Phillip Antonio Gerald Harold Willie Andre Ronald Samuel Benjamin Kenneth Philip Marcus Arthur Carl Fred Edward Jonathan Eric Frank Anthony William Richard Robert **enact Capitol killed** Ididn't honestly myself **I I** my we **the** 'd 'm **did was**",

where the terms that were already present in d_2 are highlighted in bold. Note that since the term i' is missing from our vocabulary, it is missing from d_3 as well, although it may be present in the vocabulary of the search engine. In general, it makes more sense to compute $\mathbf{W}^{-1}(\mathbf{W}\mathbf{x}^{\beta})^{\mathsf{T}}\mathbf{S} - \mathbf{x}^{\beta}$ and add the nonzero coordinates in the result to the original query. In this way, no terms will be removed.

Suppose the search engine uses cosine similarity (i.e. the cosine of the angle between the two vectors under the incorrect assumptions that the coordinates of the vectors are given in an orthogonal basis) to compute the similarity between a query and a document. Then the final similarity score between the query vector $\mathbf{x}$ and the document vector $\mathbf{y}$ comes down to

$$\frac{\mathbf{W}_2\lfloor\mathbf{W}_1^{-1}(\mathbf{W}_1\mathbf{x}^\beta)^\mathsf{T}\mathbf{S}\rceil\mathbf{W}_2\mathbf{y}^\beta}{\sqrt{\lfloor\mathbf{W}_1^{-1}(\mathbf{W}_1\mathbf{x}^\beta)^\mathsf{T}\mathbf{S}\rceil^\mathsf{T}\lfloor\mathbf{W}_1^{-1}(\mathbf{W}_1\mathbf{x}^\beta)^\mathsf{T}\mathbf{S}\rceil}\sqrt{(\mathbf{W}_2\mathbf{y}^\beta)^\mathsf{T}\mathbf{W}_2\mathbf{y}^\beta}},$$

where $\mathbf{W}_1$ is the matrix $\mathbf{W}$ in the client software, and $\mathbf{W}_2$ is the matrix $\mathbf{W}$ in the search engine. If we suppose $\mathbf{W}_1 = \mathbf{W}_2 = \mathbf{W}$ and that we can provide non-integral term frequencies in our text query (a function provided by e.g. Apache Lucene) and can thus avoid rounding, then the above equation becomes

$$\frac{(\mathbf{x}^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}^\gamma}{\sqrt{((\mathbf{x}^\gamma)^\mathsf{T}\mathbf{S})^\mathsf{T}(\mathbf{x}^\gamma)^\mathsf{T}\mathbf{S}}\sqrt{(\mathbf{y}^\gamma)^\mathsf{T}\mathbf{y}^\gamma}}.$$

Since the query vector $\mathbf{x}$ stays constant during the retrieval, the above equation produces the same ordering of document vectors $\mathbf{y}$ as

$$\frac{(\mathbf{x}^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}^\gamma}{\sqrt{(\mathbf{x}^\gamma)^\mathsf{T}\mathbf{x}^\gamma}\sqrt{(\mathbf{y}^\gamma)^\mathsf{T}\mathbf{y}^\gamma}}. \tag{4.7}$$

which is equivalent to the inner product between $\mathbf{x}$ and $\mathbf{y}$ divided by the cosine similarity normalization factor. For ease of reference, I will call this the *hard normalization* as opposed to the *soft normalization* in the soft cosine similarity formula from equation 4.3.

If we additionally assume that the search engine performs no normalization and computes only the algebraic dot product between the coordinates of the query and document vectors $\mathbf{x}$ and $\mathbf{y}$ (i.e. the inner product under the incorrect assumptions that the coordinates are expressed in an orthogonal basis), then the above equation becomes

$$(\mathbf{x}^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}^\gamma$$

which corresponds to the actual inner product between $\mathbf{x}$ and $\mathbf{y}$.

In my experiments, I evaluated the quality of the model using hard normalization with various corpora.

4.6.2 Vector database

Inverted-index-based search engines work directly with text documents. Vector databases such as Faiss [36] make no such assumption and perform the more general task of storing and retrieving vectors. Note that unlike the inverted-index-based engines, which generally automatically compute the matrix $\mathbf{W}$ and perform the conversion from the basis β to the basis γ for us, we will need to store vectors expressed in the basis γ already with vector databases.

In this section, I outline two approaches to storing document vectors in a vector database that supports the retrieval of the document vectors closest to a query vector according to the cosine similarity, or the algebraic dot product. We have recently suggested [4] to use inverted-index-based engines as vector databases; in the light of this research, techniques described in this section can be directly adapted to inverted-index-based engines.

Orthonormalization Given a query vector $\mathbf{x}$ and a document vector $\mathbf{y}$, we can express both $\mathbf{x}$ and $\mathbf{y}$ in an orthonormal basis using the maps Φ_1 and Φ_2 described in section 4.4.2. In the case of Φ_1, we need to compute and store the matrix $\mathbf{E}$ and there is no upper limit to the density of the resulting vectors. In the case of Φ_2, the resulting vectors are sparse, but quadratic in their dimensionality. Nevertheless, the cosine similarity between the resulting vectors corresponds to the soft cosine similarity, and the algebraic dot product between the resulting vectors corresponds to the actual inner product.

Partial application Given a query vector $\mathbf{x}$ and a document vector $\mathbf{y}$, let us first assume that the vector database supports the retrieval of document vectors closest to $\mathbf{x}$ according to the algebraic dot product. Then let $\mathbf{x}'$ be the original query vector and $\mathbf{y}'$ the original document vector. Let us first construct $\mathbf{x}$, and $\mathbf{y}$ as follows:

$$\mathbf{x}^\gamma = \mathbf{x}'^\gamma, \text{ and } \mathbf{y}^\gamma = \frac{\mathbf{S}\mathbf{y}'^\gamma}{\sqrt{(\mathbf{y}'^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}'^\gamma}}.$$

Then the algebraic dot product between $\mathbf{x}^\gamma$ and $\mathbf{y}^\gamma$ comes down to

$$\frac{(\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}'^\gamma}{\sqrt{(\mathbf{y}'^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}'^\gamma}}.$$

Since the query vector $\mathbf{x}'$ stays constant during the retrieval, the above equation produces the same ordering of document vectors $\mathbf{y}'$ as the soft cosine similarity formula from equation 4.3.

Let us now construct $\mathbf{x}$, and $\mathbf{y}$ as follows:

$$\mathbf{x}^\gamma = \mathbf{S}^\mathsf{T}\mathbf{x}'^\gamma, \text{ and } \mathbf{y}^\gamma = \mathbf{y}'^\gamma.$$

Then the algebraic dot product between $\mathbf{x}^\gamma$ and $\mathbf{y}^\gamma$ comes down to the inner product between $\mathbf{x}'$ and $\mathbf{y}'$. Note that in this case, I was able to remove the matrix $\mathbf{S}$ from the document vectors $\mathbf{y}$, which allows us to update $\mathbf{S}$ over time without reindexing the document collection.

Let us now assume that the vector database supports the retrieval of document vectors closest to $\mathbf{x}$ not according to the algebraic dot product, but according to the cosine similarity. Then let $\mathbf{x}'$ be the original query vector and $\mathbf{y}'$ be the original document vector. Then we will construct $\mathbf{x}$ and $\mathbf{y}$ expressed in a non-orthogonal normalized basis γ' of $\mathbb{R}^{n+1}$ as follows:

$$\mathbf{x}^{\gamma'} = \begin{bmatrix} \frac{\mathbf{S}^\mathsf{T}\mathbf{x}'^\gamma}{\sqrt{((\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S})^\mathsf{T}(\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S}}} \\ 0 \end{bmatrix}, \text{ and } \mathbf{y}^{\gamma'} = \begin{bmatrix} \mathbf{z}^\gamma \\ \sqrt{1-(\mathbf{z}^\gamma)^\mathsf{T}\mathbf{z}^\gamma} \end{bmatrix},$$

where $\mathbf{z}^\gamma = \mathbf{y}'^\gamma / \sqrt{(\mathbf{y}'^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}'^\gamma}$. As shown by Neyshabur; Srebro [37, sec. 4.3], both $\mathbf{x}$ and $\mathbf{y}$ are normalized, and therefore the cosine similarity between $\mathbf{x}^{\gamma'}$ and $\mathbf{y}^{\gamma'}$ comes down to the algebraic dot product:

$$\begin{aligned}(\mathbf{x}^{\gamma'})^\mathsf{T}\mathbf{y}^{\gamma'} &= \left(\frac{\mathbf{S}^\mathsf{T}\mathbf{x}'^\gamma}{\sqrt{((\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S})^\mathsf{T}(\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S}}}\right)^\mathsf{T} \frac{\mathbf{y}'^\gamma}{\sqrt{(\mathbf{y}'^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}'^\gamma}} = \\ &= \frac{(\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}'^\gamma}{\sqrt{((\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S})^\mathsf{T}(\mathbf{x}'^\gamma)^\mathsf{T}\mathbf{S}}\sqrt{(\mathbf{y}'^\gamma)^\mathsf{T}\mathbf{S}\mathbf{y}'^\gamma}}\end{aligned}$$

Since the query vector $\mathbf{x}'$ stays constant during the retrieval, the above equation produces the same ordering of document vectors $\mathbf{y}'$ as the soft cosine similarity formula from equation 4.3.

Unless we apply the map Φ_1, the document vectors that we store in the vector database are going to be sparse, but high-dimensional regardless of our approach. However, most vector databases are designed to store dense low-dimensional vectors instead. By computing a low-rank approximation of the term-document matrix given in an orthonormal basis using e.g. the latent semantic analysis (LSA), we can derive a mapping to a lower-dimensional vector space.

In my experiments, I evaluated the quality of the model using just the inner product between the original vectors $\mathbf{x}'$ and $\mathbf{y}'$.

4.7 Experimental setup

In this section, I give an overview of the experiments performed on the QA datasets described in chapter 2.

4.7.1 Language modeling

The texts in the datasets were lower-cased and tokenized on white spaces and punctuation. Tokens shorter than two characters or longer than 15 characters were removed and so were English stopwords. Images and URLs were replaced with special tokens. Notice that I used a slightly different model compared to section 3.4.1; this was in order to more closely mimic the preprocessing steps taken by Charlet; Damnati [7].

By default, I used the unannotated subtask A datasets to built the matrix $\mathbf{W}$, where $w_{ii} = \ln \frac{N}{n_i}$, N is the total number of documents in the collection, and n_i is the number of documents that contain the term i. I then used the soft cosine from equation 4.3 as a measure of similarity between two documents.

4.7.2 MAP-density plots

As a part of my experiment, I constructed the matrices $\mathbf{S}_{\text{lev}}$ and $\mathbf{S}_{\text{rel}}$ using a range of parameters described in section 4.5. For the matrix $\mathbf{S}_{\text{lev}}$, I varied the parameters $C \in \{1, 10, 100, 1000\}$ and $\theta_3 \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. For the matrix $\mathbf{S}_{\text{rel}}$, I varied the parameters $C \in \{1, 10, 100, 1000, 10000\}$, `min_count` $\in \{0, 5, 50, 500, 5000\}$, and $\theta_3 \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. For both matrices, I only varied a single

parameter and kept the remaining parameters at their default values. I then valuated each resulting model on the subtask B dev and test datasets, and I plotted the matrix densities against the achieved MAP scores to show the trade-off between the sparsity of $\mathbf{S}$ and the model quality.

4.7.3 Other configurations

Apart from the models constructed for the MAP-density plots, I also evaluated a number of different models on the subtask B test datasets.

In the context of section 4.5, I constructed a matrix $\mathbf{S}_{\text{avg}}$ using the default parameters. In the context of section 4.6.1, I constructed a vocabulary and the matrices $\mathbf{S}_{\text{rel}}$ and $\mathbf{W}$ from each of the following term embedding models:

w2v.ql the unannotated subtask A datasets, where the Word2vec model for matrix $\mathbf{S}$ was built using the continuous bag of words architecture,

w2v.googlenews the Google News Word2vec model distributed with the C implementation of Word2vec[3],

glove.enwiki_gigaword5 a GloVe model trained on the English Wikipedia and the Gigaword 5 dataset[4],

glove.common_crawn a GloVe model trained on the Common Crawl dataset[5], and

fasttext.enwiki a FastText model trained on the English Wikipedia[6].

I then used the soft cosine with hard normalization as a measure of similarity between two documents. In the context of section 4.6.2, I used the inner product as a measure of similarity between two documents.

As a baseline, I used the cosine as a measure of similarity between two documents, which directly corresponds to the standard model. I also list the subtask B winners and baselines.

---

3. https://code.google.com/archive/p/word2vec/
4. https://nlp.stanford.edu/data/glove.6B.zip
5. https://nlp.stanford.edu/data/glove.840B.300d.zip
6. https://github.com/facebookresearch/fastText

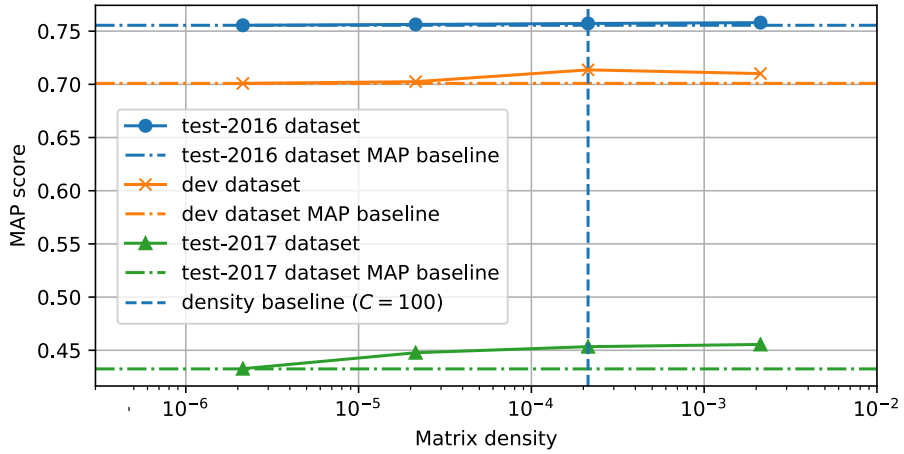

Figure 4.1: The MAP score plotted against the density of the matrix $\mathbf{S}_{\text{lev}}$ as we increase the parameter C from the value of 1 (leftmost) to the value of 1,000 (rightmost). For every dataset, the baseline MAP score corresponds to using the cosine as a measure of similarity between two documents. Baseline density corresponds to the default value of C.

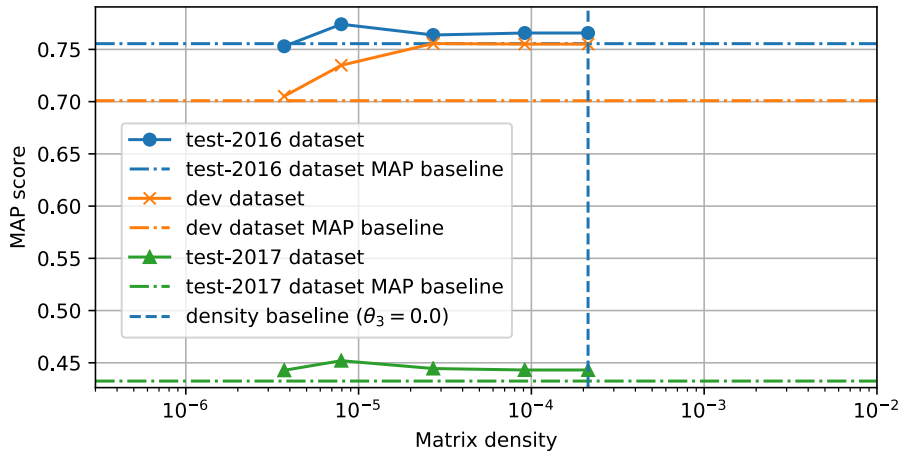

Figure 4.2: The MAP score plotted against the density of the matrix $\mathbf{S}_{\text{lev}}$ as we increase the parameter θ_3 from the value of 0.8 (leftmost) to the value of 0 (rightmost). For every dataset, the baseline MAP score corresponds to using the cosine as a measure of similarity between two documents. Baseline density corresponds to the default value of θ_3.

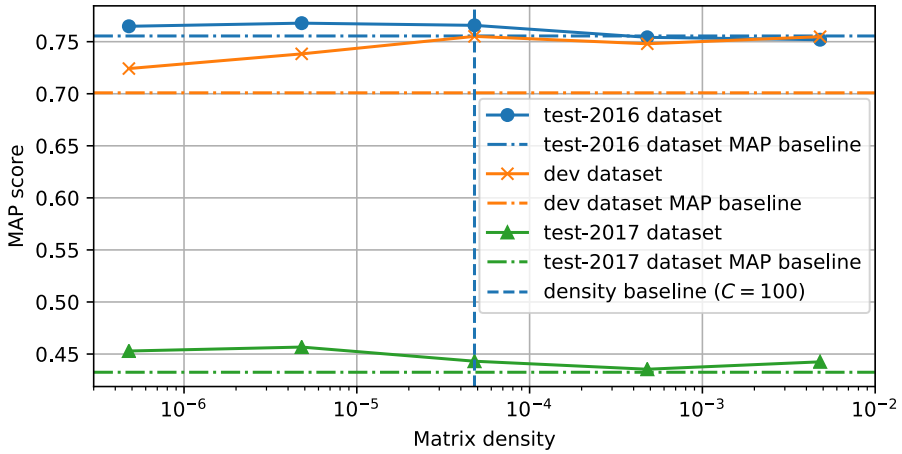

Figure 4.3: The MAP score plotted against the density of the matrix $\mathbf{S}_{\text{rel}}$ as we increase the parameter C from the value of 1 (leftmost) to the value of 10,000 (rightmost). For every dataset, the baseline MAP score corresponds to using the cosine as a measure of similarity between two documents. Baseline density corresponds to the default value of C.

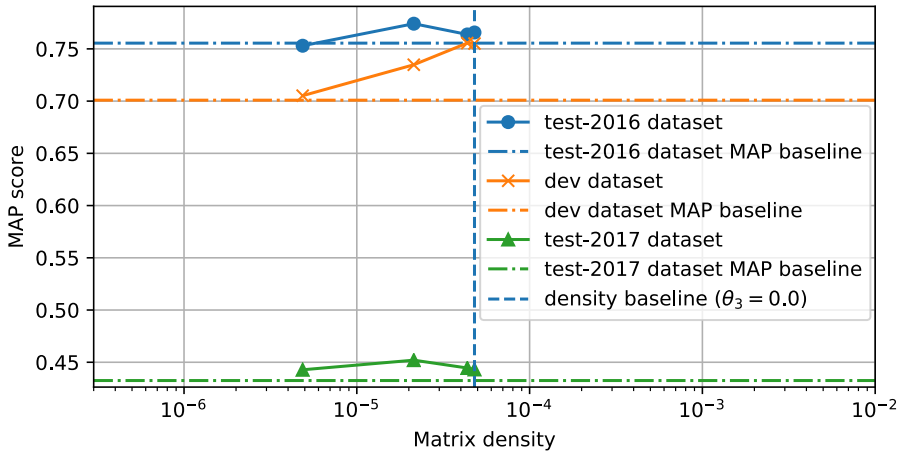

Figure 4.4: The MAP score plotted against the density of the matrix $\mathbf{S}_{\text{rel}}$ as we increase the parameter θ_3 from the value of 0.8 (leftmost) to the value of 0.2 (rightmost). For every dataset, the baseline MAP score corresponds to using the cosine as a measure of similarity between two documents. Baseline density corresponds to the default value of θ_3.

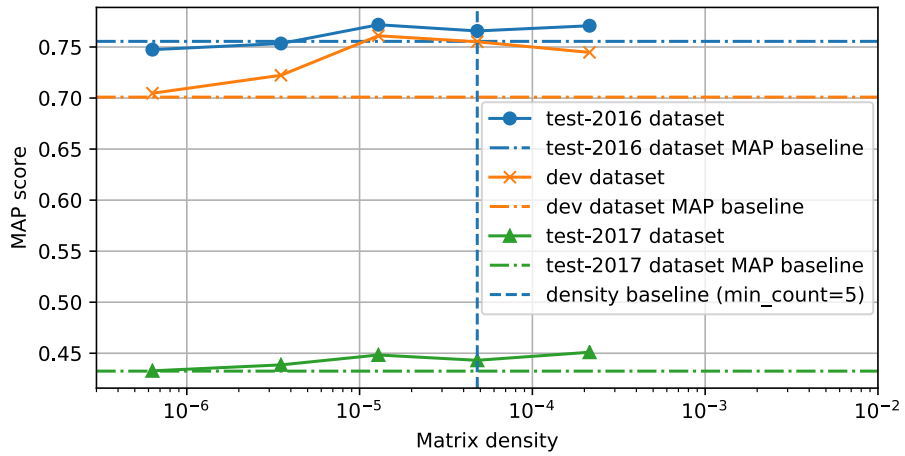

Figure 4.5: The MAP score plotted against the density of the matrix $\mathbf{S}_{\text{rel}}$ as we increase the parameter `min_count` from the value of 5,000 (leftmost) to the value of 0 (rightmost). For every dataset, the baseline MAP score corresponds to using the cosine as a measure of similarity between two documents. Baseline density corresponds to the default value of `min_count`.

4.8 Results

By looking at MAP-density plots of matrices $\mathbf{S}_{\text{lev}}$ and $\mathbf{S}_{\text{rel}}$ in figures 4.1–4.5, several patterns emerge. Adjusting the parameter C leads to higher MAP scores that adjusting the parameter θ_3 does. Increasing the density of $\mathbf{S}$ leads to small or no increase in MAP, which suggests that the extended model quickly reaches a saturation point after which additional density of $\mathbf{S}$ does not lead to an improvement in quality.

Looking at the individual figures, figure 4.2 shows that with the matrix $\mathbf{S}_{\text{lev}}$, the elements with a value of s_{ij} below 0.4 are of limited usefulness to the model. Reintroducing these elements increases the density, but leads to no improvement in the MAP score as evidenced by the flat right tails of the plotted line segments. Figure 4.3 shows that the subtask B dev and test datasets behave very differently with respect to the parameter C for the matrix $\mathbf{S}_{\text{rel}}$. Whereas the test datasets seem to benefit from the sparsification of $\mathbf{S}$, the dev dataset follows a reverse trend and consistently benefits from the increased density of $\mathbf{S}$. Figure 4.4 shows that with the matrix $\mathbf{S}_{\text{rel}}$, all $C = 100$ nearest

neighbors j of each term i receive a value of s_{ij} above 0.2 and lower values of the parameter θ_3 are therefore ineffective in further increasing the density of $\mathbf{S}$.

Considering the results in tables 4.1 and 4.2, the highest rank across both subtask B test datasets was achieved by the ensemble matrix $\mathbf{S}_{\text{avg}}$. The matrices $\mathbf{S}_{\text{lev}}$ and $\mathbf{S}_{\text{rel}}$ using default parameters steadily outperformed the standard model.

The results for the models using soft cosine with hard normalization vary wildly between the datasets; the most stable results were achieved with the Google News Word2vec term embedding model, which stays within a 1.2 MAP score radius around the matrices $\mathbf{S}_{\text{lev}}$ and $\mathbf{S}_{\text{rel}}$ using default parameters. This shows that the query expansion approach described in section 4.6.1 is in general not inferior to soft cosine similarity with soft normalization and can be used to implement the extended model in a way that allows us to update $\mathbf{S}$ over time without reindexing the document collection. Using just the inner product instead of soft cosine similarity in the context of section 4.6.2 shows promise on the 2016 test dataset, but performs poorly in the 2017 test dataset.

4.9 Conclusion and future work

In this chapter, I developed an extension to the standard model for modeling synonymy and I showed that is outperforms the standard model on the text similarity task in the context of QA. Importantly, I also showed that, unlike other models such as LSA, a close approximation of the described model can be implemented in a way that allows us to update $\mathbf{S}$ over time without reindexing the document collection. I also developed several matrices that model different facets of term similarity; this is in contrast to models such as LSA, which are confined to a single measure of similarity such as term co-occurences.

The discussion in section 4.4 showed that little is known about the density of the change-of-basis matrix and of the vectors that were cast to the orthonormal basis δ. In section 4.5, I introduced the notion of supervised ensemble term similarity matrices, but no concrete implementation was described and evaluated. These are both fruitful venues for future work.

Matrix **S**	Term embedding model	Normalization	Measure	C	θ_3	`min_count`	MAP
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.6	5	77.41
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	50	77.18
$\mathbf{S}_{\text{rel}}$	w2v.ql	hard	soft cosine	100	0	5	77.10
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	0	77.08
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	10	0	5	76.78
SemEval-2016 task 3 subtask B winner (UH-PRHLT-primary)				—	—	—	**76.70**
$\mathbf{S}_{\text{avg}}$	w2v.ql	soft	soft cosine	100	0	5	76.61
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.2	5	76.57
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	5	76.57
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	1	0	5	76.48
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.4	5	76.38
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.4	—	75.88
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.6	—	75.81
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	1000	0	—	75.80
$\mathbf{S}_{\text{rel}}$	w2v.ql	—	inner product	100	0	5	75.79
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.2	—	75.72
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0	—	75.72
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.8	—	75.67
$\mathbf{S}_{\text{rel}}$	w2v.googlenews	hard	soft cosine	100	0	5	75.64
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	10	0	—	75.63
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	1	0	—	75.55
—	—	—	**cosine**	—	—	—	**75.55**
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	1000	0	5	75.42
$\mathbf{S}_{\text{rel}}$	glove.common_crawl	hard	soft cosine	100	0	5	75.42
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	500	75.34
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.8	5	75.30
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	10000	0	5	75.19
SemEval-2016 task 3 subtask B IR baseline			—	—	—	—	**74.75**
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	5000	74.74
$\mathbf{S}_{\text{rel}}$	fasttext.enwiki	hard	soft cosine	100	0	5	74.73
$\mathbf{S}_{\text{rel}}$	glove.enwiki_gigaword5	hard	soft cosine	100	0	5	72.78

Table 4.1: Results for all configurations on the 2016 subtask B test dataset. Baselines are highlighted in bold.

Matrix **S**	Term embedding model	Normalization	Measure	C	θ_3	`min_count`	MAP
SemEval-2017 task 3 subtask B winner (SimBow-primary)				—	—	—	**47.22**
$\mathbf{S}_{\text{avg}}$	w2v.ql	soft	soft cosine	100	0	5	46.04
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	10	0	5	45.67
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	1000	0	—	45.55
$\mathbf{S}_{\text{rel}}$	glove.common_crawl	hard	soft cosine	100	0	5	45.49
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.6	—	45.35
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.0	—	45.34
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.2	—	45.31
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	1	0	5	45.29
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.4	—	45.22
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.6	5	45.20
$\mathbf{S}_{\text{rel}}$	glove.enwiki_gigaword5	hard	soft cosine	100	0	5	45.13
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	0	45.10
$\mathbf{S}_{\text{rel}}$	fasttext.enwiki	hard	soft cosine	100	0	5	44.94
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	50	44.83
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	10	0	—	44.77
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	100	0.8	—	44.65
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.4	5	44.45
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.2	5	44.31
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	5	44.31
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0.8	5	44.28
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	10000	0	5	44.25
$\mathbf{S}_{\text{rel}}$	w2v.googlenews	hard	soft cosine	100	0	5	44.13
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	500	43.86
$\mathbf{S}_{\text{rel}}$	w2v.ql	hard	soft cosine	100	0	5	43.76
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	1000	0	5	43.54
$\mathbf{S}_{\text{rel}}$	w2v.ql	soft	soft cosine	100	0	5000	43.28
$\mathbf{S}_{\text{lev}}$	—	soft	soft cosine	1	0	—	43.27
—	—	—	**cosine**	—	—	—	**43.25**
$\mathbf{S}_{\text{rel}}$	w2v.ql	—	inner product	100	0	5	42.17
SemEval-2017 task 3 subtask B IR baseline			—	—	—	—	**41.85**

Table 4.2: Results for all configurations on the 2017 subtask B test dataset. Baselines are highlighted in bold.

5 Conclusion

In this work, I developed two models based on the standard vector space bag-of-words model. Both models were described in mathematical terms, reviewed from the implementation standpoint, and evaluated on the SemEval-2016 and 2017 task 3 datasets.

The first model exploits the fact that modern text retrieval systems employ text segmentation during the indexing of documents. Using this knowledge, the model achieves state-of-the-art results on our datasets.

The second model captures the similarity between different terms. Different facets of synonymy can be modeled and changes to the similarity relations do not require the reindexing of a document collection. The model surpasses the standard vector space bag-of-words model on our datasets.

Since both models are mutually independent and each exploits a distinct concept, they can be both implemented in a single IR system.

Acknowledgments

TAR Omega grant TD03000295 is gratefully acknowledged.

Bibliography

1. GANTZ, John; REINSEL, David. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*. 2012, vol. 2007, no. 2012, pp. 1–16.
2. SHILAKES, Christopher C; TYLMAN, Julie. Enterprise information portals. *Merrill Lynch*. 1998, vol. 16.
3. MOOERS, Calvin. Information retrieval viewed as temporal signaling. In: *Proceedings of the international congress of mathematicians*. 1950, vol. 1, pp. 572–573.
4. RYGL, Jan; POMIKÁLEK, Jan; EHEK, Radim; RIKA, Michal; NOVOTNÝ, Vít; SOJKA, Petr. Semantic Vector Encoding and Similarity Search Using Fulltext Search Engines. In: *Proc. of the 2nd Workshop on Representation Learning for NLP, RepL4NLP 2017*. Vancouver, Canada: ACL, 2017, pp. 81–90. Available from DOI: 10.18653/v1/W17-2611.
5. RYGL, Jan; SOJKA, Petr; RIKA, Michal; EHEK, Radim. ScaleText: The Design of a Scalable, Adaptable and User-Friendly Document System for Similarity Searches: Digging for Nuggets of Wisdom in Text. In: HORÁK, Ale; RYCHLÝ, Pavel; RAMBOUSEK, Adam (eds.). *Proc. of the 10th Workshop on Recent Advances in Slavonic NLP, RASLAN 2016*. Brno: Tribun EU, 2016, pp. 79–87. ISBN 978-80-263-1095-2. Available also from: https://nlp.fi.muni.cz/raslan/2016/paper08-Rygl_Sojka_etal.pdf.
6. SALTON, Gerard; WONG, Anita; YANG, Chung-Shu. A vector space model for automatic indexing. *Communications of the ACM*. 1975, vol. 18, no. 11, pp. 613–620. Available also from: http://www-lb.cs.umd.edu/class/fall2009/cmsc828r/PAPERS/VSM_salton-2.pdf.
7. CHARLET, Delphine; DAMNATI, Geraldine. SimBow at SemEval-2017 Task 3: Soft-Cosine Semantic Similarity between Questions for Community Question Answering. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Compu-

tational Linguistics, 2017, pp. 315–319. Available from DOI: 10.18653/v1/S17-2051.

8. TANGE, O. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine*. 2011, vol. 36, no. 1, pp. 42–47. Available from DOI: http://dx.doi.org/10.5281/zenodo.16303.

9. EHEK, Radim; SOJKA, Petr. Software Framework for Topic Modelling with Large Corpora. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, 2010, pp. 45–50. http://is.muni.cz/publication/884893/en.

10. JONES, Eric; OLIPHANT, Travis; PETERSON, Pearu, et al. *SciPy: Open source scientific tools for Python* [online]. 2001– [visited on 2017-12-12]. Available from: http://www.scipy.org/.

11. NAKOV, Preslav; MÀRQUEZ, Lluís; MOSCHITTI, Alessandro; MAGDY, Walid; MUBARAK, Hamdy; FREIHAT, abed Alhakim; GLASS, Jim; RANDEREE, Bilal. SemEval-2016 Task 3: Community Question Answering. In: *Proc. of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: ACL, 2016, pp. 525–545. Available from DOI: 10.18653/v1/S16-1083.

12. NAKOV, Preslav; HOOGEVEEN, Doris; MÀRQUEZ, Lluís; MOSCHITTI, Alessandro; MUBARAK, Hamdy; BALDWIN, Timothy; VERSPOOR, Karin. SemEval-2017 Task 3: Community Question Answering. In: *Proc. of the 11th International Workshop on Semantic Evaluation*. Vancouver, Canada: ACL, 2017, pp. 27–48. SemEval '17.

13. BIAECKI, Andrzej; MUIR, Robert; INGERSOLL, Grant; IMAGINATION, Lucid. Apache Lucene 4. In: *SIGIR 2012 workshop on open source information retrieval*. 2012, p. 17.

14. SALTON, Gerard. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall, 1971.

15. SALTON, Gerard; BUCKLEY, Chris. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*. 1988, vol. 24, pp. 513–523.

16. SINGHAL, Amit; BUCKLEY, Chris; MITRA, Mandar; SALTON, Gerard. *Pivoted Document Length Normalization*. Ithaca, NY, 1995. Available also from: `http://singhal.info/pivoted-dln.pdf`. Technical report. Cornell University.

17. MURATA, Masaki; MA, Qing; UCHIMOTO, Kiyotaka; OZAKU, Hiromi; UTIYAMA, Masao; ISAHARA, Hitoshi. Japanese probabilistic information retrieval using location and category information. In: *International Workshop on Information Retrieval With Asian Languages*. 2000, pp. 81–88. Available also from: `https://doi.org/10.1145/355214.355226`.

18. ROBERTSON, Stephen E.; JONES, Karen Spärck. Relevance weighting of search terms. *JASIS*. 1976, vol. 27, no. 3, pp. 129–146. Available also from: `http://www.staff.city.ac.uk/~sb317/papers/RSJ76.pdf`.

19. ROBERTSON, Stephen; WALKER, S.; JONES, S.; HANCOCK-BEAULIEU, M. M.; GATFORD, M. Okapi at TREC2. In: *The 2nd TREC (TREC2)*. Gaithersburg, MD: NIST, 1994, pp. 21–34. Available also from: `http://kak.tx0.org/Information-Retrieval/.files/1994-TREC2-OKAPI.pdf`.

20. ROBERTSON, Stephen; WALKER, S.; JONES, S.; HANCOCK-BEAULIEU, M. M.; GATFORD, M. Okapi at TREC3. In: *Overview of the Third TREC (TREC3)*. Gaithersburg, MD: NIST, 1995, pp. 109–126. Available also from: `http://research.microsoft.com/pubs/67649/okapi_trec3.pdf`.

21. SINGHAL, Amit. Modern Information Retrieval: A Brief Overview. *IEEE Data Eng. Bull.* 2001, vol. 24, pp. 35–43. Available also from: `http://singhal.info/ieee2001.pdf`.

22. MASS, Yosi; MANDELBROD, Matan; AMITAY, Einat; CARMEL, David; MAAREK, Yoëlle S; SOFFER, Aya. JuruXML An XML retrieval system at INEX'02. In: *INEX Workshop*. 2002, pp. 73–80.

23. KOCZ, Aleksander; PRABAKARMURTHI, Vidya; KALITA, Jugal. Summarization as feature selection for text categorization. In: *Proceedings of the ACM CIKM Conference*. ACM, 2000, pp. 365–370.

24. KO, Youngjoong; PARK, Jinwoo; SEO, Jungyun. Improving text categorization using the importance of sentences. *IP&M*. 2004, vol. 40, no. 1, pp. 65–79. Available also from: https://dl.acm.org/citation.cfm?id=975971.

25. SIDOROV, Grigori; GELBUKH, Alexander; GÓMEZ-ADORNO, Helena; PINTO, David. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*. 2014, vol. 18, no. 3, pp. 491–504.

26. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 2013.

27. MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

28. YANNAKAKIS, Mihalis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*. 1981, vol. 2, no. 1, pp. 77–79.

29. PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. GloVe: Global Vectors for Word Representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. Available also from: http://www.aclweb.org/anthology/D14-1162.

30. BOJANOWSKI, Piotr; GRAVE, Edouard; JOULIN, Armand; MIKOLOV, Tomas. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*. 2016.

31. RESNIK, Philip. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., 1995, pp. 448–453. IJCAI'95. ISBN 1-55860-363-8, 978-1-558-60363-9. Available also from: http://dl.acm.org/citation.cfm?id=1625855.1625914.

32. LIN, Dekang. An Information-Theoretic Definition of Similarity. In: *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 296–304. ICML '98. ISBN 1-55860-556-8. Available also from: `http://dl.acm.org/citation.cfm?id=645527.657297`.

33. JIANG, Jay J.; CONRATH, David W. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *CoRR*. 1997, vol. cmp-lg/9709008. Available also from: `http://arxiv.org/abs/cmp-lg/9709008`.

34. LEACOCK, Claudia; CHODOROW, Martin. Combining Local Context and WordNet Similarity for Word Sense Identification. In: FELLBAUM, Christiane (ed.). *WordNet: An electronic lexical database.* MIT Press, 1998, chap. 13, pp. 265–283. ISBN 978-0262061971.

35. WU, Zhibiao; PALMER, Martha. Verbs Semantics and Lexical Selection. In: *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*. Las Cruces, New Mexico: Association for Computational Linguistics, 1994, pp. 133–138. ACL '94. Available from DOI: `10.3115/981732.981751`.

36. JOHNSON, Jeff; DOUZE, Matthijs; JÉGOU, Hervé. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*. 2017.

37. NEYSHABUR, Behnam; SREBRO, Nathan. On Symmetric and Asymmetric LSHs for Inner Product Search. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. Lille, France: JMLR.org, 2015, pp. 1926–1934. ICML'15. Available also from: `http://dl.acm.org/citation.cfm?id=3045118.3045323`.

38. ZEZULA, Pavel; AMATO, Giuseppe; DOHNAL, Vlastislav; BATKO, Michal. *Similarity Search: The Metric Space Approach*. Springer, 2006. Advances in Database Systems. ISBN 0-387-29146-6.

Index