A Quantitative Evaluation of the Enhanced Topic-Based Vector Space Model

Artem Polyvyanyy, Dominik Kuropka

Technische Berichte Nr. 19

des Hasso-Plattner-Instituts für Softwaresystemtechnik
an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik
an der Universität Potsdam

Nr. 19

A Quantitative Evaluation of the Enhanced Topic-Based Vector Space Model

Artem Polyvyanyy, Dominik Kuropka

Potsdam 2007

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar

Die Reihe *Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam* erscheint aperiodisch.

Herausgeber: Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Redaktion: Artem Polyvyanyy, Dominik Kuropka

E-mail: {artem.polyvyanyy; dominik.kuropka}@.hpi.uni-potsdam.de

Vertrieb: Universitätsverlag Potsdam
Am Neuen Palais 10
14469 Potsdam
Fon +49 (0) 331 977 4517
Fax +49 (0) 331 977 4625
e-mail: ubpub@uni-potsdamde
http://info.ub.uni-potsdam.de/verlag.htm

Druck: allprintmedia gmbH
Blomberger Weg 6a
13437 Berlin
email: info@allprint-media.de

Heft Nr 19 (2007)
ISBN 978-3-939469-95-7
ISSN 1613-5652

Abstract

This contribution presents a quantitative evaluation procedure for Information Retrieval models and the results of this procedure applied on the enhanced Topic-based Vector Space Model (eTVSM). Since the eTVSM is an ontology-based model, its effectiveness heavily depends on the quality of the underlaying ontology. Therefore the model has been tested with different ontologies to evaluate the impact of those ontologies on the effectiveness of the eTVSM. On the highest level of abstraction, the following results have been observed during our evaluation: First, the theoretically deduced statement that the eTVSM has a similar effecitivity like the classic Vector Space Model if a trivial ontology (every term is a concept and it is independet of any other concepts) is used has been approved. Second, we were able to show that the effectiveness of the eTVSM raises if an ontology is used which is only able to resolve synonyms. We were able to derive such kind of ontology automatically from the WordNet ontology. Third, we observed that more powerful ontologies automatically derived from the WordNet, dramatically dropped the effectiveness of the eTVSM model even clearly below the effectiveness level of the Vector Space Model. Fourth, we were able to show that a manually created and optimized ontology is able to raise the effectiveness of the eTVSM to a level which is clearly above the best effectiveness levels we have found in the literature for the Latent Semantic Index model with compareable document sets.

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
BPT	Business Process Technology
BSD	Berkeley Software Distribution
e.g.	*exempli gratia*—Latin for “for the sake of example”
eEPC	extended Event Process Chain
ERD	Entity-Relationship Diagram
etc.	*et cetera*—Latin for “and the others”
eTVSM	enhanced Topic-based Vector Space Model
HPI	Hasso-Plattner-Institute
ID	IDentifier
IF&IR	Information Filtering and Information Retrieval
IF	Information Filtering
iid	independent and identically-distributed, in respect to random variables
IR	Information Retrieval
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
NIST	National Institute of Standards and Technology
OS	Operating System
PL	Procedural Language
PL/SQL	Procedural Language / Structured Query Language
PL/pgSQL	Procedural Language / PostgreSQL Structured Query Language
SQL	Structured Query Language
SVD	Singular Value Decomposition
tf-idf	term frequency-inverse document frequency

TREC	Text Retrieval Conference
TVSM	Topic-based Vector Space Model
URL	Uniform Resource Locator
VSM	Vector Space Model
w.r.t.	with respect to

Chapter 1

Introduction

Information Retrieval deals with the following major challenge: given a set of documents and a query, Information Retrieval deals with finding a set of documents relevant to the query. In many Information Retrival approaches a query itself is considered as a (virtual) document. To simplify life even further, we consider only the content part of a document—the plain text of natural language. Document markup or metadata is left out. This is the task and the setup that a natural language Information Retrieval system should normally deal with. The way the system does this work is expressed in the Information Retrieval model. This model can incorporate algorithms from trivial word occurrence frequency lookup, to an extensive statistical analysis or tracking of word relations. However, at the end, this model should get evaluated upon the effectivness of the returned results which itself is a non-algorithmic task. So, we have documents created by humans, humans as the sources of information construct queries and humans who at the end judge on the results. On the other hand, we want to derive a concrete system that is capable of solving this problem systematically.

The goal of this work is, to contribute to the global knowledge of the solution of Information Retrieval problem as it was presented before. To look for the new ways to obtain more accurate results produced by the Information Retrieval models on a large scale of use cases. To evaluate how semantic relations can be applied for the tasks of this class and how introduction of semantics in Information Retrieval is compared to former known statistical approaches. Another challenge is to check how is it possible to tune results by changing model configuration, and finally: What degree of configuration freedom does the model provide to a user?

The enhanced Topic-based Vector Space Model (eTVSM) was selected as the object for study and evaluation. The eTVSM is an advanced Information Retrieval model that integrates stemming and stopword removal and can represent most of the linguistic phenomena. It provides a great deal of configuration freedom through considering ontology concepts and their relationships to encode linguistic phenomena. On the other hand, the eTVSM model has a set of heuristic points in its end-configuration which greatly influence model output. These points will be identified and their influence will be studied. The main goal of the eTVSM model is to "understand" document content to the level which is encoded in the ontology. A modeler goal is to understand how to encode a domain ontology in an efficient way. Our goal is to optimize both of these processes.

Finally, as the end point artifacts of this work we can see an implementation of the eTVSM which will be used to perform evaluations. Afterwards, the evaluation

results and the comparisons to other models will be presented and grounded. As a side-effect, we can identify some optimization approaches for model implementation as well as model feasibility judgments for large scale databases retrieval and its adoptability characteristics to a dynamic environment of document collection content on the one hand and model configuration on the other.

The work reported here can be split into two parts. In the first part we present theoretical bases required in order to perform eTVSM evaluations and comparisons. While in the second part we actually present practical steps that were taken in order to obtain evaluation results. We present you these evaluation results and justify on their comparisons.

We will start this contribution by presenting Information Retrieval models that led to the idea of creating eTVSM—the target Information Retrieval model of our evaluations. Therefore, in chapter 2 you will find descriptions of the classic Vector Space Model (VSM) and the Topic-based Vector Space Model (TVSM) along with some terms and definitions that are common in Information Retrieval field and also in our work. Chapter 3 is completely devoted to eTVSM. Here, you will find the description of the ontology eTVSM operates upon, and approaches to its modeling. Also, we will present formalism of obtaining eTVSM document models and document similarities. Afterwards, in chapter 3, we will present heuristic assumptions incorporated in eTVSM which might influence the results of our further evaluations. In chapter 4 we concentrate our attention on measurements that are commonly used to measure Information Retrieval model effectiveness. These will be the measurements we will obtain for each our model evaluation. Also, some basic ideas on these measurements comparisons and their possible graphical representations in a form of plots are presented. In chapter 5 we define an evaluation design for performing an evaluation of an Information Retrieval system. Furthermore, we talk about existing test collections suitable for performing evaluations. Following the description of the preparation phase, we describe our experimental design. The analysis phase of the overall proposed evaluation design consists of two presented approaches for models effectiveness comparisons, through statistical aggregation of observed measurements, with an additional significance study. Chapter 5 constitutes the theoretical part of our work.

Furthermore, in chapter 6 we present you with Themis—the Information Retrieval framework which was developed in order to perform eTVSM evaluations. It includes an eTVSM implementation and components required for the model configuration, carrying out evaluations and collecting and aggregating measurements. Chapter 7 then presents evaluation results that were obtained for various eTVSM configurations. Obtained results get compared with evaluations of other Information Retrieval models. Finally, in chapter 8 we summarize our experience obtained in the course of this work with conclusions.

This contribution is an extended and reworked version of the master thesis [46] of Artem Polyvyanyy.

Chapter 2

Information Retrieval Models: VSM and TVSM

This chapter provides a brief overview on the background of Information Retrieval, the common terms and definitions used, as well as two selected models will be presented. The two presented models, Vector Space Model (VSM) and Topic-based Vector Space Model (TVSM) are ancestors of the enhanced Topic-based Vector Space Model (eTVSM) which will be studied in more detail in the remaining chapters of this contribution. If you are familiar with Information Retrieval and the VSM as well as the TVSM we recommend you to go directly to chapter 3 where we present the eTVSM in detail.

2.1 Common Terms and Definitions

We will introduce several Information Retrieval models and we aim to present common terms and definitions for all these models. The model specific definitions will be provided later, with corresponding Information Retrieval model descriptions.

Information Retrieval Model

A retrieval model specifies representations used for documents and queries, and how they are compared [60]. An Information Retrieval model is a formalisation of the way of thinking about Information Retrieval. Such formalism can be defined in form of algorithms, mathematical formulas, etc. Formally, an Information Retrieval model is a quadruple $[D, Q, F, R(q_i, d_j)]$ [8] where:

- D, is a set of representations for the documents in the collection;
- Q, is a set of representations for the user information needs (queries);
- F, is a framework for modeling document representations, queries, and their relationships;
- $R(q_i, d_j)$, is a ranking function which associates a real number with a query $q_i, (q_i \in Q)$ and document representation $d_j, (d_j \in D)$.

The ranking function is often called the similarity function, because of its semantic aspect of expressing similarity between a document and a query.

Document Model

A document model is a formal document representation used by an Information Retrieval model to obtain document similarities.

Document Preprocessing

Document preprocessing is a stage before a document model construction, when data is analyzed, transformed and filtered from the document content. This might include stopword removal, stemming, etc.

Stopword Removal

The process of stopword removal deals with filtering out "non-content words" (usually named as stopwords) from the document content, prior construction of the document model. Stopwords are words like *a*, *the*, *is*, etc. So that, at the end, only the content bearing words [51] get represented in the document model. One might track stopwords by analyzing term frequencies. An alternative is the usage of predefined set of stopwords called stop list, a list of words to be filtered out [51, 62]. In general, 40–50% of the total number of words in a document are usually removed with the help of a stop list [51].

Stemming

Stemming is a process of determining a stem form of a given inflected (or, sometimes, derived) word form. Stemmer is a computer algorithm used for a purpose of stemming [19, 30, 42]. The first ever published stemmer was written by Julie Beth Lovins in 1968 [40]. In July of 1980 Martin Porter has published his Porter stemmer [47]. Till now, this algorithm is a de-facto standard algorithm for English language stemming. Over the next few years, he extended his work by building Snowball [48], a framework for writing stemming algorithms, and he implemented an improved English stemmer together with stemmers for several other languages.

Information Retrieval Environment Formalization

Additionally to Information Retrieval model formalization we want to provide an Information Retrieval environment formalism. These are common pieces that are used by model framework to operate on documents and queries. Considering term as the smallest information unit that is distinguishable by Information Retrieval model:

- T, is a set of terms which appear in $D \cup Q$;
- $\alpha_{d,t}$, is the occurence of the term $t \in T$ in the document $d \in D$
  $\alpha_{q,t}$, is the occurence of the term $t \in T$ in the query $q \in Q$;
- $\omega_{d,t}$, is the weight of the term $t \in T$ in the document $d \in D$
  $\omega_{q,t}$, is the weight of the term $t \in T$ in the query $q \in Q$.

2.2 Linguistic Phenomena

Natural language documents are more than just sequences of words. Complex word relations exist between words. These relations are hidden in the word meanings on the semantic level. Consideration of semantics can improve the quality of information

search. A scientific field that encompasses the study of semantics is linguistics. In this section, we present semantic word relations studied by linguistics. Considering these relations should allow Information Retrieval models to operate with meaning and word relations, rather than just operating with term occurrences. Looking at document semantics is the core of the eTVSM.

Synonymy

Synonymy relation between two words exists if they are different words but have similar or identical meanings and are interchangeable [33] in a specific context. Two synonyms for example are *car* and *automobile* which usually are interchangeable in most contexts.

Inflection

Inflection is the modification or marking of a word to reflect information, such as gender, tense, number or person of a target word [59, 61]. An inflectional affix carries certain grammatical restrictions with it. Thus *walking*, *walks*, *walker* have in common the root *walk* and the affixes *-ing*, *-s*, and *-er*.

Composition

Composition is the word forming process where the resulting (formed) word consists of more than one free morpheme. English language examples of composition are *barefoot* and *blackboard*.

Derivation

Derivation is the process of creating new lexemes from other lexemes, e.g., by adding a derivational affix. It is a kind of word formation. Derivational affixes usually apply to words of one syntactic category and change them into words of another syntactic category. E.g., the English derivational suffix *-ly* changes adjectives into adverbs (*quick* $\rightarrow$ *quickly*).

Derivational affixes can also modify the meaning. E.g., the derivational prefix *un-* applies to adjectives (*do* $\rightarrow$ *undo*). Derivation may occur without any change of form, for example *telephone* (noun) and *to telephone* (verb). In such a case it is assumed that null morpheme was affixed.

Hyponymy

According to [25, 28], hyponyms are a set of related words whose meaning are specific instances of a more general word (so, e.g., *red*, *white*, *blue*, etc., are hyponyms of *color*). Hyponymy is thus the relationship between a general term and specific instances of it. E.g., *sedan* as the instance of *car*.

Meronymy

Meronymy denotes a constituent *part of*, or a *member of* relation [14, 15]. That is,

`X` is a meronym of `Y` if `Xs` are parts of `Ys`, or
`X` is a meronym of `Y` if `Xs` are members of `Ys`.

E.g., *room* as the part of an *apartment* as the part of a *floor* as the part of a *building*. A closely related concept is that of mereology [58], which specifically deals with *part/whole* relations and is used in logic. It is formally expressed in terms of first-order logic.

Homography

Homography is a precedent when two words have the same orthography but different interpretations, e.g. *mouse*—rodent, and *mouse*—computer device.

Metonymy

Metonymy is the substitution of one word for another with which it is associated. Metonymy refers to the use of a single characteristic to identify a more complex entity [21, 23]. The following are clear, commonly used examples of metonymy: the *press* for the news media, a *dish* for an entrée.

Word Groups

Sometimes a semantic meaning is encoded not in one separate word, but in a word group, e.g. *New York City*—"*the largest city in New York State and in the United States*". Thus, a semantic unit consists of three separate words. Treating each word separately will in most cases deliver a different semantic interpretation of the word group. Assuming a notion of terms, a word group can be considered as a compound term.

2.3 Vector Space Model

The Vector Space Model (VSM) or term vector model, is an algebraic model used for Information Filtering, Information Retrieval, indexing and relevancy rankings. It represents natural language documents in a formal manner by the use of vectors in a multi-dimensional space which has only positive axis intercepts. It was used for the first time by the SMART Information Retrieval system [13] which was developed at Cornell University in the 1960s. The VSM formal operational procedure can be divided into three stages. The first stage is document indexing. Here content bearing terms are extracted. The second stage deals with weighting of indexed terms. Finally, the third stage is responsible for calculating similarities between the input query and indexed documents.

Document Indexing

Document indexing incorporates document preprocessing which in fact might include stopword removal and, or stemming. Non-linguistic methods for indexing have also been implemented. Probabilistic indexing is based on the assumption that there is some statistical difference in the distribution of content bearing words, and stopwords [62]. Another indexing approach might be indexing method which uses serial clustering of words in text [12].

Term Weighting

The term weighting for the vector space model is handled by statistics. There are three main factors of term weighting: term frequency factor, term collection frequency

factor and document vector length normalization factor. The end term weight might be constructed from all or a subset of mentioned factors. E.g., the inverse document frequency assumes that the importance of a term is proportional with the number of documents the term appears in [51]. Evaluations show that best results w.r.t. recall and precision, are obtained by using weighting schemes that incorporate all three factors [53, 39].

Obtaining Similarities

The document similarity is determined by using associative coefficients based on the inner product of a document vector and a query vector (queries are treated as regular documents), where word overlap indicates similarity. The inner product is usually normalized. In most cases the cosine coefficient, which measures the angle between document vectors is used as the similarity measure. However, other measures are also applicable, e.g. Jaccard and Dice coefficients [52].

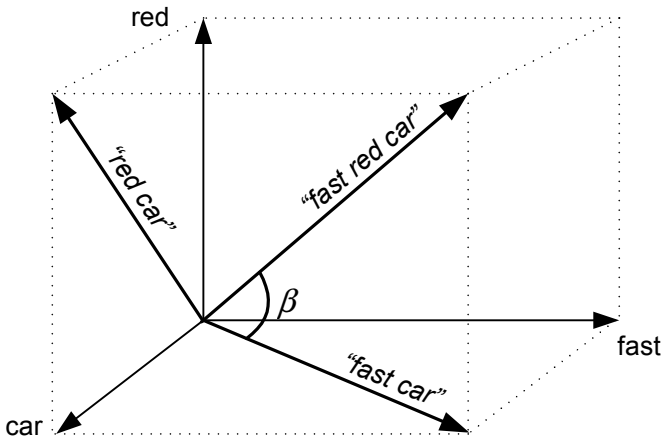

Figure 2.1: Vector space model visualization.

In Figure 2.1 we provide visualization of VSM document models. Here, the multi-dimensional space consists of three dimensions. These dimensions represent *car*, *fast* and *red* terms. Further, document models are constructed as vectors with term weights at respective term dimension places. In our example, document vectors "*fast car*", "*red car*" and "*fast red car*" are visualized. Finally, the level of document similarity is expressed by angles between corresponding document vectors. In our example the β angle defines similarity level between "*fast car*" and "*fast red car*" documents. Further, we want to define VSM similarity between two documents using the formalism defined in section 2.1:

$$\begin{aligned} \vec{d_i} &= \left(\omega_{d_i,t_1}, \omega_{d_i,t_2}, \ldots, \omega_{d_i,t_{\#T}}\right) \\ \vec{d_j} &= \left(\omega_{d_j,t_1}, \omega_{d_j,t_2}, \ldots, \omega_{d_j,t_{\#T}}\right) \\ sim(d_i, d_j) &= \frac{\vec{d_i}\vec{d_j}}{|\vec{d_i}||\vec{d_j}|} \\ &= \frac{\sum_{t \in T} \omega_{d_i,t}\omega_{d_j,t}}{\sqrt{\sum_{t \in T} \omega_{d_i,t}^2}\sqrt{\sum_{t \in T} \omega_{d_j,t}^2}} \end{aligned}$$

The simplest term weighting schema assumes term weights to be equal to the document term occurrences $\omega_{d,t} = \alpha_{d,t}$. The classic vector space model as proposed by Salton, Wong and Yang [54] had both local and global parameters incorporated in the term weight ω equation (known as the *tf-idf* [51, 53]):

$$\omega_{d,t_i} = \frac{\alpha_{d,t_i}}{\max_{t \in T} \alpha_{d,t}} \log \frac{\#D}{\#\{e \in D : \alpha_{e,t_i} > 0\}}$$

tf-idf allows seldom terms to get higher weight, since they are good discriminators, however it requires costly implementation—especially when a document collection is part of a dynamic environment. The VSM has the following limitations:

- Long documents are considered poor representatives of the vector space model because they have poor similarity values (a small scalar product and a large dimensionality);
- Search keywords must precisely match document terms; word substrings might result in "false positive match[1]" It occurs when we are observing a difference in terms, when in truth there is none (e.g., due to the different word inflected forms);
- The search keywords that are typed during the search in an inappropriate manner give poorer results. This can be explained the same way as the previous statement;
- Semantic word relations are not considered. Similarity between terms "car" and "auto" referring to the same class of objects is neglected.

The VSM has been criticized for being ad hoc. However, to the positives of the model one can count its simplicity and intuitive graphical representation of documents and their similarities. For a thorough theoretical analysis of the vector space model please refer to [49].

2.4 Topic-based Vector Space Model

The Topic-based Vector Space Model (TVSM) is a vector-based approach for document comparison. In contrast to the classical VSM the approach proposed by the TVSM does not assume independence between terms by being flexible in definition of

[1] This type of error occurs when we are observing a difference when in truth there is none (or more specifically—no statistically significant difference).

term similarities. The main difference of TVSM as compared to VSM is the operational vector space. TVSM represents documents as vectors in a k dimensional space R which has only positive axis intercepts [11, 36].

$$R = \mathbb{R}^k_{\geq 0} \quad \text{with} \quad k \in \mathbb{N}_{\geq 1}$$

Each dimension of R represents a so-called fundamental topic. These fundamental topics are assumed to be orthogonal and independent from each other.

A term $t_i \in T$, with T beeing the set of all terms, is represented by a term vector $\vec{t_i}$ in the R vector space. Each term vector is assigned a term weight between zero and one. A term vector direction represents term relevance according to fundamental topics, whereas the algebraic term vector length $\left|\vec{t_i}\right|$ corresponds to a term weight which is defined to be not larger than one and not lesser than zero:

$$\begin{aligned} \vec{t_i} &= (t_{i,1}, t_{i,2}, \ldots, t_{i,k}) \\ \left|\vec{t_i}\right| &= \sqrt{t_{i,1}^2 + t_{i,2}^2 + \ldots + t_{i,k}^2} \in [0,1] \end{aligned}$$

A document model of a document $d_j \in D$ is represented in TVSM by a document vector $\vec{d_j} \in R$. Further, the document vector length is normalized to the length of one for further convenient usage (since only the vector direction is of interest):

$$\forall d_j \in D : \vec{d_j} = \frac{1}{\left|\vec{\delta_j}\right|}\vec{\delta_j} \quad \Rightarrow \quad \left|\vec{d_j}\right| = 1 \quad \text{with} \quad \vec{\delta_j} = \sum_{t_i \in T} \omega_{d_j,t_i}\vec{t_i}$$

Here, ω_{d_j,t_i} represents the weight of a fundamental topic t_i in a document d_j. One might apply different weighting schemes, e.g. similar to those described in the context of the VSM (refer section 2.3).

In Figure 2.2 we present TVSM document model visualization for a "*fast red car*" document. The TVSM document model is obtained as a sum of term vectors representing terms in the document.

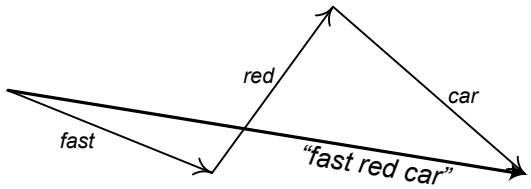

Figure 2.2: TVSM document model visualization.

The similarity between two documents d_i and d_j is defined as the scalar product of document vectors. This is equal to the cosine of the angle β_{d_i,d_j} between document vectors, since document vectors always have the vector lenght of one:

$$\begin{aligned} sim(d_i, d_j) &= \vec{d_i}\vec{d_j} \\ &= \left|\vec{d_i}\right|\left|\vec{d_j}\right|\cos\beta_{d_i,d_j} \\ &= \cos\beta_{d_i,d_j} \end{aligned}$$

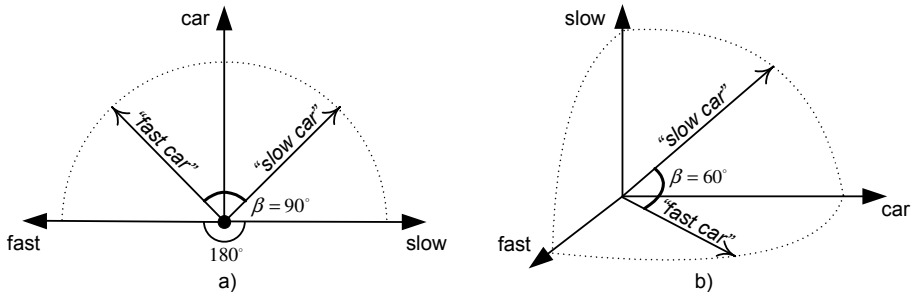

Figure 2.3: TVSM operational vector space construction principles. a) negative and positive axis intercepts b) only positive axis intercepts.

The rationale behind the usage of only positive axis intercepts in the TVSM is shown in the visualization of two possible approaches in Figure 2.3: usage of negative and positive axis intercepts—part *a*, and usage of only positive axis intercepts—part *b*. The part *b* scenario allows only the angles between term vectors from $0°$ to $90°$. This allows term similarity values to stay between zero and one as: $\forall\beta \in [0°, 90°] \rightarrow \cos(\beta) \in [0, 1]$. This allows intuitive understanding of the document similarities as 0—to represent documents being totally not similar and 1—to represent absolutely similar documents. The part *a* scenario, however, allows similarity values to be in the interval $[-1, 1]$ as for: $\forall\beta \in [0°, 180°] \rightarrow \cos(\beta) \in [-1, 1]$. Basically, it is more of a convenience issue, as now, documents "*fast car*" and "*slow car*" from Figure 2.3.a have a similarity of 0, which intuitively says against any similarity between these two documents.

With the knowledge of term vector lengths and angles between them one computes documents similarities as follows:

$$
\begin{aligned}
sim(d_i, d_j) &= \vec{d_i}\vec{d_j} \\
&= \frac{1}{\left|\vec{\delta_i}\right|}\vec{\delta_i}\frac{1}{\left|\vec{\delta_j}\right|}\vec{\delta_j} \\
&= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\vec{\delta_i}\vec{\delta_j} \\
&= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\sum_{t_k \in T}\omega_{d_i,t_k}\vec{t_k}\sum_{t_l \in T}\omega_{d_j,t_l}\vec{t_l} \\
&= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\sum_{t_k \in T}\sum_{t_l \in T}\omega_{d_i,t_k}\omega_{d_j,t_l}\vec{t_k}\vec{t_l}
\end{aligned}
$$

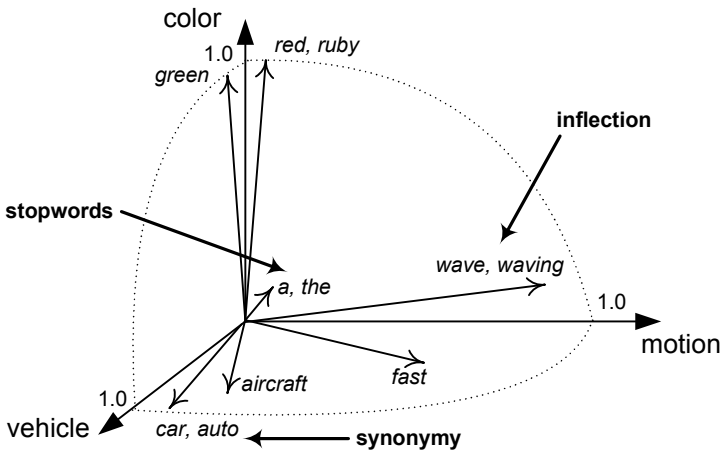

Figure 2.4: Topic-based vector space model visualization.

The length of the unnormed document vector d_i can be computed as follows:

$$\begin{aligned}\left|\vec{\delta}_i\right| &= \left|\sum_{t_k \in T} \omega_{d_i,t_k} \vec{t}_k\right| \\ &= \sqrt{\left|\sum_{t_k \in T} \omega_{d_i,t_k} \vec{t}_k\right|^2} = \sqrt{\left(\sum_{t_k \in T} \omega_{d_i,t_k} \vec{t}_k\right)^2} \\ &= \sqrt{\sum_{t_k \in T} \sum_{t_l \in T} \omega_{d_i,t_k} \omega_{d_i,t_l} \vec{t}_k \vec{t}_l}\end{aligned}$$

In Figure 2.4 we provide visualization of the TVSM operational vector space. Here, the multi-dimensional space consists of three dimensions. But, as opposed to the VSM example presented in Figure 2.1, dimensions are constructed from fundamental topics. In our example these are *vehicle*, *color* and *motion* topics. Afterwards, we visualize one possible term vectors assignment.

By introducing such a topic-grounded vector space the TVSM is capable of expressing synonymy, inflection, composition, derivation, hyponymy and meronymy (see section 2.2) through assigning appropriate angles between term vectors. Furthermore, the TVSM is compatible with stop lists and stemming, which has also been formally shown in [36]:

- Synonymy—high linguistic interdependency, term vector angles tend or equal to $0°$, like with *car* and *auto* term vectors in our example;
- Inflection—maximal linguistic interdependency, term vector angles equal $0°$, like with *wave* and *waving* term vectors in our example;
- Composition—language dependent, e.g., in English—low interdependency, term vector angles tend to $90°$;
- Derivation—high linguistic interdependency, term vector angles tend to $0°$;

- Hyponymy—high interdependency, term vector angles are low, heuristics that assigns vector angles might consider the number of intermediate relation layers;
- Meronymy—high interdependency, term vector angles are low, heuristics that assigns vector angles might consider the number of intermediate relation layers.

The procedure of obtaining term vector lengths and angles for TVSM is not formally defined. One might come up with own feasible approaches like for example presented in [11]. Also, the term vector length concept is left open for own contributions. In general it is recommended to set vector length to one for content bearing terms and to zero for stop list terms, or use approaches which are depending on the inverse term frequency regarding the whole document base. By this, seldom (and therefore well discriminating) terms get a higher weight over frequent terms.

Chapter 3

Enhanced Topic-based Vector Space Model

It was already mentioned in section 2.4 that the TVSM takes the word semantic relations better into account compared to the VSM. However, it still lacks a formal approach for obtaining term vector lengths and inter-term vector angles. Further, it is not possible to express such linguistic phenomena like homography, metonymy and word groups (see section 2.2) in the TVSM. The enhanced Topic-based Vector Space Model (eTVSM) aims to close this gap. This is implemented through obtaining document similarities not on the basis of term similarities, but on the basis of term interpretations. The operational vector space in the eTVSM is similar to the vector space of the TVSM, however document models are now constructed from interpretation vectors. Interpretation vectors in turn are constructed following a predefined formal procedure. The eTVSM operates with the following concepts: word, word stem, term, interpretation and topic. Term relations are expressed in an ontology which operates with term, interpretation and topic concepts. The eTVSM was first proposed in [36]. This chapter is devoted to the eTVSM formalism.

3.1 eTVSM Ontology

The eTVSM operational vector space has to be derived from a domain ontology which is responsible to hold information about relationships between various concepts of a domain. In computer science, an ontology is a data model (data structure) that represents a domain and is used to reason about objects in that domain and relations between them [24]. To construct an ontology which will represent word relations, the eTVSM uses concepts of terms, interpretations and topics. These concepts are organized in a hierarchical, non-cyclic directed graph [27] structure. Edges of this graph aim to specify semantic relations of concepts of the same class, as well as inter-conceptual semantic relations.

3.1.1 Topic Map

Just as in the case with the TVSM (see section 2.4) the operational vector space dimensionality of the eTVSM is defined by topics. However, the difference is that topic vectors now do not need to be orthogonal. Angles between topic vectors are defined by

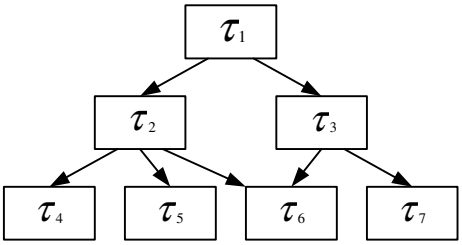

Figure 3.1: Abstract topic map example.

topic concepts relation level defined in ontology. The topic relations are expressed in a topic map, which is a part of a domain ontology used by the eTVSM. A Topic map is a directed graph with topics as nodes. The graph edges assign super-, sub-topic relations. The only constraint on the topic map structure is that directed edges should not form cycles. The edges are not typed. Therefore, one is free to define any kind of relations by the edges. Such relations might be: *is a*, *part of* or *member of* relation. Angles between topic vectors are, therefore, only dependent on topic map configuration, but not on the type of edges used.

In Figure 3.1 an abstract topic map example is shown which will be used here to illustrate computation of topic similarities which are taken up by the eTVSM for the calculation of term and document similarities. A super-topic can relate to an arbitrary number of sub-topics. Moreover, two different topics can share a common sub-topic. Such a graph, which consists of all connected topics represents a domain where all topics will gain similarity level based on the amount of intermediate topics in the topic map structure. Therefore, it is feasible to model several domains by introducing several topic map graph-like structures that are not connected. Topics from one of such disconnected domain will be orthogonal (topic similarity will be equal to zero) to all the topics from the other domains. In general, it is possible to specify topic map data structure as it is shown in Figure 3.2 provided in Entity-Relationship-Diagram notation (see [17]).

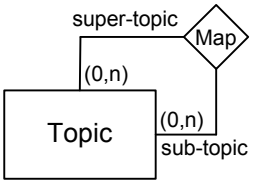

Figure 3.2: Topic map data structure.

Obtaining Topic Similarities

The process of obtaining topic similarities can be subdivided into two routines. First, eTVSM proposes formal procedure for gaining topic vectors from the structure of a topic map. This heuristic process is proposed in [36]. Afterwards, topic similarities are obtained as scalar product of topic vectors. The operational vector space dimensional-

ity is specified by the number of topics in the system.

Being t the number of topics in the topic map, a set of all topics can be given as $\Theta = \{\tau_1, \tau_2, \ldots, \tau_t\}$. In order to represent the topic map structure we can use a super-topic relation $S(\tau_i) \subseteq (\Theta \setminus \tau_i)$. By defining the set $S(\tau_i)$ for each topic we completely define structure of the topic map. Returning to the abstract topic map example from Figure 3.1, the super-topic relations are defined as follows:

$$\begin{aligned} S(\tau_1) &= \{\} \\ S(\tau_2) &= \{\tau_1\} \\ S(\tau_3) &= \{\tau_1\} \\ S(\tau_4) &= \{\tau_2\} \\ S(\tau_5) &= \{\tau_2\} \\ S(\tau_6) &= \{\tau_2, \tau_3\} \\ S(\tau_7) &= \{\tau_3\} \end{aligned}$$

The super-topic relation allows us to construct more complex relations, such as a p-level super-topic relation. This transitive relation provides super-topics that are p levels above the target topic.

$$\begin{aligned} S^p(\tau_i) &= S(\tau_i) && \text{for} \quad p = 1 \\ S^p(\tau_i) &= \textstyle\bigcup_{\tau_k \in S^{p-1}(\tau_i)} S(\tau_k) && \text{for} \quad p > 1 \end{aligned}$$

To obtain all super-topics of the target topic we can use an unbound transitive super-topic relation:

$$S^*(\tau_i) = S^1(\tau_i) \cup S^2(\tau_i) \cup S^3(\tau_i) \cup \ldots$$

Again, returning to the abstract topic map example from Figure 3.1 the unbound transitive super-topic relations are defined as follows:

$$\begin{aligned} S^*(\tau_1) &= \{\} \\ S^*(\tau_2) &= \{\tau_1\} \\ S^*(\tau_3) &= \{\tau_1\} \\ S^*(\tau_4) &= \{\tau_1, \tau_2\} \\ S^*(\tau_5) &= \{\tau_1, \tau_2\} \\ S^*(\tau_6) &= \{\tau_1, \tau_2, \tau_3\} \\ S^*(\tau_7) &= \{\tau_1, \tau_3\} \end{aligned}$$

A set of leaf topics Θ_L contains all topics that are not included into any super-topic relation of any topic from a topic map which means that they don't have any sub-topics:

$$\Theta_L = \{\tau_i \in \Theta : \neg\exists\ \tau_k \in \Theta \ \text{ with } \ \tau_i \in S(\tau_k)\}$$

In our abstract topic map example, the leaf topics set consists of:

$$\Theta_L = \{\tau_4, \tau_5, \tau_6, \tau_7\}$$

Respectively, complementary to the Θ_L is a set of internal topic nodes Θ_N. Θ_N includes topics that have at least one sub-topic:

$$\Theta_N = \complement\Theta_L = \Theta \setminus \Theta_L$$

In our abstract example, the internal topics set consists of:

$$\Theta_N = \{\tau_1, \tau_2, \tau_3\}$$

As we have stated before, a topic vector $\vec{\tau_i} = (\tau_{i,1}, \tau_{i,2}, \ldots \tau_{i,t}) \in \mathbb{R}^t$ is assigned to each topic from a topic map. The approach for gaining topic vectors separates two formal procedures for leaf topics Θ_L and internal topics Θ_N as proposed in [36]. In case of leaf topics, topic vectors are obtained as:

$$\forall \tau_i \in \Theta_L : \vec{\tau_i} = \left|\left(\tau^*_{i,1}, \tau^*_{i,2}, \ldots, \tau^*_{i,t}\right)\right|$$

$$\text{with} \quad \tau^*_{i,k} = \begin{cases} 1 & \text{if} \quad \tau_k \in S^*(\tau_i) \vee i = k \\ 0 & \text{else} \end{cases}$$

While in case of internal topic nodes, topic vectors are obtained as:

$$\forall \tau_i \in \Theta_N : \vec{\tau_i} = \left| \sum_{\tau_s \in \Theta : \tau_i \in S(\tau_s)} \vec{\tau_s} \right|$$

The heuristics behind this approach is that leaf nodes are considered as main building blocks. Leaf topics are seen as the specific part of their super-topics. That is why, all dimensions of the vector that correspond to the set $S^*(\tau_i) \cup \tau_i$ acquire same value of one. Then, topic vectors for topics from set Θ_N are obtained as the sum of its direct children. The logic behind is that internal topic node describes all topics that are its direct children. Also, by normalizing vector length to one, the weight of each topic in topic vector becomes dependent on the number of intermediate topics in a topic map structure. The normalization is also performed because it is only important to know the direction (angles) of topic vectors, therefore after normalization it holds:

$$\forall \tau_i \in \Theta : |\vec{\tau_i}| = 1$$

Thus, the formal procedure for obtaining topic vectors is as follows. First, leaf topic vectors are calculated (including normalization). Then we are able to calculate all super-topic vectors of leaf topics, and so on up to the root topic node(s). A recursive approach to implement this procedure is appropriate. Further, we provide topic vectors for topics from our abstract topic map example from Figure 3.1:

$$\begin{aligned}
\vec{\tau_1} &= (0.669, 0.495, 0.429, 0.120, 0.120, 0.255, 0.174) \\
\vec{\tau_2} &= (0.642, 0.642, 0.194, 0.224, 0.224, 0.194, 0) \\
\vec{\tau_3} &= (0.607, 0.282, 0.607, 0, 0, 0.282, 0.325) \\
\vec{\tau_4} &= \left(1/\sqrt{3}, 1/\sqrt{3}, 0, 1/\sqrt{3}, 0, 0, 0\right) \\
\vec{\tau_5} &= \left(1/\sqrt{3}, 1/\sqrt{3}, 0, 0, 1/\sqrt{3}, 0, 0\right) \\
\vec{\tau_6} &= (1/2, 1/2, 1/2, 0, 0, 1/2, 0) \\
\vec{\tau_7} &= \left(1/\sqrt{3}, 0, 1/\sqrt{3}, 0, 0, 0, 1/\sqrt{3}\right)
\end{aligned}$$

Once we have all topic vectors we can obtain topic similarity of two topics as the scalar product of corresponding topic vectors. Because topic vectors are normalized, the scalar product is equal to the cosine of the angle β_{ij} between topic vectors:

$$\begin{aligned} sim(\tau_i, \tau_j) &= \vec{\tau_i}\vec{\tau_j} \\ &= \sum_{k=1}^{t} \tau_{i,k}\tau_{j,k} \\ &= \cos\beta_{ij} \end{aligned}$$

Finally, all pair-wise topic similarities from the abstract topic map example from Figure 3.1 are given in Table 3.1:

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
τ_1	1.000	0.933	0.933	0.741	0.741	0.924	0.734
τ_2	0.933	1.000	0.742	0.871	0.871	0.836	0.483
τ_3	0.933	0.742	1.000	0.513	0.513	0.888	0.888
τ_4	0.741	0.871	0.513	1.000	0.667	0.577	0.333
τ_5	0.741	0.871	0.513	0.667	1.000	0.577	0.333
τ_6	0.924	0.836	0.888	0.577	0.577	1.000	0.577
τ_7	0.734	0.483	0.888	0.333	0.333	0.577	1.000

Table 3.1: Topic similarities for the abstract topic map example.

3.1.2 Interpretations

In the eTVSM interpretations are intermediate links between topics and terms. Conceptually, interpretations play the role of semantic terms. By introducing intermediate concept you as the modeler of a domain ontology receive more freedom and opportunities to express linguistic phenomena. It is possible to map two terms *car* and *auto* to the same interpretation to express total synonymy relation; more complex structures are discussed in detail in [36].

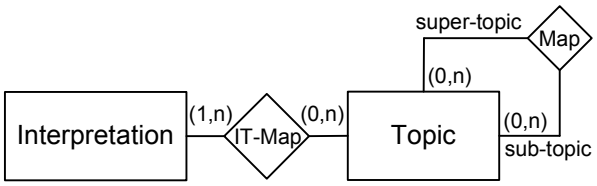

Figure 3.3: Interpretation to topic relation in eTVSM ontology.

Figure 3.3 shows the relation of interpretation concept to topic concept. Basically, an interpretation can be linked with an arbitrary number of topics. However, links between interpretations are not allowed. Because a topic map structure is free of cycles, provided extension to the topic map preserves this property. Interpretations now become leaf nodes of the ontology. A document model in eTVSM is stored in terms of interpretations.

Obtaining Interpretation Similarities

Interpretation similarities are obtained similar to topic similarities. Before we will start with constructing interpretation vectors, we will present some formalism:

- Φ, is the set of all interpretations;
- $g(\phi_i) \in [0 \dots 1]$, is the interpretation weight with $\phi_i \in \Phi$;
- $T(\phi_i) \in \wp(\Theta) \setminus \{\}$, is the interpretation to topic assignment, where $\wp(\Theta)$ is a powerset of all topics.

Now, the interpretation vector $\vec{\phi}_i = (\phi_{i,1}, \phi_{i,2}, \dots \phi_{i,t})$ can be defined as:

$$\vec{\phi}_i = \frac{g(\phi_i)}{\left|\sum_{\tau_k \in T(\phi_i)} \vec{\tau}_k\right|} \sum_{\tau_k \in T(\phi_i)} \vec{\tau}_k$$

We obtain weighted interpretation vectors (with $g(\phi_i)$ beeing the weight) as a normalized sum of topic vectors which are linked with the target interpretation. Visualization of the resulting interpretation vector is given in Figure 3.4:

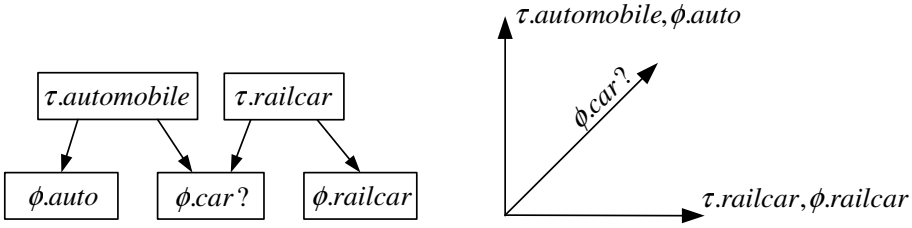

Figure 3.4: Interpretation to topic assignment example.

Interpretations similarity is defined as the scalar product of two interpretation vectors:

$$\vec{\phi}_i \vec{\phi}_j = \sum_{k=1}^{t} \phi_{i,k} \phi_{j,k}$$

The angle β_{ij} between interpretation vectors is:

$$\beta_{ij} = \cos^{-1} \frac{\vec{\phi}_i \vec{\phi}_j}{g(\phi_i) g(\phi_j)}$$

3.1.3 Terms

Term are treated as the smallest information unit that has one or several semantic interpretations. To express this multiplicity in semantic meanings, term might be linked with arbitrary number of interpretations. Figure 3.5 shows the relation between term and interpretation concepts (TI-Map). Another relation (SI-Map) is used to assign so called support terms to the term-interpretation link. The role of support terms is to explain semantic meaning of the TI-Map link. Furthermore, support terms are intended

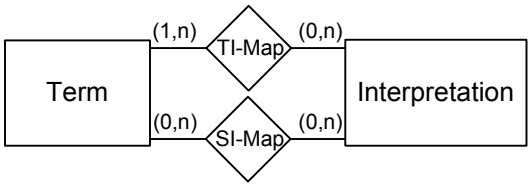

Figure 3.5: Term to interpretation relations in eTVSM ontology.

to be used for disambiguation of terms which means: to find a proper decision what is the "right" interpretation of a term (in case it might have several ones).

Basically, support terms are terms that co-occur frequently in a document with the term they are related to if a specific interpretation of the term is used. Consider interpretation to topic assignment example given in Figure 3.4. Also, consider that term *car* was designed to be linked with all three interpretations: $\phi.auto$, $\phi.car?$ and $\phi.railcar$. Consider *auto* interpretation link to be supplied by the following support terms: *auto*, *automobile*, *machine* and *motorcar*. At the same time *railcar* interpretation link has following support terms: *train*, *railway car*, *railroad car*, *railway*, *railroad*. Once we find the term *car* in a document, support terms assist us in deciding for the right interpretation to include in a document model. If a document contains terms like *automobile*, *machine*, *motorcar* it is obvious to decide for the $\phi.auto$ interpretation. Otherwise, one should decide for $\phi.railcar$ interpretation when a document discusses *trains* or *railway*. In case when it is impossible to give a significant preference to one of the interpretations a decision for the $\phi.car?$ interpretation is useful to avoid wrong decision.

A term can consist of an arbitrary number of words. Terms that consist of more than one word can be referred to as compound terms (see section 2.2). E.g., terms like *President John F. Kennedy* are compound terms. This allows treating such term occurrences as the interpretation of "*the 35th President of the United States*" rather than attempting to find appropriate interpretations from four parts of the string tokenized by space characters. However, the ability of the eTVSM to identify such compound terms greatly depends on an ontology you have modeled. You can as well define an ontology that does not recognize the term *President John F. Kennedy*. But, what is important: you as the modeler have such an opportunity to define compound terms if appropriate. Inter-term links are not allowed. Therefore, by attaching terms to interpretations we keep eTVSM ontology structure free from cycles. In Figure 3.6 we further refine our example from Figure 3.4 by attaching terms.

3.1.4 eTVSM Ontology Modeling Language

So far we have looked into concepts that contribute to a domain ontology which can be used by the eTVSM. These are topics, interpretations and terms. All together, by organizing them in a directed graph structure, they describe the world which an eTVSM system is capable of "understanding". eTVSM ontology concepts and their relations are shown in Figure 3.7.

The quality of retrieval results provided by an eTVSM system—as it will be shown later—greatly depends on the ontology upon which it operates. Therefore, the task of ontology modeling is of high importance. We have already shown how we can graphically represent an ontology. It is now time to present a common eTVSM ontology

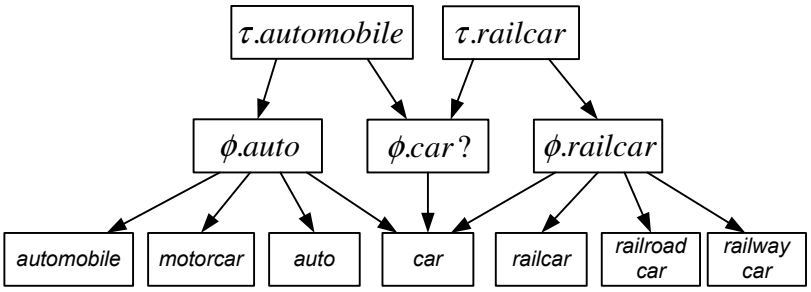

Figure 3.6: Term to interpretation to topic assignment example.

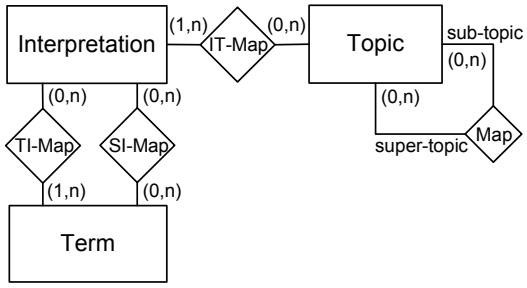

Figure 3.7: eTVSM ontology concepts relations.

graphical modeling language as it was proposed in [36]. Graphical notation for ontology concepts is shown in Figure 3.8.

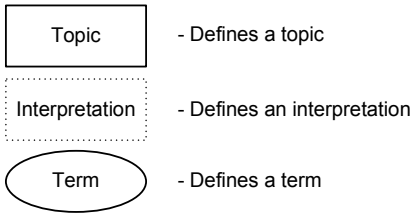

Figure 3.8: Graphical notation for eTVSM ontology concepts.

In order to define concept relations, following types of connector edges shown in Figure 3.9 are used. The original eTVSM ontology modeling language definition also proposes shortcuts (see Figure 3.10) in case when both, a topic and an interpretation, have the same name or there are no explicitly assigned terms to the interpretation, and this interpretation concept is just used to connect a term to a topic.

Support terms can be modeled as labels on term to interpretation edges. Alternatively, support terms can be derived automatically from the eTVSM ontology graph structure as terms that are assigned to neighboring interpretations. The original source of the modeling language proposes to automatically assign a term to a topic, through intermediate interpretation concept, all having the same name—the name of the term.

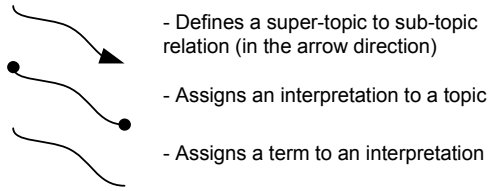

Figure 3.9: eTVSM ontology connector edges concepts.

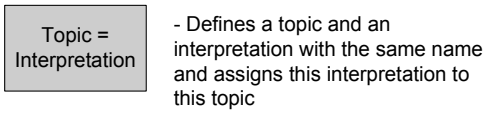

Figure 3.10: eTVSM ontology modeling notation shortcut.

However, we see this as the convenience extension to the modeling language that was presented.

3.2 eTVSM Ontology Modeling

We have claimed that eTVSM is capable of representing major linguistic phenomena like: inflection, composition, derivation, synonymy, hyponymy, meronymy, homography, metonymy and word groups. In this section we want to provide examples of eTVSM ontology modeling patterns for representing some linguistic phenomena with the graphical language presented in section 3.1.4. Explanation of proposed linguistic phenomena can be found in section 2.2.

Homography

Homography is a precedent when two words have the same orthography but different interpretations. See section 2.2.

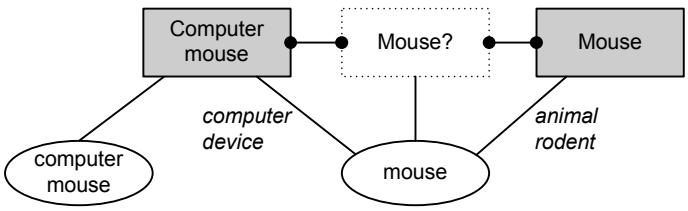

Figure 3.11: eTVSM ontology homography modeling.

Metonymy

Metonymy is the substitution of one word for another with which it is associated. See section 2.2.

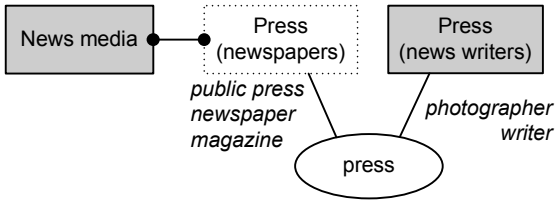

Figure 3.12: eTVSM ontology metonymy modeling.

Figures 3.11 and 3.12 present eTVSM ontology patterns to represent homography and metonymy linguistic term relations. However, the degree of freedom of eTVSM ontology composition allows for constructing of more complex arbitrary structures that do not fit into presented patterns or combine them. For further details refer to [36].

3.3 Constructing the eTVSM Document Model

The primary responsibility of the eTVSM as the Information Retrieval model is to provide similarities between documents. In order to obtain similarity values the eTVSM operates on formal document representations—document models. In this section we will explain steps which the eTVSM undertakes in order to construct document models. Figure 3.13 presents the overall picture of the schema for eTVSM document model construction. Heuristics, related with such an approach, is discussed later in section 3.5.

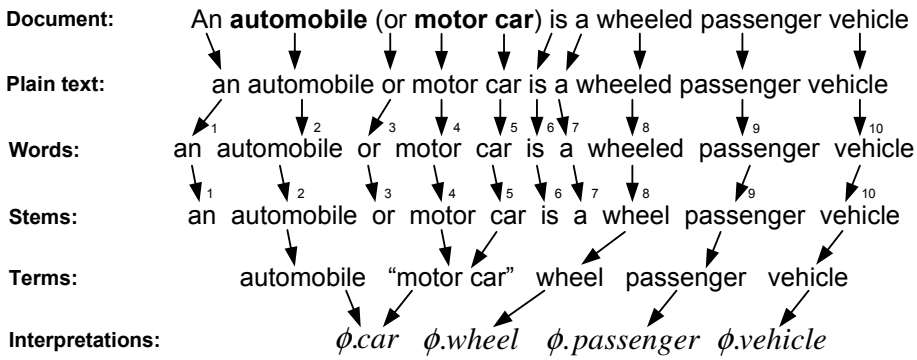

Figure 3.13: eTVSM document model construction steps.

Document to Plain Text Document

This step includes removing formatting from a source document. Also, one might consider filtering out a character subset from a string. Consider the following document:

> An **automobile** (or **motor car**) is a wheeled passenger vehicle that carries its own *motor*.

The result of applying this step to the document might return a string that looks like:

> *an automobile or motor car is a wheeled passenger vehicle that carries its own motor*

After this step all the formating of the document is filtered out.

Plain Text Document to Words

At this step a document is tokenized by a space separating characters set to give an ordered set of words that appear in the document. The output of this step applied to our document example:

> *an automobile or motor car is a wheeled passenger vehicle that carries its own motor*

will result in:

> {*an, automobile, or, motor, car, is, a, wheeled, passenger, vehicle, that, carries, its, own, motor*}

The output is an ordered multiset, with the member possibly included multiple times in the multiset. The reason to preserve word ordering will become clear later in this section.

Words to Stems

At this step a stemming algorithm (see section 2.1) is applied to each word from a multiset resulting from the previous step:

> {*an, automobile, or, motor, car, is, a, wheeled, passenger, vehicle, that, carries, its, own, motor*}

will result to something like:

> {*an, automobile, or, motor, car, is, a, wheel, passenger, vehicle, that, carry, it, own, motor*}[1]

This step preserves word ordering provided from the previous step and results to an ordered stems multiset.

Stems to Terms

At this step we extract terms that are present in eTVSM ontology from a document.

> {*an, automobile, or, motor, car, is, a, wheel, passenger, vehicle, that, carry, it, own, motor*}

will result to something like:

> {{*automobile*}, {*motor, car*}, {*wheel*}, {*passenger*}, {*vehicle*}, {*carry*}, {*motor*}}

[1] The result heavily depends on the stemmer applied.

Now, the reason for preserving stem order becomes clear. This was primarily done to be able to recognize compound terms that consist of more than one word, like *motor car* in our example. Note, that *motor car* string has a chance to be recognized as the term {*motor, car*} only, if such a compund was explicitly modeled in eTVSM ontology. From this moment on, the order of concepts becomes worthless for the eTVSM.

Stopword removal is implicitly included in this step. If the word is not found as a part of any term modeled in the ontology it will be filtered out. Preserving stopwords up to this step allows recognition of compounds that include stopwords, e.g. *head of state*—"*the chief public representative of a country who may also be the head of government*" gets the chance to be recognized as a single term. In case stopword removal has happened before, in previous steps, this step will probably classify this string as composed of two terms *head* and *state*. The impact of such misjudgment might result in "wrong" document model and as a consequence distortion of document similarity values.

Terms to Interpretations

At this step we resolve terms to interpretations by selecting interpretations that are modeled in domain ontology as connected to a target term. In case there are multiple interpretations connected to the target term, the decision should be carried out based on the document content analysis.

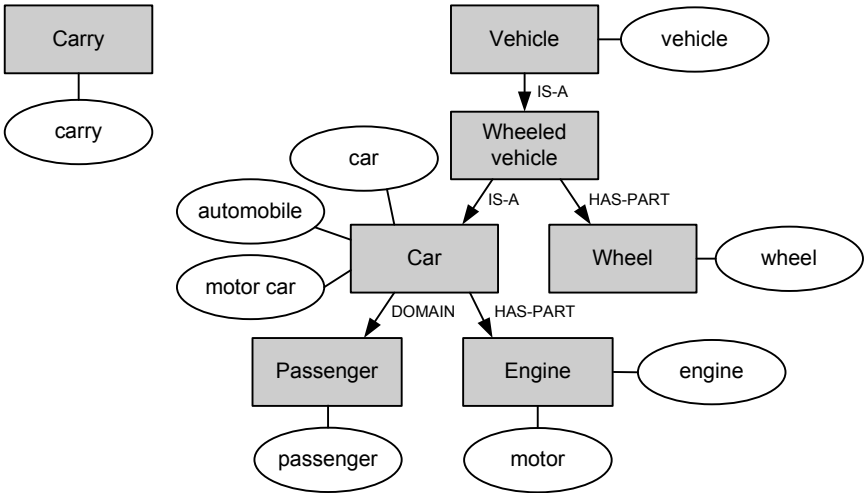

Figure 3.14: eTVSM ontology extract example.

Considering the eTVSM ontology extract proposed in Figure 3.14 the application of this step to the result of the previous step:

{{*automobile*}, {*motor, car*}, {*wheel*}, {*passenger*}, {*vehicle*}, {*carry*}, {*motor*}}

will result to:

{ϕ.*car*, ϕ.*car*, ϕ.*wheel*, ϕ.*passenger*, ϕ.*vehicle*, ϕ.*carry*, ϕ.*motor*}

This multiset of interpretations is the eTVSM document model for the initial document. The content and semantics of the document is completely defined by eTVSM ontology concepts and can be obtained by looking into the relations between concepts included into the document model.

Automobile and *motor car* terms both get mapped to the ϕ*.car* interpretation as these terms are both members of the total synonymy pattern instance for topic τ*.car*. Resolution of other terms is trivial one-to-one term to interpretation mapping in our example. The domain ontology of the example also sets a meronymy relation between τ*.engine* and τ*.car*, τ*.wheel* and τ*.wheeled vehicle* topics. A hyponymy relation is set between τ*.car* and τ*.wheeled vehicle*, τ*.wheeled vehicle* and τ*.vehicle* topics. Thus, we obtain an automobile domain as the directed graph of interconnected automobile related topics. Orthogonal to automobile domain is the τ*.carry* topic which in our example is not connected to any other topic.

3.4 eTVSM Document Similarity

Now, as we have formal document representation (the document model), we can derive a procedure for obtaining document similarities. eTVSM document similarity calculation is similar to one in TVSM. The difference is that TVSM operates directly on terms and eTVSM has an intermediate concept of interpretation between topic and term concepts which is used for obtaining similarities. Thus, eTVSM treats a document as the entity composed of interpretation concepts. It was already shown how this aids in representing linguistic phenomena (see section 3.2). A document vector is defined as:

$$\forall d_j \in D : \vec{d_j} = \frac{1}{\left|\vec{\delta_j}\right|}\vec{\delta_j} \quad \Rightarrow \quad \left|\vec{d_j}\right| = 1 \quad \text{with} \quad \vec{\delta_j} = \sum_{\phi_i \in \Phi} \omega_{d_j,\phi_i}\vec{\phi_i}$$

The document vector is a weighted sum of interpretation vectors which are included in the document model. The document vector is normalized to the length of one. The document vector length is calculated similar to the TVSM:

$$\begin{aligned}\left|\vec{\delta_i}\right| &= \left|\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi_k}\right| \\ &= \sqrt{\left|\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi_k}\right|^2} = \sqrt{\left(\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi_k}\right)^2} \\ &= \sqrt{\sum_{\phi_k \in \Phi}\sum_{\phi_l \in \Phi} \omega_{d_i,\phi_k}\omega_{d_i,\phi_l}\vec{\phi_k}\vec{\phi_l}}\end{aligned}$$

Finally, the similarity between two documents d_i and d_j is again obtained analogously to the TVSM, as the scalar product of corresponding document vectors. Considering the document vector normalization, the similarity value becomes equal to the cosine of the angle between document vectors:

$$
\begin{aligned}
sim(d_i, d_j) &= \vec{d_i}\vec{d_j} \\
&= \frac{1}{\left|\vec{\delta_i}\right|}\vec{\delta_i}\frac{1}{\left|\vec{\delta_j}\right|}\vec{\delta_j} \\
&= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\vec{\delta_i}\vec{\delta_j} \\
&= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi}_k \sum_{\phi_l \in \Phi} \omega_{d_j,\phi_l}\vec{\phi}_l \\
&= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\sum_{\phi_k \in \Phi}\sum_{\phi_l \in \Phi} \omega_{d_i,\phi_k}\omega_{d_j,\phi_l}\vec{\phi}_k\vec{\phi}_l
\end{aligned}
$$

3.5 Heuristics for the eTVSM

In this section we would like to discuss heuristics in eTVSM. These are the points where one might apply several solutions in eTVSM implementation that consequently will lead to different similarity values produced for the same document set. It is up to a developer and an ontology modeler to find the best approach compromise between factors that will be discussed now.

Ontology modeling. The eTVSM extends the TVSM by defining a formal approach for acquiring document vectors and consequently angles between these vectors from the eTVSM ontology structure. Like any other modeling task, eTVSM ontology modeling can be accomplished in numerous ways, applying different principles leading to different results. Different modeling examples, including same concepts that yield different similarity values, were presented in [36].

Ontology mapping on vectors. The formal procedure for deriving document vectors which was described in this chapter is just one reasonable approach. One might think of any other usable approach which will transform eTVSM ontology concepts to document vectors which will express ontology relations in a vector space through angles between concept vectors.

Interpretation weights. As it was shown in section 3.1.2 interpretation vectors are obtained as a weighted sum of topic vectors. However, it is not defined how to obtain weight values. It is just stated that these values should be assigned values on the $[0, 1]$ interval. Moreover, [36] proposes to initially assign to all weights the value of one. The reason for the interpretation to topic relation weight assignment is to express the relation of an interpretation to the specific topic in case when this interpretation is connected to multiple topics. Thus, the assignment of values from the [0,1] interval instead of just one has an intuitive advantage.

Term to interpretation link selection. This task should be solved when migrating from terms contained in the document to interpretations. It is possible that one term is assigned to many interpretations. However, it is required to select only one interpretation. This choice is again a heuristic decision. Support terms discussed in section 3.1.3 should aid the selection decision. However, support terms alone are not enough to carry out the deterministic decision in favor of a concrete interpretation.

Word to term resolution. Terms in eTVSM ontology are considered the smallest content bearing unit with a specific semantic meaning. Consider the following word

sequence obtained from a document—*{new, york, city, tour}*. Some possible term resolutions considering that all these terms are modeled in eTVSM ontology can be:

- {{*new, york, city, tour*}};
- {{*new, york, city*},{*tour*}};
- {{*new, york*}, {*city, tour*}};
- {{*new*}, {*york, city*}, {*tour*}};
- etc.

In a real size document the number of possible resolutions explodes. One possible approach to solving this problem is at each step to take the longest possible term that matches the document starting words. Afterwards, restart the procedure from a new starting point—at the place where the last longest term has terminated. The procedure is continued till the end of a document.

Preprocessing place. eTVSM is highly sensitive to the order and place of applicability of document preprocessing steps. Consider the following document—"*United States of America*". In case the stemming and the stopword removal is done before words to terms resolution the recognition of this string sample to a single term of "*North American republic containing 50 states*" becomes impossible. Rather, the most probable resolution will result three terms: *unite*, *state* and *America*.

Though so many heuristic points that are present in eTVSM, one might find here a great potential for ad hoc model re-configuration. One possible use scenario might be considering of user feedback on certain query results. One might express user intentions in eTVSM ontology upon requirement arises to increase similarity values between certain documents.

Chapter 4

Information Retrieval Quality Measurements

4.1 Measurements

There are number of ways to measure how well the retrieved information of Information Retreival systems matches the intended one. Before we introduce these measurements lets take a look at the formalism needed to formally define the measurements: Let D be a set of documents for validation and K be a set of criteria/queries. $R_k^S \subseteq D$ is a set of documents which are relevant according to the system when criteria k has been used, and $\complement R_k^S = D \setminus R_k^S$ is a set of documents not relevant according to the system. Analogously we can define $R_k^T \subseteq D$ as a set of documents "really" relevant according to the user and $\complement R_k^T = D \setminus R_k^T$ as a set of documents not relevant according to the user. Figure 4.1 visualizes a single Information Retrieval experiment which consists of criteria/query input and an analysis of retrieved documents.

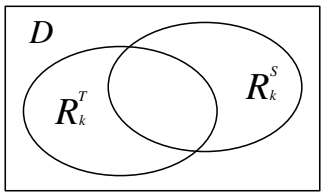

Figure 4.1: Information Retrieval experiment visualization.

4.1.1 Recall

In these terms recall can be defined as the ratio of correct assignments by the system divided by the total number of correct assignments:

$$R_k = \frac{\#\left(R_k^S \cap R_k^T\right)}{\#R_k^T} \in [0 \ldots 1]$$

Graphically this is depicted in Figure 4.2.

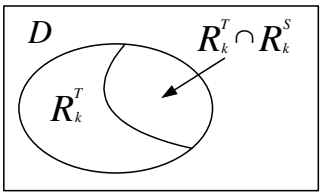

Figure 4.2: Recall visualization.

$$R = \frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ relevant\ documents}$$

Recall measures the completeness of the results; therefore, high values are desirable.

4.1.2 Precision

Precision is the ratio of correct assignments by the system divided by the total number of system assignments:

$$P_k = \frac{\#\left(R_k^S \cap R_k^T\right)}{\#R_k^S} \in [0 \ldots 1]$$

Graphically this can be depicted as in Figure 4.3:

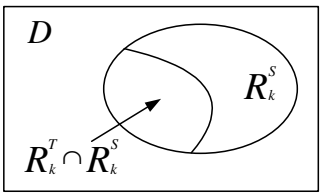

Figure 4.3: Precision visualization.

$$P = \frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ documents\ retrieved}$$

Precision measures the accuracy of the results; therefore, high values are desirable. Recall and precision can also be evaluated at a given cut-off rank, denoted as $R@n$ and $P@n$, where instead of all retrieved documents only top n are considered.

4.1.3 F-measure

F-measure can be used as a single quality measure of retrieval experiment. F-measure is the weighted harmonic mean of precision and recall. F-measure can be obtained as:

$$F = \frac{2pr}{(p+r)}$$

It combines recall (r) and precision (p). This is also known as F_1 measure because recall and precision are evenly weighted. F_1 measure was initially introduced by van Rijsbergen [62]. In general case the formula for F_N is given for arbitrary N as:

$$F_N = \frac{\left(1 + N^2\right) pr}{p + N^2 r}$$

Two other commonly used F measures are $F_{0,5}$, which weights precision twice as much as recall, and F_2 measure, which weights recall twice as much as precision.

4.1.4 Error rate

As the negative measure of the system performance one can introduce the error rate:

$$E_k = \frac{\#\left(\complement R_k^S \cap R_k^T\right) + \#\left(R_k^S \cap \complement R_k^T\right)}{\#D} \in [0 \ldots 1]$$

The error rate measures the quota of errors; therefore, low values are desirable.

During the evaluation measurements for each single experiment should be collected. The task of the systems comparison based only on a large amount of individual observations is not feasible since this means that multi-dimensional values have to be compared. Therefore an aggregation of these measurements is useful. One approach is based on acquiring individual measurement for each experiment and then averaging over experiments. Or, the measurement might be computed globally over all experiments. The former way is called macro-averaging and the latter way is called micro-averaging. Further, the approach of macro averaging of measurements over experiments with further aggregation of results through statistical tests for acquiring the significance level of the evaluated parameter is proposed (see section 5.6).

4.2 Measurements Graphical Representation

Precision and recall are the most common Information Retrieval quality measurements. These two measurements usually show an inverse relation. So that when one measurement value increases—the other one decreases. The desired system state is such when both recall and precision are equal to one. This is the case when an Information Retrieval system returns exactly the number of documents which are relevant $\left(\#R_k^T\right)$ and all the returned documents are relevant $\left(R_k^T = R_k^S\right)$. This condition should hold for every input query. Of course, such an ideal system is not realistic. In the real world, Information Retrieval systems would behave more like it is shown in Figure 4.4.

Two lines may represent the performance of different retrieval systems. One might derive the graph like this by simply plotting the average over queries (recall, precision) points, as the function of a cut-off rank, and connecting them. Further, one might fit in a trendline or use any kind of smoothing technique. While the exact slope of the curve may vary between systems, the general inverse relationship between recall and precision remains. Much of this relationship has to do with language. If the goal of a search is comprehensive retrieval, then the searcher must include synonyms, related terms, broad or general terms, etc. for each concept. He may decide to combine terms using boolean rather than proximity operators. In addition, some secondary concepts may be omitted. As a consequence of these decisions, precision will suffer. Because synonyms may not be total synonyms and the probability of retrieving irrelevant material might

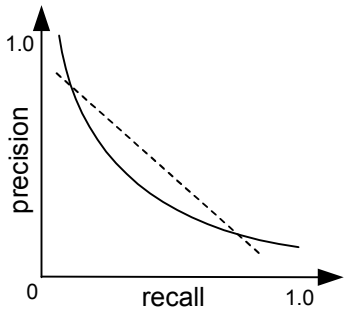

Figure 4.4: Recall-precision plot.

increase. Broader terms may result in the retrieval of material which does not discuss the narrower search topic. Using boolean operators rather than proximity operators may increase the probability that the terms won't be in context. Unfortunately, if the searcher doesn't use these techniques he won't achieve high recall.

As it was shown in section 4.1—precision and recall are set-based measures. They evaluate the quality of an unordered set of retrieved documents. Further, in chapter 5 we propose a new approach to Information Retrieval system comparison where the comparison judgment has a strong statistical reasoning and provides significance level of derived outcome, rather than leaving decision solely based on a visual plots comparison.

However, often, the Information Retrieval scenario is such that it returns an ordered by relevance list of documents. To evaluate such a ranked list, precision can be plotted against recall after each retrieved document. Because, each input query might result a different number of relevant documents, individual query precision values are interpolated to a set of standard recall levels. Intuitively, we are trying to discover precision once recall reaches a specific (standard) level. Usually, eleven standard recall levels are used (from 0 to 1 in increments of 0.1).

The particular rule used to interpolate precision at standard recall level i is to use the maximum precision obtained for the query for any actual recall level greater than or equal to i. This interpolation rule also defines precision for a standard recall level of 0.0. Note, in general, precision is not defined for a recall of 0.0. A recall level corresponds to a recall obtained with R_k^S set constructed from top n documents returned after a search procedure. Precision that corresponds to a recall level should be obtained on same R_k^S set (when a relevant document is not retrieved at all, its precision is assumed to be 0). Interpolation to standard recall levels then should take place on all recall levels and corresponding precision values for $n = 1 \ldots N$. Here, N is the number of top documents that include all the relevant documents for the query searched.

Let us take a look at the following example. Let $R_k^T = \{d_3, d_{51}, d_{67}\}$. Further, let us assume that our system has returned the following ordered by relevance sequence of documents $R_k^S = \{d_7, d_{51}, d_{33}, d_{61}, d_3, d_{22}, d_1, d_{87}, d_6, d_{67}, \}$. Here, the first document is considered the most relevant—this is document d_7. In our example, the first relevant document retrieved is d_{51}, which gives recall of $0.\bar{3}$ and precision of 0.5. Next relevant document is d_3 which gives us $0.\bar{6}$ recall at 0.4 precision. Next, and last, relevant document d_3 gives us 1.0 recall with 0.3 precision. In Figure 4.5, all actual precision values are plotted with circles (and connected by a solid line) and the inter-

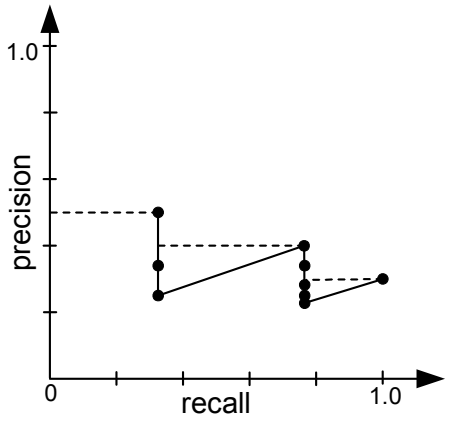

Figure 4.5: Precision at standard recall levels plot.

polated precision is shown with the dashed line.

Afterwards, two Information Retrieval models can be compared based on the average precisions over all queries on 11 standard recall levels graph. Each recall-precision average is computed by summing the interpolated precisions at the specified recall cut-off rank (denoted by $\sum P@N_i$ where $P@N_i$ is the interpolated precision at recall level i and a cut-off rank N) and then dividing by the number of queries:

$$\frac{\sum_{j=1}^{\#Queries} P_j@N_i}{\#Queries} \qquad i = \{0.0, 0.1, 0.2, \ldots, 1.0\}$$

The average plots of different runs/models can be superimposed on the same graph to determine which run is superior. Curve which is closest to the upper right-hand corner of the graph (where recall and precision are maximized) indicates the best recall-precision performance system. Also, often such comparisons are carried out in three recall ranges: $[0...0.2)$, $[0.2...0.8)$ and $[0.8...1.0]$. Interval borders are dictated by the interpolation rule defined before. These ranges characterize high precision, middle recall, and high recall performance, respectively.

Chapter 5

Comparison of Information Retrieval Models

Many Information Retrieval models and systems exist. Some of them are quite simple and do not require much computation power, others have sophisticated algorithms claiming to deliver better results. In addition, new and promising models appear on the market. In fact, in chapters 2 and 3 we have described the evolution steps of eTVSM retrieval model from VSM, through TVSM. However one question still remains: is it better than the other ones presented in this work or any other existing models?

In this chapter we discuss the method of performing quantitative evaluation of an Information Retrieval system implementing an Information Retrieval model. We propose statistical approaches tailored for the purpose of Information Retrieval model comparison. As a consequence we can come up with a model effectiveness order through their pair-wise comparison. In this case we can say which system performs better under the given conditions. Another perspective of this problem arises when instead of two alternative systems, we have one system which we think might be capable of beeing better then the other. In this situation, we might for instance want to evaluate how well it performs under specific model setup and how it reacts to changes of setup variables. This is especially important when evaluating a model which performance highly depends on its internal configuration. This is exactly the case with the eTVSM. The results provided by the eTVSM highly depend on the configuration of the domain ontology which is used by the eTVSM.

This chapter includes a complete description of all steps that should be taken to perform a quantitative evaluation of Information Retrieval models. It includes preparation of the workload, following by an experiment description with further identification of statistical aggregation methods of the collected measurements. The proposed approach suits both described scenarios—inter-model comparisons, and study of the model effectiveness under different model configurations.

5.1 Design of a Model Evaluation

Referring to [45], we aim to derive a design for a pure laboratory test under the assumption that our test setup is configured to correspond to the real world model and therefore statistical generalization is adequate. Moreover, we pay more attention to a system centered approach as we are particularly interested in the "quality" of the results

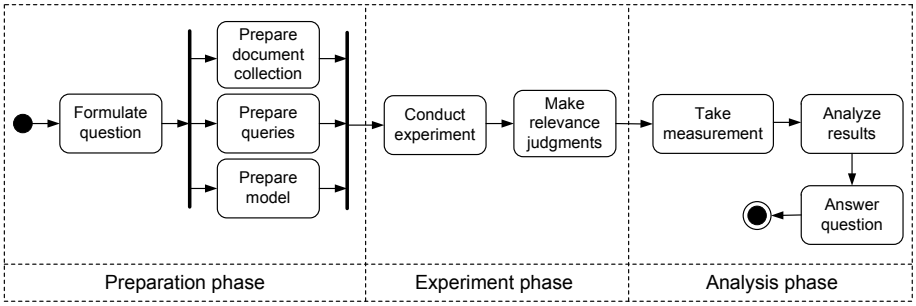

Figure 5.1: Design of a model evaluation procedure.

of a model. In this section we will discuss the design of the overall evaluation setup to perform such an evaluation. In other words we will discuss what steps are needed and in which order these steps should be executed in order to perform the evaluation. In Figure 5.1 you can see the proposed "big picture" of evaluation design that is given in the notion of an activity diagram.

Tests in general, and experiments in particular, are usually intended to answer specific questions. In our case the question that we want to answer is: "*How effective is the Information Retrieval system?*"

> Effectiveness is how well the system does what it is supposed to do; its benefits are the gains deriving from what the system does; its efficiency is how cheaply it does what it does. [32]

We understand effectiveness as the ability of the Information Retrieval system to retrieve relevant documents and to suppress non-relevant documents. In this context we can assume benefit to be proportional to the effectiveness. The efficiency of the system is out of scope as it can be evaluated as a part of the user centered approach [45]. As you can see in Figure 5.1, we start with the question and then perform steps in order to be able to answer this question at the end.

5.2 Elements of the Evaluation

We will come up with an detailed approach of how to perform evaluation of an Information Retrieval system. Moreover, we will make this approach suitable for the evaluation of the eTVSM and make it such that it will aid us in choosing an optimal strategy for the eTVSM configuration. Furthermore, we will propose a short description of the basic phases that we propose for the evaluation process.

The preparation phase deals with preparing all necessary components needed for performing an evaluation. The actual system evaluation will start in the next phase; however, it is preferable that prior this all necessary measures are taken. These measures include getting the workload for the system that will be tested as well as preparation of the system itself. All internal variables that influence systems work should be configured.

An important component of any test is the experimental design—the way in which the test is organized in order to answer our questions that we pose to the system. At

the experimant phase all the information prepared in the previous phase should be effectively fed into the system. By saying effectively, we mean the way that allows completely exploiting all available information, and transforming it into the data array that is suitable for statistical analysis. We do this by collecting measurements (see chapter 4) in the course of the evaluation for every single experiment that accumulates the knowledge about this experiment and that characterizes the system in the way it is needed for the evaluation.

In the analysis stage we use statistical methods to work with measurements collected during the experiment phase. This is a pure statistical phase, where by statistical means we try to understand what retrieved results say about the system, and to which extent they can be generalized. After this stage either we answer the question, or decide that the data processed was not informative enough to aid us in answering the question—which is also a valid outcome.

5.3 Test Collection

To conduct evaluation of a system we need to come up with an appropriate workload. In case of Information Retrieval systems this workload is represented in the form of a test collection. What do we need in order to be able to perform evaluation? To answer this question we should look on how does a regular interaction of a user with an Information Retrieval system look like. The user wants to retrieve data from a large array of documents. To do this he/she formulates information demands in the form of a query request. Afterwards, the user feeds this query into a system and receives document from the system which are evaluated by the system as beeing relevant. The relevance here is considered by the Information Retrieval system. From this simplistic example it is clear that we need a document collection upon which our system will perform the retrieval task. Furthermore, we need queries that will initiate retrieval process and will simulate user requests. This already enables us to make the system work: perform some actions and return results. However, these results are pointless without actual users deciding whether documents retrieved by the system are really what he/she was looking for. To be able to judge the work of our system we need to have relevance judgments or relevance assessments that can be seen as the following function:

$$rel\,(d, q) \in [0, 1]$$

where d is a document from our document collection and q is a query. In general this function is defined on the interval from zero to one. "0" relevance means that this document is absolutely not related to the query, and "1" means that this document is absolutely related to it. Other values in the interval reflect upon the relevance level. As a special case we can work with the rel function that is defined as returning values which are members of the set: $\{0, 1\}$. At the end these three components: *document collection*, *queries* and *relevance judgments* form a test collection. We will now describe each of these components.

Document Collection

There are plenty of documents available on the web: news articles, electronic libraries, web pages. All of them suit for our purpose. Alternatively it is possible to do some simulation experiments using pseudo-documents which are generated in some fashion

(perhaps involving Monte Carlo techniques) [32]. This kind of simulation of course has its role and can aim answering specific research questions. But, in most cases it is easier and better to use genuine documents.

Queries

The situation with queries is more challenging than in case with documents. And there are number of reasons for this. If we stick to the idea of simulating real world conditions, we need to obtain real queries. The problem with genuine queries is that it is hard to trap them as they exist for a short period of time and the location of such request acts is usually sparse. But this is nothing in comparison with the actual time needed to collect the queries that will be targeted for the use with acquired document collection. The major challeng is: Can we guarantee that genuine queries will at least target the topics of the documents in such collection? Here arises a strong necessity in artificial queries construction. Such artificial queries may vary in their degree of realism; the main point is that they should exploit the document collection properties. The problem here is that we do not really know what are the important characteristics which we should be trying to reproduce [32]?

It is critical to understand the importance of queries selection. These are the queries that will initiate retrieval process of the evaluated system. Thus, obtained queries collection should be able to numerously exploit all aspects of the retrieval model to allow statistical generalization on results. In order to achieve this goal queries should be formed to be matched to the most of documents from the document collection and carry different levels of relevance to these documents.

Relevance Judgments

Effectiveness, as we have defined it, is good for the understanding of what is happening, but is definitely not sufficient as a formal basis for an experiment. It still remains unclear how can we come up with appropriate relevance judgments? As the output of the Information Retrieval system is targeted for the human use it is only the user who can make such judgments. The next question which arises is which user's judgments can be considered correct? Quite often people have different preferences and different point of views on common issues. Therefore, relevance judgments cannot be done by a single person but should be performed by a group of people simultaneously on the same query on the same document collection. Furthermore, the estimated measured parameter should be obtained as the mean value over calculated measurements from all the judges. First of all this approach allows us to construct confidence interval for the measurement estimator and make further statistical reasoning on the number of the judges being sufficient. Also, we can classify queries that cause judges to provide sparse relevance assessments.

Relevance judgments can be considered as a bottle-neck of the overall test collection creation process. Ideally, relevance judgments should be made for each document upon each query. Considering the fact that each query-document pair is judged by several people the amount of work needed to be done explodes. Additionally, by assuming an ideal case when judgments are made on the full content of the document this work can be out of manageable scope for an experimenter. Due to this reason we advise to take a look at existing test collections that were used for other Information Retrieval experiments and are publicly available online.

5.4 Existing Test Collections

By now, it should be clear that a test collection creation itself is a challenging task. Additionally, failing to come up with representative test collection may, and probably will spoil all further calculations. Therefore, it makes sense to take a look at existing test collections that have been used in different Information Retrieval experiments and are publicly available. These collections are already tested and are proven to deliver usable results. Further, by carrying out evaluation of our system on these collections we get a chance to compare our system with already available evaluations of other systems on these collections.

Classical test collections can be considered those that were used in the first experiments of Information Retrieval and still appear in the literature as those proved to be useful in the past. In Table 5.1 you can find a list of selected commonly known test collections:

Name	#Docs	#Queries	Size [MByte]
ADI	82	35	0,04
Time	425	83	1,5
Medline	1033	30	1,1
Cranfield	1400	225	1,6
CISI	1460	112	2,2
CACM	3204	64	2,2
LISA	5872	35	3,4
NPL	11429	93	3,1

Table 5.1: Classical test collections.

Test collections provided in Table 5.1 are available online and include documents, queries and relevance judgements as discussed above.

In the next category we distinguish test collections that appeared later. These collections are usually larger in size. The first one that can be classified to this category is *Reuters-21578*. It consists of 21578 documents and is the ancestor of *Reuters-22173* [55]. These collections are approximately 20Mb in size. Long time *Reuters-21578* was the most widely used collection for text categorization evaluations, but now the situation seems to change in favor of larger collections such as *Reuters Corpora Vol.1* and *Reuters Corpora Vol.2*. Though all these test collections were originally designed for Information Filtering evaluations and thus have no queries provided for conducting Information Retrieval evaluations, they are widely used for retrieval evaluations as well. An approach for creating proper queries is described in [55]. Also, during TREC 2002 [3], 50 search topics (test queries) were developed on *Reuters Corpora Vol.1* for the filtering track. They can be applied for testing ad hoc searches [6]. *Reuters-21578* is available for download on the web [2], while *Reuters Corpora* Volumes one and two are available from NIST, the National Institute of Science and Technology. Other often used test collections are available from National Institute of Informatics [1].

Finally, we want to draw your attention to test collections provided by TREC [3]. The Web research collections are distributed by the University of Glasgow for research purposes only. In order to receive copies of one or more of these collections, you must sign an agreement with the University of Glasgow and pay a contribution to the University's various costs in preparing and distributing the data. These collections and contribution amounts are provided in Table 5.2:

Name	Size [GByte]	Fee
WT2g	2	£250
WT210g	10	£400
.GOV	18	£400
.GOV2	426	£600

Table 5.2: TREC test collections.

5.5 Experimental Design

To support the procedure proposed in Figure 5.1, the experimental design has to bind the preparation phase with available workload, a configured system and acquired measurements. Most of such existing experiments use just one set of requests/queries to evaluate or compare a number of systems. It is called "matched pairs" procedure [32], when the efficiency of the systems is compared on the same request. Moreover, there is a clear statistical reason for such approach. Any statistical significance testing will be much more efficient with this method. Again, this approach is oriented to decrease the influence of the bottle-neck of the whole evaluation process which is the amount of requests. With this approach it is possible to reuse the requests decreasing the need in the larger number of distinct requests. Therefore, the experimental design can be quite simple. Each request/query is searched against every system or every system configuration. Since the searching part of the system is controlled by simple rules, there is no problem in relation to replicating searches or the order in which the systems are tried [3]. The only matter of convenience in case of a single system evaluation is performing the evaluation for the whole request set for a single configuration, further reconfiguring the system and performing the complete course of evaluation for a new setup.

5.6 Statistical Comparison Justification

So far we have came up with measurements that characterize Information Retrieval model effectiveness (see section 4.1) and we have came up with an evaluation design (see section 5.1) that incorporates the collection of these measurements. In this section we will discuss approaches of statistical analysis of collected measurements. There are two approaches: The first aims to select the best out of two systems. These can be different systems or the same system being evaluated in different configurations. The second approach is a test, which is designed to check the level of significance in the difference of effectiveness of two systems. This test is known as t-Test [44] in statistics. Furthermore, proposed statistical methods can be applied to all presented measurements. Interpretation of results, however, depends on the comparison order of specific measurement.

5.6.1 Assumptions

Prior to starting statistical evaluations on retrieved measurements we will make some assumptions on the nature of the data processed. The key assumption is that the measurement values of Information Retrieval systems are normally distributed. This assumption follows immediately from the way these measurements are collected. The observed precision, recall, F, and other measures are obtained through averaging mea-

surements obtained by different people. Therefore, the Central Limit Theorem [57] comes into play: Be $(X_i)_{i\in\mathbb{N}}$ a sequence of independent and identically distributed (*iid*) random variables with actuarial expectation $\mu = E[X_i] \in \mathbb{R}$ and variance $\sigma^2 = Var[X_i] \in \mathbb{R}$. We can then define:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad Z_n = \frac{S_n - \mu}{\sqrt{\sigma^2 n^{-1}}}$$

$$F_n(t) = Pr[Z_n \leq t]$$

Here, Z_n is "standardized" S_n. Being $F_n(t)$ the distribution function of Z_n, then

$$F_n \xrightarrow{d} \Phi$$

as $n \to \infty$. Here Φ is the distribution function of a normal distribution with mean value 0 and variance 1. Therefore, we can assume our measurement as being a realization of a random variable that has normal distribution with unknown parameters μ and σ^2.

5.6.2 Comparing Two Systems

In this section we will provide an approach for comparison of Information Retrieval systems. We will restrict our method to the case of comparing two systems. As already mentioned, these can be completely different systems or the same system evaluated under different internal configurations. We use the above stated assumption of observed measurement being *iid* for both systems. Further, let μ_1 and μ_2 respectively be the true measurement value expectations for test-systems A and B. The key idea of this comparison method is to find a confidence interval I_α for the difference $\mu = \mu_1 - \mu_2$ and a given confidence level α. μ can be estimated through $\hat{\mu}$ (the mean value). Figure 5.2 shows possible outcomes for $\hat{\mu}$ including the confidence interval.

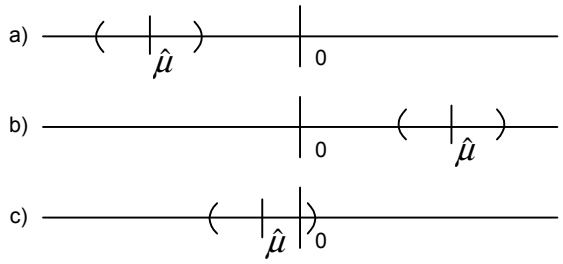

Figure 5.2: Possible outcomes for $\hat{\mu}$ confidence interval.

There are three different outcomes when comparing two systems for difference in their mean values of the observed parameter. Outcome (*a*) corresponds to the case when system B is significantly better than system A, while outcome (*b*) will mean that system A is significantly better than system B. In case of outcome (*c*) a clear distinction is not possible. In this case further measurements needs to be made until the confidence interval does not contain 0.

To reconstruct the confidence interval we will use Paired-t confidence interval [41]. It is used in Student's t-test for the statistical significance of the difference between two

sample means, and gives confidence interval for the difference between two population means. To the already discussed assumptions one should add the one that the number of replications (experiments) is exactly the same for both systems (observations are "paired"—measured for the same input for different systems). With these assumptions the random variables

$$Z_i = X_i - Y_i$$

with observations $z_i = x_i - y_i$ are *iid*. This allows us to compute:

$$\hat{z}(n) = \frac{1}{n}\sum_{i=1}^{n} z_i \qquad S^2(n) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \hat{z}(n))^2$$

as unbiased estimations of $E[Z]$ and $Var[Z]$. Now we can construct α level confidence interval as:

$$\left[\hat{z}(n) - t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2(n)}{n}}, \hat{z}(n) + t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2(n)}{n}}\right]$$

Here, t is the quantile of the Student-t distribution [26].

In case the number of observations differs for both systems we can use a modified procedure for confidence interval construction. This procedure is known as Welch Procedure [16]. It assumes that n_1 needs not be equal to n_2:

$$\hat{x}(n_1) = \frac{1}{n_1}\sum_{i=1}^{n_1} x_i \qquad \hat{y}(n_2) = \frac{1}{n_2}\sum_{i=1}^{n_2} y_i$$

$$S_x^2(n_1) = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1}(x_i - \hat{x}(n_1))^2$$

$$S_y^2(n_2) = \frac{1}{n_2 - 1}\sum_{i=1}^{n_2}(y_i - \hat{y}(n_2))^2$$

Then we compute the estimated degree of freedom as:

$$\hat{f} = \frac{\left(\frac{S_x^2(n_1)}{n_1} + \frac{S_y^2(n_2)}{n_2}\right)^2}{\frac{(S_x^2(n_1))^2}{n_1^2(n_1-1)} + \frac{\left(S_y^2(n_2)\right)^2}{n_2^2(n_2-1)}}$$

Finally we compute the confidence interval for $\hat{\mu} = \hat{x}(n_1) - \hat{y}(n_2)$ as:

$$u_{u,l} = \hat{\mu} \pm t_{\hat{f},1-\frac{\alpha}{2}}\sqrt{\frac{S_x^2(n_1)}{n_1} + \frac{S_y^2(n_2)}{n_2}}$$

$\hat{f}$ is usually not an integer, therefore one should look at neighbored integers when finding $t_{\hat{f},1-\frac{\alpha}{2}}$ and pick the larger one.

5.6.3 Significance Test (*t*-Test)

In the previous section we have discussed an approach for comparing two systems. This approach can be extended in a way to determine the level to which system A outperforms system B. This kind of analysis can be performed with the help of statistical test, and to be more precise—the t-Test [41]. The evaluation that was discussed in the previous section should be continued to the point when the confidence interval for the observed parameter does not contain 0. This means that there is no need to perform all experiments (feed in all queries), but continue evaluation to the point when clear distinction can be made. However, the number of queries we posses might not be sufficient for the experiment to deliver any results. This might be the case when using test collections proposed in Table 5.1. In this case one might use the approach proposed in this section. It will always deliver results, but again the level of significance will depend on the amount of information processed in the experiment.

Again, we have observations of the tested measurement for both systems. X_i for the system A and Y_i for the system B. As we have explained in the assumptions section 5.6.1:

$$X_i \sim N(\mu_x, \sigma_x^2) \qquad Y_i \sim N(\mu_y, \sigma_y^2)$$

This gives us possibility to construct a test. The null hypothesis is $H : \mu_x - \mu_y \leq d_0$, or that the difference of expectations for the observed parameter is less than some tested value d_0. The alternative hypothesis is that this difference is larger than tested value d_0 meaning that system A performs for this parameter better than system B, $K : \mu_x - \mu_y > d_0$. Therefore, the aim of the test is to reject H and to confirm K to some significance level. According to the t-Test we reject H when:

$$T > t_{n_1+n_2-2,1-\alpha} \qquad (*)$$

where n_1 is the number of experiments for system A and n_2 is respectively the number of experiments for system B. In most cases $n_1 = n_2 = n$. However, they need not to be equal. α is the significance level to which we test d_0. t is the quantile of the Student-t distribution [26] and T is our test statistics which is obtained as:

$$T = \frac{\bar{X} - \bar{Y} - d_0}{\sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\hat{\sigma}^2}}$$

where $\bar{X}$ and $\bar{Y}$ are the arithmetic averages of the observed parameter for both systems. And $\hat{\sigma}$ is the standard quadratic failure for the random variable $X_i - Y_i$. Because T is calculated using random variables, T itself is a random variable. It can be shown that T has a Student-t distribution:

$$T \sim t_{n_1+n_2-2}$$

where $n_1 + n_2 - 2$ is the degree of freedom used in the Student-t distribution. The last thing we need in order to be able to perform all the calculations is the method of calculating $\hat{\sigma}^2$. $\hat{\sigma}^2$ is the estimator of the variance for the random variable $X_i - Y_i$ and can be obtained as:

$$\hat{\sigma}^2 = \frac{(n_1 - 1)\, S_x^2 + (n_2 - 1)\, S_y^2}{n_1 + n_2 - 2}$$

where S_x^2 and S_y^2 are the estimators of the variance of random variables X_i and Y_i. $\hat{\sigma}^2$ is used to compute test statistics in two-sample pooled t-Test when number of observations for both systems differs [41]. The formula to obtain estimator of the variance was already provided in section 5.6.2.

With this approach an experimenter is able to vary parameter d_0 which specifies the level to which system A outperforms system B in evaluated parameter and the α level. Also, it is possible to get the corresponding α level for the tested data by calculating test statistics and solving inequality $(*)$.

There is no such thing as a watertight method for evaluating an Information Retrieval system [32]. Any existing approach to evaluating or comparing Information Retrieval systems will have to deal with heuristics to some extend only for the reason of this process been highly dependent on human factor. Researchers in the Information Retrieval field have devoted a significant amount of time in developing good, standardized evaluation techniques. The approach proposed here is a standardized approach suitable for a typical Information Retrieval model evaluation. By typical, we mean system that suites common work scenario described in section 5.3, and which evaluation design can be adjusted to the general framework proposed in section 5.1.

Chapter 6

Themis Implementation

In this chapter we will discuss our eTVSM implementation. Of course, the eTVSM implementation itself is essential, however, we see the desired system more like a framework for Information Retrieval models implementation. In other words, we want our system to be a well-defined support structure in which other Information Retrieval models can be organized and developed. This should aid the development of new Information Retrieval models, which are not yet supported by the system and facilitate evaluation and comparison of models through a common testing component.

6.1 What is Themis?

We have chosen "Themis" as the name for our Information Retrieval framework. In mythology, Themis was the Titaness of order and justice, her word was seen as law. Themis is depicted as the blindfolded goddess, seated upon her throne, carrying a sword and a scale. The blindfold symbolizes her impartiality in judgment and setting reward or penalty. The scale was used to measure the deed at hand, and the sword was used to deliver justice.

Figure 6.1: Themis framework logo.

These are the features we expect from an Information Retrieval model. We want to be impartial for setting reward or penalty to terms in a document in a form of weights. At the end we aim to derive a measure of document relevance to assign appropriate rank to each document according to a query. Themis is an in-database data-structure with clear access interface. This data-structure is optimized for storing English natural language document models and performs document similarity judgments based on

the enhanced Topic-based Vector Space Model. As well it includes a database implementation of other fundamental Information Retrieval models, such as VSM. Future implemented Information Retrieval models should correspond to Themis interface and might use the common functionality provided by the framework for their benefit.

6.2 Technology

The implementation technology choice was mainly driven by the following factors:

- *Efficiency*. The selected technology should provide efficient algorithms for handling common Information Retrieval model tasks like sorting, fast key record access, etc;
- *Price*. The price of the technology used should not be a barrier for the end-user. Preferably the technology should be distributed under the open-source license;
- *Compatibility*. The system should be deployable on different platforms. The final implementation should be made to work on multiple computer platforms.

A relational database was chosen for the purpose of acting as a storage container for documents and document models. Moreover, most of the commonly used methods in Information Retrieval model are already implemented in relational databases. The efficiency of these methods is driven by the relational database core competence in these methods. PostgreSQL was chosen as the technology that meets proposed requirements. PostgreSQL is a free object-relational database server (database management system), released under a flexible BSD-style license[1]. PostgreSQL can run on most of the platforms like Windows, Mac OS X or Linux. Since our use of PostgreSQL exceeds just the purpose of a simple storage container, we have decided to implement the system logics in PL/pgSQL. PL/pgSQL (Procedural Language / PostgreSQL Structured Query Language) is a procedural language. It closely resembles Oracle's PL/SQL language. PL/pgSQL extends SQL to a true programming language, and allows much more control, including the ability to use loops and advanced control structures. Though PostgreSQL supports many languages, PL/pgSQL is the only programming language installed by default. So, by using PL/pgSQL we are able to express the framework and Information Retrieval models logics and limit the software installation on an end user system to just a PostgreSQL server. The programming interface to the system (Themis API) is written in Java primarily for the reason of compatibility and licensing cost issues.

6.3 Themis System Architecture

The system architecture is related to component aspect of the structure of a system. In Figure 6.2 we propose the overall system architecture of the Themis framework. It incorporates structural variance for a concrete Information Retrieval model which should allow fitting multiple models into Themis framework. The crawler component is designed as an external one.

Any Information Retrieval model included in the Themis framework should implement three components. These are the Search Engine and the Document Model

[1] Free or open source license that derives from or is similar to the BSD license.

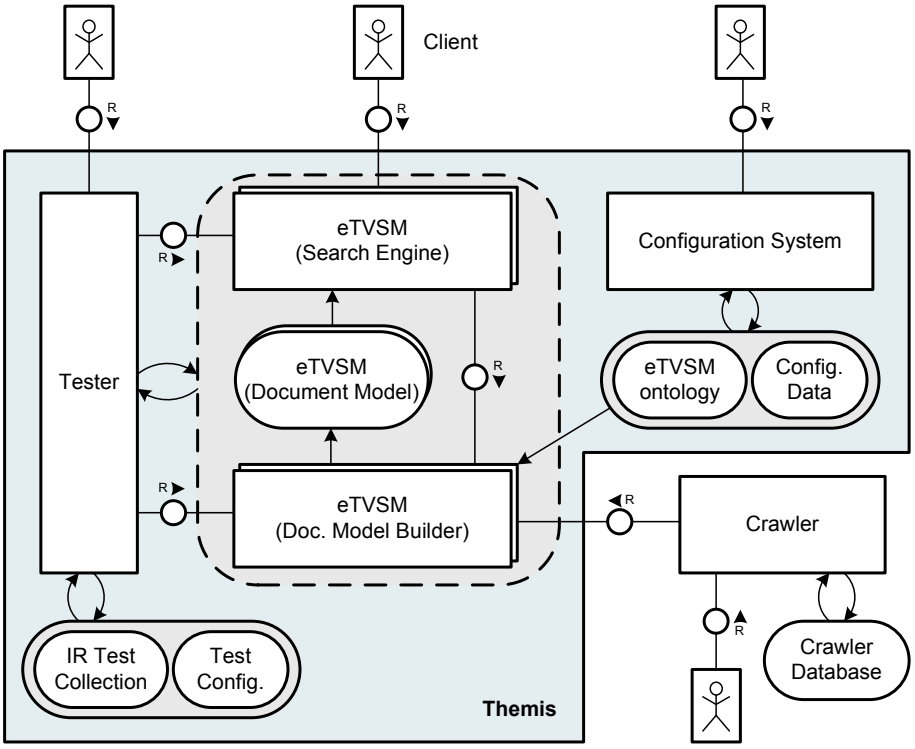

Figure 6.2: Themis system architecture.

Builder components which implement the algorithmic part of the model. The Document Model provides a model specific persistence layer. All together they build up the Information Retrieval model component. A structural variance on Information Retrieval model component should allow unified interaction with Themis environment. This includes a common interface for models evaluation, interaction with the crawler as source of document input and model configuration through a common system configuration repository.

Clients are external components or systems that use Themis API to solve their tasks. Themis API consists of several APIs provided by depicted components. Proposed system architecture was used as the guidelines for Themis implementation. Now, we will discuss all active components of the architecture in detail.

6.3.1 Search Engine

The Search Engine component is a part of the Information Retrieval model component. Its main responsibility is the comparison of documents based on their document models through obtaining documents similarity values. In Figure 6.3 a system architecture extract specifying the Search Engine component and passive/active systems with access types that are relevant to the Search Engine component are shown. The Search Engine component provides a Search API to enable access to the Search Engine internal functionality. The Search API is responsible for:

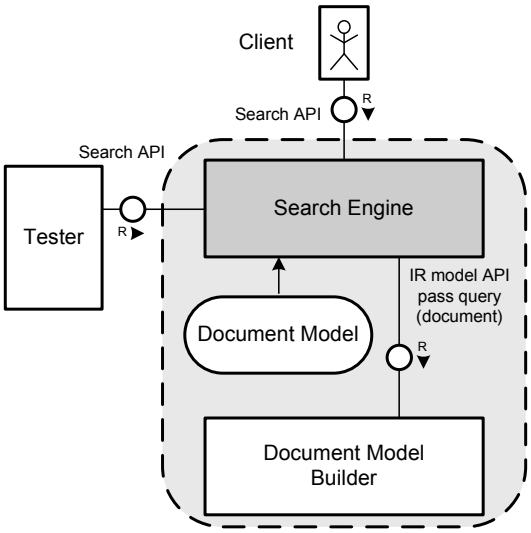

Figure 6.3: Themis Search Engine system architecture.

- Query search: performs the document comparison of the target document with all documents currently stored in the system in the form of document models. Think of this functionality as of a normal Web search scenario of entering a query and obtaining a list of relevant documents ordered by their relevance to the query. The input parameters should be a document identifier (document text or unique document ID, note that queries are treated as regular documents). Query search returns a list of relevant documents at specified range;

- Document comparison: returns similarity of two documents according to the corresponding Information Retrieval model based on two document identifiers (document texts or unique IDs).

The Search API is used by the Tester component to perform Information Retrieval model evaluation. The Search Engine component performs its computations based on the document models which it receives from the Document Model storage. It is important to understand that the Search Engine only retrieves document models from the Document Model storage. How these models appear in the storage is not known to the Search Engine component. The Search API assumes receiving documents or queries (treated as documents in Themis). To be able to acquire access to the corresponding document model the Search Engine component uses an Information Retrieval model API (see section 6.3.2). The Search Engine passes a document to a Document Model Builder component which ensures the existence of a document model in the Document Model storage and will reply with the document model unique identifier.

6.3.2 Document Model Builder

The Document Model Builder component is a part of the Information Retrieval model component. Its main responsibility is the construction of document models as defined by the corresponding Information Retrieval model. Every Information Retrieval model

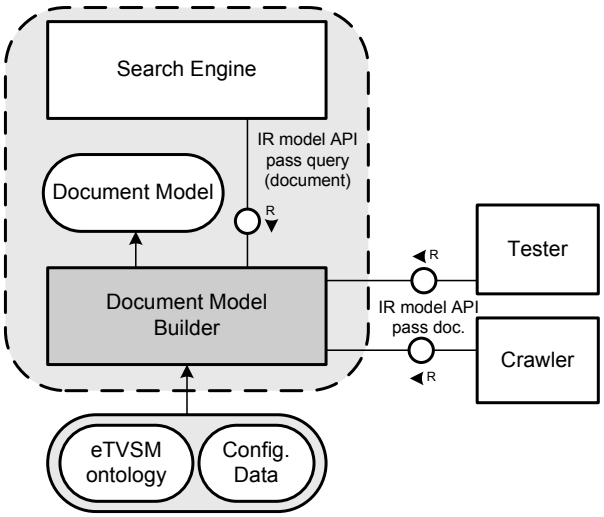

Figure 6.4: Themis Document Model Builder system architecture.

incorporated into the Themis framework is expected to provide an own Document Model Builder component implementation. In Figure 6.4 a system architecture extract is show that specifies the Document Model Builder component and passive/active systems along with access types that are relevant to the Document Model Builder component. The Document Model Builder component provides an Information Retrieval model API which provides the following functionality:

- Document model insertion: constructs a document model for a document so that it corresponds to the needs of the implemented Information Retrieval model. If the document model for the incoming document already exists, it gets updated;
- Document model removal: if there is no further need for keeping a document model in the system this functionality alows the deletion of a document model.

Consumers of the IR model API are the Search Engine, the Tester and the Crawler components. The Search Engine component communicates with the Document Model Builder component to ensure that the query document, which initializes the search, is present in the system in the form of document model. As the prerequisite to a test, the Tester component might require creation of document models for documents from test collections that act as the test workload. The Themis framework assumes the existence of a Crawler component. This component is responsible for the content of the Document Model storage and its update frequency.

Finally, the Document Model Builder work is configured through the Themis Configuration Data storage. Such a configuration data might include information on, e.g. stemming algorithm used, whether stopword removal should be applied, etc. In general, one might recognize common settings applicable to any Information Retrieval model, like in case with document preprocessing steps, and model specific configurations. Because Themis was implemented primarily for the reason of evaluation of the eTVSM, the implementation also includes support for eTVSM ontology (as described in section 3.1). The eTVSM Document Model Builder component instance constructs document models based on the data structure stored in the eTVSM ontology storage.

6.3.3 Configuration System

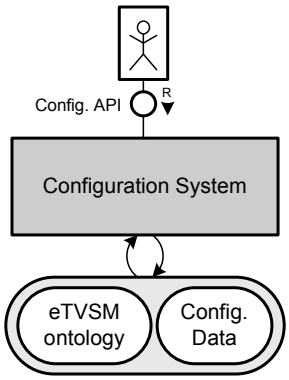

Figure 6.5: Themis Configuration System system architecture.

The Configuration System component provides access to the configuration parameters of the Themis framework. This includes common and model specific configuration parameters. External components gain access to the Configuration System through the Configuration API. Figure 6.5 shows a system architecture extract that specifies the Configuration System component and passive/active systems along with access types that are relevant to the Configuration System component. The Configuration API provides the following functionality:

- Setting common, as well as model specific, configuration parameters in the Themis framework;
- Accessing common, as well as model specific, configuration parameters in the Themis framework;
- Add/Edit/Delete functionality for the eTVSM ontology concepts. This functionality set completely supports eTVSM ontology data structure maintenance.

The Configuration System has a full read/write access to the eTVSM ontology and Configuration Data storages to fulfill its tasks.

6.3.4 Tester

The Tester component is responsible for carrying out automatic Information Retrieval model evaluations, collecting measurements, and aggregating these measurements. In Figure 6.6 you see a system architecture extract that specifies Tester component and passive/active systems along with access types that are relevant to Tester component. The Tester component provides a Test API which provides the following functionality:

- Setting test configuration parameters;
- Accessing test configuration parameters;

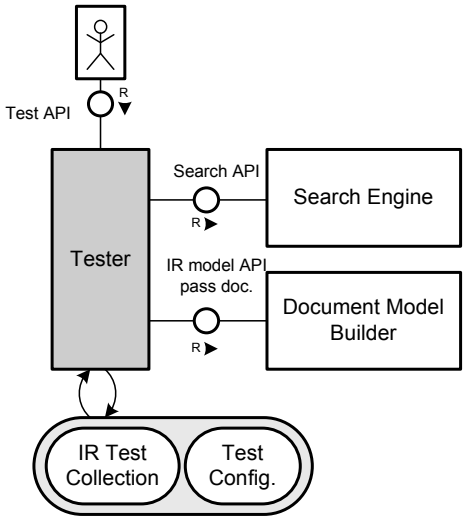

Figure 6.6: Themis Tester system architecture.

- Loading/unloading test collections. This functionality allows loading, editing, unloading Information Retrieval test collections. Tester component uses IR model API provided by Document Model Builder component to perform these tasks. For the description of a common test collection content please refer to section 5.3. Test collections are stored in the corresponding storage system;

- Performing tests. Tester component communicates with Search Engine to conduct experiments. It uses Search API to initialize search and as a respond obtains search results produced by a corresponding Information Retrieval model;

- Collecting measurements. The Tester component supports collecting of measurements and their statistical aggregation described in chapters 4 and 5.

The test procedure carried out by the Tester component can be described by the following guidelines: First, the document models for the documents from the target test collection and the target Information Retrieval model are constructed. Further, the Tester component collects search results for the target test collection queries. Then, the measurements and results of statistical aggregations become available through the Test API. The Test Configuration storage is designed for keeping test runtime configuration parameters. These include model specific settings, e.g. eTVSM ontology automatic construction scheme to be used while testing. Basically, these are the settings that do not influence document preprocessing and work of the Information Retrieval model.

6.3.5 Crawler

The Crawler is treated by the Themis framework as an external component. A crawler is a program or automated script which browses the document collection in a methodical, automated manner. A good example of a crawler is a Web crawler which operates on World Wide Web. In Figure 6.7 a system architecture extract that specifies the

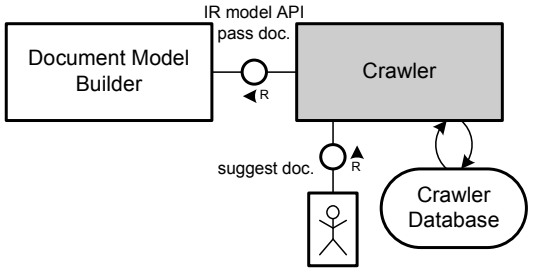

Figure 6.7: Themis external Crawler system architecture.

Crawler component and passive/active systems along with access types that are relevant to the Crawler component is presented. The Crawler has access to the persistent storage (Crawler Database) to save results of its work. In this configuration the crawler discovers new documents by itself, during its work, and provides an API for the external components to "suggest" new documents.

In our scenario the Themis framework expects the Crawler to provide natural language documents along with their URLs on regular basis. Themis might incorporate functionality for extracting content bearing part of a document. Document models are obtained for new incoming documents, while already existing documents get updated.

6.4 Themis Data Model

Above, we have discussed tasks active components of the Themis system architecture are performing. In this section we will discuss the persistence layer these components rely on. A data model is a model that describes in an abstract way how data is represented in an information system, such as Themis.

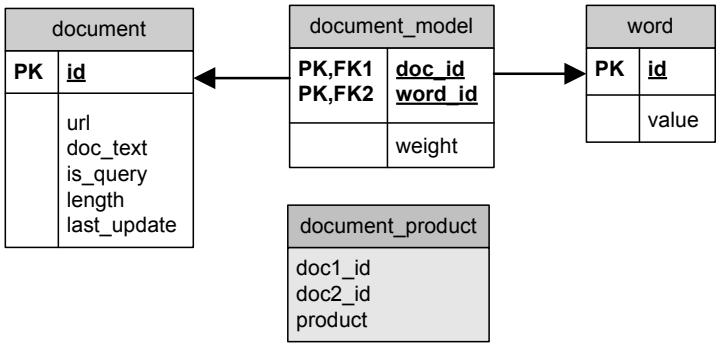

Figure 6.8: Themis VSM data model.

Let us start by taking a look at the document model storage for the VSM. For the description of VSM please refer to section 2.3. The physical schema for the storage of VSM Document Models is shown in Figure 6.8. The `document` entity aims to represent documents which have a VSM document model representation in the sys-

tem. The `url` attribute is used to reference the actual location of the document, while `last_update` attribute specifies when this document was cached last time from the specified location. `doc_text` is the cached document. `is_query` is a flag used to distinguish between documents and queries. Finally, the vector length of the document is stored in the `length` attribute. This value is stored for performance optimization. It is possible to obtain this parameter after the document gets cached. Afterwards, we refer these calculated values while obtaining similarities.

A general search scenario requires obtaining similarity values of a target query with all documents from the Document Model storage. This happens in two steps: First, the query is transformed into a query document—including the calculation of the document vector length. Second, the similarity between the query document and all other documents is calculated by using the stored vector lengths. Since the calculation of the vector length is one half of the similarity calculation algorithm, the usage of stored document vector lengths significantly raises the speed of the calculation.

The `document_model` entity is used to store document models. Here, `doc_id` references a document, `weight` attribute obtains a word weight value according to applied weighting scheme. The word access is done through a reference `word_id` to the `word` entity. In Themis, a word represents the smallest information unit. As stated in section 6.1, Themis is primarily designed for indexing English text documents. This fact allows us to identify words as sequence of characters between common separator characters like spaces[2]. All unique words are then stored as `word` entities. Further, all other entities that encapsulate words, would reference the `word` entity. This forbids multiplicity on the same word concept representation in the Themis framework.

The inner document product is implemented as a database view—a logical unit that represents product between each possible document pair in the system. Together with computation of the `length` attribute in `document` entity, the `document_product` view covers the algorithm of obtaining VSM similarities.

The Document Model storage is Information Retrieval model specific. The physical schema for eTVSM Document Model storage is shown in Figure 6.9. The conceptual difference here, as compared to VSM schema, is the interpretation entity reference in the `document_model` entity instead of word reference. Also, `document_term` entity was introduced to allow automatic judgments while interpretation selection during eTVSM document model construction. For more information on eTVSM document model refer to chapter 3. `nocc` attribute refers to the number of occurrences of the specific term in the specific document.

Contrary to the VSM case, the eTVSM Document Model storage alone does not provide enough information for obtaining eTVSM similarities. What lacks are interpretation similarities, which are the part of the eTVSM ontology storage. `interpretation` and `term` entities, as well as `word` entity discussed before, are also eTVSM ontology concepts.

The Tester component persistence layer is shown in Figure 6.10. The proposed physical schema already incorporates IR Test Collection and Test Configuration storages. Test Collection storage consists of `collection`, `document`, `query`, `relevance` and `stopword` entities. `collection` entity specifies collection name and assigns it a unique key. All other entities reference `collection` to express their relation to a specific test collection. `doc_id` attribute of `document` entity and

[2] This fact does not apply to all the languages. E.g., such approach would not be applicable in case of Japanese or Chinese languages. Especially confusing is Vietnamese, here spaces do not necessary denote a word break. Other languages exist (like Inuktitut) where a word extracting procedure that relies on separator characters would result sentences consisting of a single word.

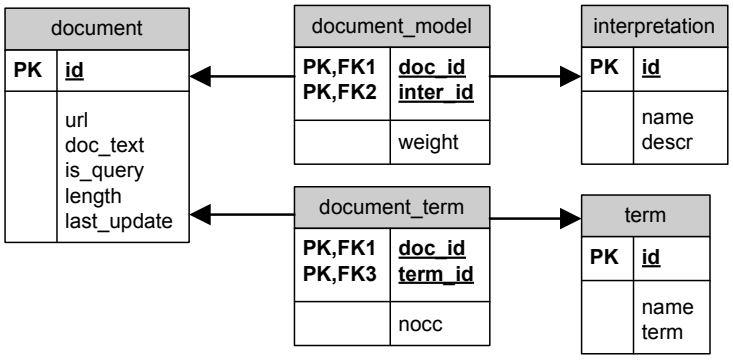

Figure 6.9: Themis eTVSM data model.

`query_id` attribute of `query` entity assign unique entity IDs within test collection.

The `relevance` entity aims to express relevance between a query and a document. The specific `relevance` entity instance will define that a certain document is relevant to the specific query in a given test collection. Some test collections specify suggested stopword list to use while retrieval process. `stopword` entity is modeled specifically for these cases, to store suggested stopwords to use while retrieval. The `test` and `test_cfg` entities are designed to be able to distinguish between different tests. Each test is identified by its name and by parameter values that configure this test. Initially, two test configuration parameters were proposed. These are the start position and the number of retrieved documents that identify search results interval to use in a test. These might be useful if the test is designed to get measurements at a given cut-off rank.

Finally, test run results need to be stored in order to be able to obtain the desired measurements. For this reason we have introduced a notion of a run. Basically, `run` entity specifies that the test with `test_id` was performed on the test collection with `col_id`. `start` and `finish` attributes specify a time span of the test run. Also, one might specify a description in the `descr` attribute. The `search` entity is used to store results of a run. Additionaly to a run reference, the `search` entity includes `query_id`, `doc_id` and `sim` attributes. They specify what similarity value was obtained for a given query-document pair. The `document_map` entity specifies the mapping between document IDs in a test collection and IR model specific Document Model storage. This is required because Document Model storage IDs might not match to the ones defined in a test collection, which consequently causes wrong relevance judgments match.

Now, we will discuss the domain ontology physical data schema (see Figure 6.11) as used by the eTVSM. For the description of the eTVSM ontology concepts and their roles refer to section 3.1. Here, the `topic` entity is used to represent topics of the eTVSM ontology concept. Together with the `map` entity, the `topic` entity defines a topic map (see section 3.1.1). It is then an algorithmic task to ensure that topic map is free of cycles. The `map` entity defines a parent topic to child topic relation in a topic map and sets the specific link type to this relation. The link type is specified in the `map_type` entity. Further, interpretations (`interpretation` entity) are linked to the topics with the help of the `imap` entity. Similarly, terms (`term` entity) are

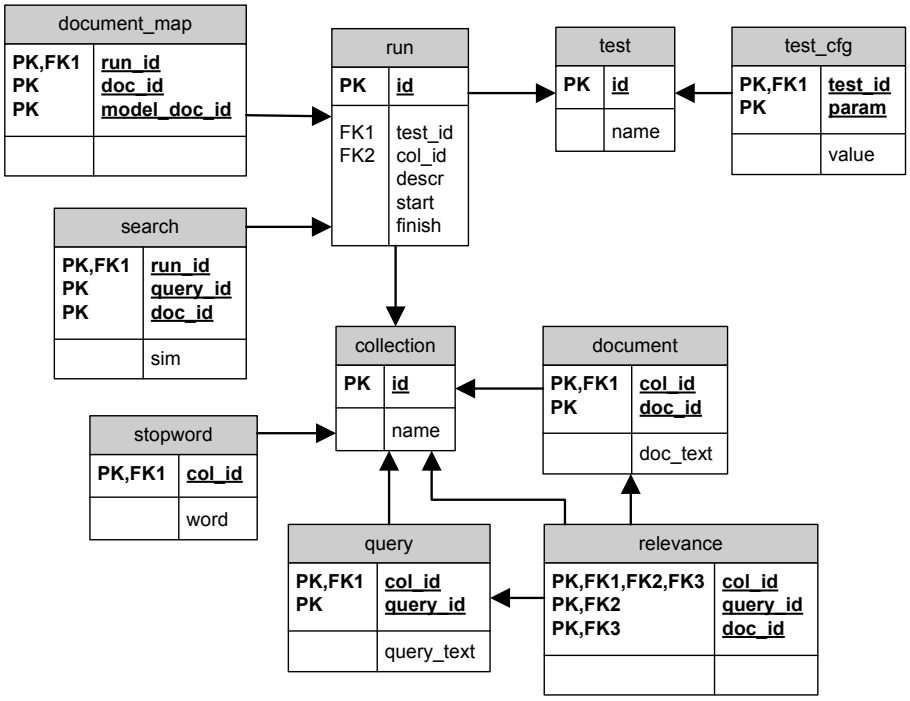

Figure 6.10: Themis Tester data model.

connected to interpretations with the help of the `tmap` entity. `vector` and `ivector` entities are used to store topic and interpretation vectors respectively. It should be algorithmically supported to maintain correct topic and interpretation vectors upon any changes in eTVSM ontology structure as vector values greatly depend on the ontology graph structure.

While obtaining eTVSM document similarities the interpretation similarities are required. Similarly to the case with both vector entities the `isim` entity is used to store pre-computed interpretation similarities which will be then reused while calculation of the document similarities. Finally, the `word` entity is responsible for storing the smallest Themis information units—words. All the word instances contained in other Themis entities that were already presented are the references to the `word` entity. These are, e.g. the `name` attributes of the entities presented in Figure 6.11. Also, the `term` attribute of the `term` entity was designed as the array attribute of elements with type similar to the `id` attribute of the `word` entity (e.g. integer). This allows to represent compound terms (see section 3.1.3) as an ordered list of references to the `word` entity. Contrary to the `word` entity—the `stopword` entity is used to reference words that should be ignored while obtaining documnents similarity values. These are the stopwords that will get filtered out from a document content prior to document model construction. `word_id` attribute is a reference to the corresponding `word` entity designed as a stopword.

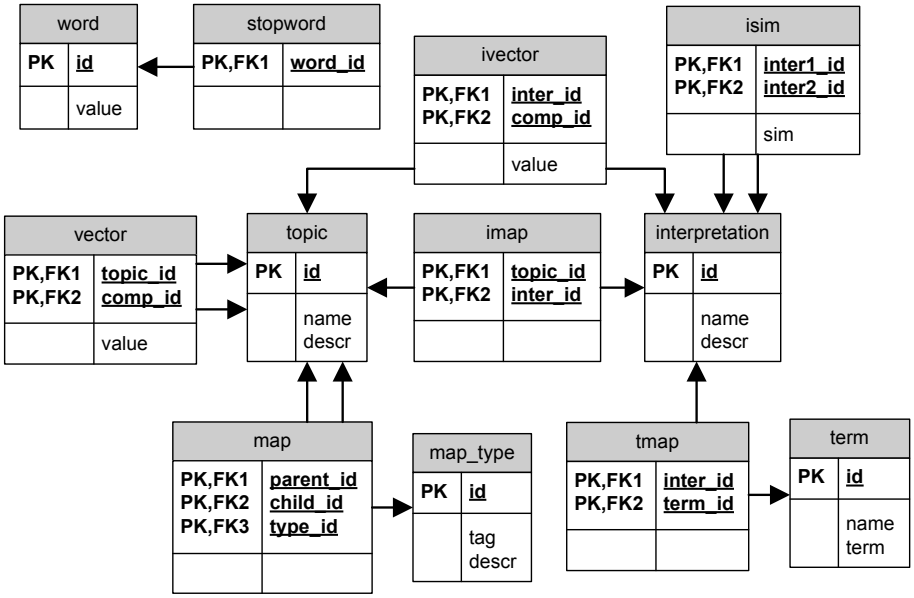

Figure 6.11: Themis eTVSM ontology data model.

6.5 Time vs. Space Complexity Compromise

One of the eTVSM disadvantages is its high computational complexity. An not optimized eTVSM implementation will not be feasible in a real world problem size scenario. The key to the computational problem is that document model consists from eTVSM ontology interpretations. Once new interpretations appear or old get removed from eTVSM ontology, document models for all the documents need to get updated. Also, when computing document similarities, interpretation similarities are used. Interpretation similarity computation is itself a complex task with prior obtaining of topic vectors, what requires non-local study of eTVSM ontology graph structure. In such a situation one has to start looking for a compromise between additional storage for pre-computed intermediate values and algorithm execution time in order to obtain a usable combination of both. Further, one might loose the requirements on data being up-to-date. Time precious tasks, such as response to a user query input, might be performed on "old" data which gets periodically updated in a separate process chain.

We want to present our solution that was implemented in Themis. In Figure 6.12 we propose an integrated view on main Themis process chains. We have applied the eEPC notation. Here, hexagons represent triggering events, rounded rectangles represent activities and shaded rectangles represent information units consumed and produced by activities. Further, we present the description of three main process chains and their dependencies:

- *Ontology control chain*: In this process chain we track changes in eTVSM ontology that influence interpretations similarity values. This chain is initialized by the "*domain ontology is changed*" event. Further, based on the current domain ontology structure we update interpretations similarity values. Interpretation similarities are the pre-computed values that will be used for documents

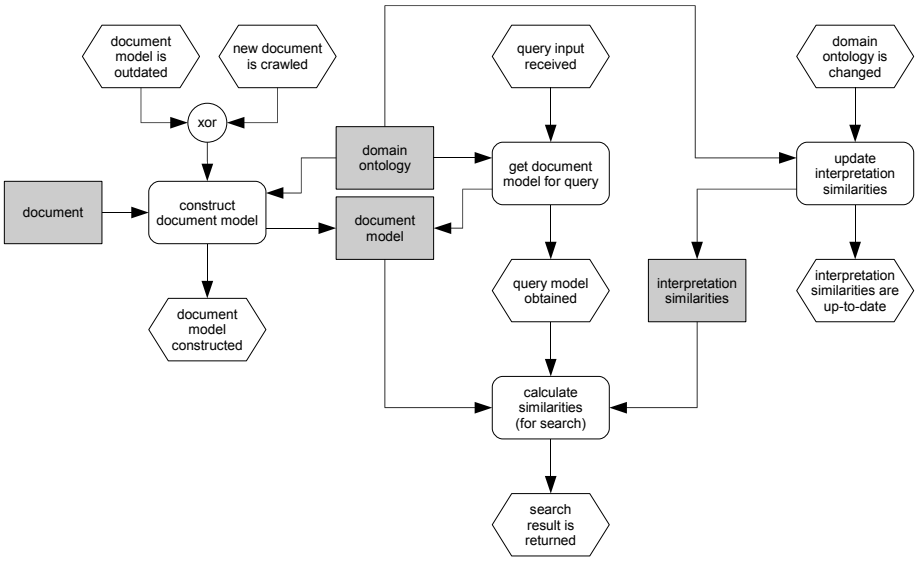

Figure 6.12: Themis event-process chains.

similarity computation. Document models do not get updated synchronously with domain ontology changes;

- *Document model control chain*: This chain gets initialized by one of two events. Either a new document gets crawled by the system ("*new document is crawled*"), or the timeout since last document model update has expired ("*document model is outdated*"). Afterwards, document model gets constructed for the corresponding document based on the current domain ontology configuration. At this point, document model becomes up-to-date with respect to domain ontology;

- *Query search chain*: This chain gets initialized by a user entering a search query into the system. First, a document model for the query is obtained. The difference with document model construction in document model control chain is that the document model is not constructed if the query document model is already present in the system. Here, we assume that the query document model will get updated on regular bases in document model control chain. Thus, a query model is constructed inside this chain only if this query was never sent to the system before. This should decrease response time for already searched queries. After the "*query model obtained*" event, the actual document search is performed.

The presented approach was developed with optimization on accelerating response to user query input in mind. The basic assumption is that domain ontology changes are seldom activities with small effects. In a regular scenario, one usually start with an already constructed ontology allowing minor changes to it over time. In such a case, document model changes are more frequent because of changes in document source rather than because of ontology changes. Thus, it is heuristically reasonable to reduce document model update to only the point of time when it gets re-crawled by the system. Document models are aggregating over time and collectively compose the informative system value. Therefore, the reconstruction task of all document models upon change

in the domain ontology in a short time is not feasible. The assumption of seldom eTVSM ontology changes allows us to store pre-computed up-to-date interpretation similarity values. This results in to a minimum reduction of computational complexity for eTVSM documents similarity calculations.

Chapter 7

eTVSM Evaluation

In the previous chapters we have introduced tasks Information Retrieval deals with. We have looked into approaches that can be applied to solve these Information Retrieval tasks. Approaches presented in chapter 2 included the VSM and the TVSM models. Furthermore, we have introduced a novel Information Retrieval model in chapter 3—the eTVSM. We have provided measurements and methodologies for Information Retrieval models evaluation and comparison in chapter 5. Finally, we have provided hints on implementation of Information Retrieval models, in particular eTVSM, which are the target for the evaluation design described in chapter 6.

In this chapter we want to use the theoretical knowledge that was presented before in order to obtain practical results. We will present our evaluations of the VSM and the eTVSM using different domain ontology configurations. We will limit our presentation to "milestone" eTVSM evaluations and configurations only. We have performed many more evaluations and configuration to incrementally raise the effectiveness of the eTVSM, however due to time and space constraints it is reasonable to put the focus on configurations which either significantly improved the effectiveness or which unexpectedly decreased the effectiveness of the model. Finally we will also compare our evaluations with some other results from the literature, especially with the results for the Latent Semantic Analysis (LSA) model.

7.1 Evaluation Setup

In this section we pursue two goals. First, we want to present the evaluation setup which is used for our evaluations. This includes identification of the test collection, the specification of the document preprocessing steps and their configurations, and document model creation particularities. Another, very important goal that we aim to fulfill is to make evaluations carried in this work comparable with other model evaluations done by other researchers. The matter is that by having access to such an internal model configuration it is possible to provide same evaluations of other models with similar configurations. This will make evaluation results comparable. Unfortunately, our own experience shows that researchers usually provide final results but neglect stating all of the common configuration parameters that greatly influence presented results. At the very end, it is not important which Information Retrieval system performs better on the evaluated measurement parameter, but which performs better in the same configuration environment. E.g. evaluations show that measurement values differ considerably

when incorporating stopword removal (with most common stopwords) as compared to no stopword document preprocessing for the same Information Retrieval model. Same applies if evaluations are done on different stopword sets which are disjoint or intersect instead of being totally equal. Therefore, we are able to perform true inter-model evaluations and comparisons. It's our aim to give other researchers the opportunity to compare their model effectiveness with our results under the same configuration.

To perform our evaluations we will use the Time test collection which was already presented in section 5.4. The Time test collection consists out of 425 documents and 83 queries. These are the articles from the Time magazine. The reason for selecting such a small collection is that we had not much practical experience with eTVSM behaviour and therefore we prefer a small document model evaluation cycle for eTVSM evaluation under different ontology configurations. It was already mentioned in section 3.5 that the domain ontology configuration is a heuristic task which greatly influences model performance. Also, it has to be mentioned that the Time collection consists of about 250000 words and about 20000 unique words. This gives a huge variety of possible eTVSM ontology configurations—what initially is the target of our evaluation. Aside of this, there are many Information Retrieval model evaluations for Time test collection which are available in the literature [10, 22, 50, 34, 56, 35] for comparison.

At the document preprocessing stage there is no letter case distinction. All the text is considered to be in lower case. This does not allow distinction between such terms like *it* (a common stopword) and *IT* (abreviation for "*Information Technology*"). However, this choice was carried out not only for the sake of simplicity, the Times test collection itself does not provide character case distinction. The stopword set which is used for stopword removal is the original Time collection stopword set. It is provided as a part of the Time test collection in a separate file. For the stemmer we have used the Porter stemmer [47].

A weighting scheme is required to assign weights to stem words if using the VSM, or interpretation weights to interpretations in eTVSM (here, document model weights are assumed, not domain ontology interpretation weights which are all set to be equal to one). In our evaluations we have used a plain concept occurrence as the measure of its weight in the document. A separate attention requires the eTVSM document preprocessing. While our evaluations we have used the document model creation approach described in section 3.3 as the guideline. To make the document model creation process more sophisticated we integrated stopword removal as not including stopword terms in eTVSM ontology, this approach was already discussed. Furthermore, stemming of words is performed only if the word is not found in the domain ontology, afterwards term search attempt repeats. Finally, the longest matching term principle is used. All these principles were already discussed in section 3.5.

7.2 VSM Simulation with eTVSM

In chapters 2 and 3 we have shown the evolution of the VSM over the TVSM to the eTVSM. The eTVSM operates on the vector space defined by the modeled topics and allows more degree of freedom as compared to TVSM by introducing intermediate interpretation concepts. Nevertheless, all of the mentioned models are vector based models. Because the eTVSM operational vector space is configurable through the used ontology, VSM can be seen as the specific case of the eTVSM. In order to simulate VSM behavior eTVSM must operate on a specially configured eTVSM ontology. Such

ontology must result in the operational vector space which is compareable to the one of the VSM.

In this section we will show how it is possible to simulate the VSM with the eTVSM and we will perform evaluations of both models. The practical goal that we pursue by such evaluations is to obtain a VSM evaluation measurement which can be later compared to further eTVSM evaluations. Also, we are able to partly check our eTVSM implementation and obtain first eTVSM evaluation measurements. Finally, we can use the methods proposed in section 5.6 to show the insignificant difference between both these models evaluation measurements as a proof of their equivalence.

7.2.1 Formal Simulation eTVSM Ontology Derivation

In this section we will derive an approach for a domain ontology configuration to ensure an eTVSM configuration which behaves identical like the classic VSM. Under identical Information Retrieval models we understand the models which produce identical similarity values for any given pair of documents. The eTVSM document similarity function which was derived in section 3.4 is defined as following:

$$\begin{aligned} sim(d_i, d_j) &= \vec{d_i}\vec{d_j} \\ &= \frac{1}{\left|\vec{\delta_i}\right|}\vec{\delta_i}\frac{1}{\left|\vec{\delta_j}\right|}\vec{\delta_j} \\ &= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\vec{\delta_i}\vec{\delta_j} \\ &= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi}_k \sum_{\phi_l \in \Phi} \omega_{d_j,\phi_l}\vec{\phi}_l \\ &= \frac{1}{\left|\vec{\delta_i}\right|\left|\vec{\delta_j}\right|}\sum_{\phi_k \in \Phi}\sum_{\phi_l \in \Phi} \omega_{d_i,\phi_k}\omega_{d_j,\phi_l}\vec{\phi}_k\vec{\phi}_l \end{aligned}$$

with $\left|\vec{\delta_i}\right|$ defined as:

$$\begin{aligned} \left|\vec{\delta_i}\right| &= \left|\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi}_k\right| \\ &= \sqrt{\left|\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi}_k\right|^2} = \sqrt{\left(\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k}\vec{\phi}_k\right)^2} \\ &= \sqrt{\sum_{\phi_k \in \Phi}\sum_{\phi_l \in \Phi} \omega_{d_i,\phi_k}\omega_{d_i,\phi_l}\vec{\phi}_k\vec{\phi}_l} \end{aligned}$$

Now, let us assume interpretation similarities to be such that:

$$\vec{\phi_i}\vec{\phi_j} = \begin{cases} 1, & when \;\; i = j \\ 0, & otherwise \end{cases}$$

Under such conditions $\left|\vec{\delta}_i\right|$ can be obtained as:

$$\begin{aligned}\left|\vec{\delta}_i\right| &= \sqrt{\sum_{\phi_k \in \Phi} \sum_{\phi_l \in \Phi} \omega_{d_i,\phi_k} \omega_{d_i,\phi_l} \vec{\phi}_k \vec{\phi}_l} \\ &= \sqrt{\sum_{\phi \in \Phi} \omega_{d_i,\phi}^2}\end{aligned}$$

Interpretation similarities assumption will leave only the items in our sum for which $k = l$. Finally, following the same considerations as in the previous step we obtain eTVSM similarity function as:

$$etvsm.sim(d_i, d_j) = \frac{\sum_{\phi \in \Phi} \omega_{d_i,\phi} \omega_{d_j,\phi}}{\sqrt{\sum_{\phi \in \Phi} \omega_{d_i,\phi}^2} \sqrt{\sum_{\phi \in \Phi} \omega_{d_j,\phi}^2}}$$

This function resembles the function for obtaining VSM document similarities:

$$vsm.sim(d_i, d_j) = \frac{\sum_{t \in T} \omega_{d_i,t} \omega_{d_j,t}}{\sqrt{\sum_{t \in T} \omega_{d_i,t}^2} \sqrt{\sum_{t \in T} \omega_{d_j,t}^2}}$$

with the only difference of having document interpretation weights instead of document stem word weights. However, the principles of obtaining these weights are similar in both models (this can be plain document occurrence or *tf-idf* approach discussed in section 2.3 or any other). So, by ensuring interpretation concepts from the domain ontology of the eTVSM to be identical to the word stem concepts from the VSM, we ensure models to be identical. This, as well as interpretation similarity assumption stated before, can be realized by an ontology modeling approach shown in Figure 7.1.

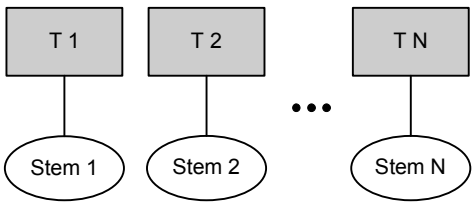

Figure 7.1: VSM simulation eTVSM ontology.

Such a modeling unit which consists of a topic, an interpretation and a term should be created for each word stem recognized by the VSM. Term is exactly the VSM word stem, while the interpretation and the topic are "dummy" concept instances. The fact that there are no inter-topic relations ensures interpretation similarities to be orthogonal towards each other. Also, eTVSM interpretations now correspond to VSM word stems with one-to-one relation. An eTVSM which operates on an ontology which was constructed using the principles shown in Figure 7.1 simulates VSM.

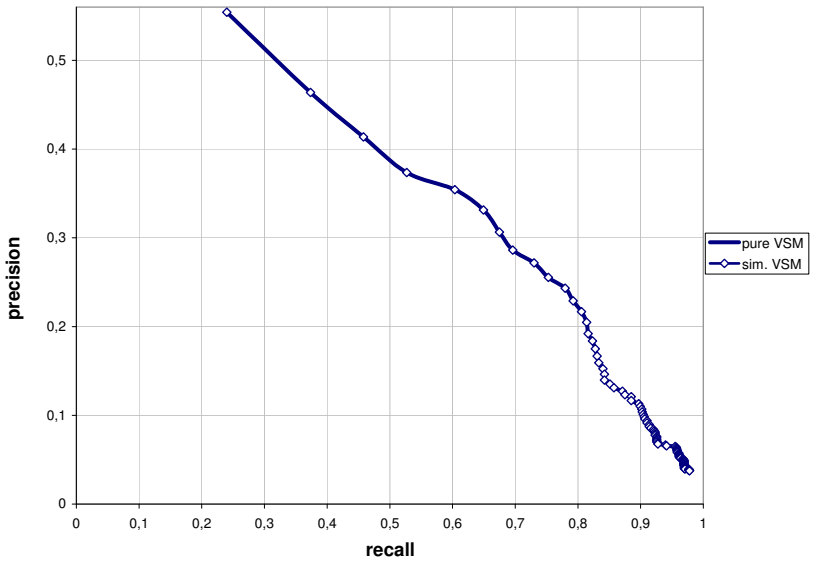

Figure 7.2: VSM simulation recall-precision plot.

7.2.2 Evaluation of VSM Simulation with eTVSM

The preparation phase of our evaluation includes loading of the Time test collection into Themis through the Tester component. The comparison procedure requires us to perform evaluation of both Information Retrieval models on the same test collection. In case of eTVSM we require to configure a domain ontology following the principles that were presented and theoretically grounded in section 7.2.1. The VSM does not require any additional configuration as long as all preprocessing steps are configured analogously to the eTVSM.

Figure 7.2 presents the recall-precision plot as it was described in section 4.2. Here, we have plotted the average over all queries (recall, precision) points for the corresponding measurements taken at a given cut-of rank $r \in [1, \ldots, 100]$. Both plots (for VSM and eTVSM) coincide to give a path shown. Minor differences in measurements might be the result of rounding performed along the calculations, and different calculation schemes applied for both models. In general, it is possible to state that both such models are identical. Also, as you can see, the general inverse relation between recall and precision remains.

We have constructed the precision at standard recall levels plots (this was described in section 4.2) for both of our evaluations. They are presented in Figure 7.3. The visual comparison of both plots also speaks for the identity of both evaluated models. These first evaluation measurements we will take as the starting point for our comparisons with the further eTVSM evaluations we are going to perform in next sections. We intend to improve these measurements by introducing inter-term semantic relations in domain ontology used by the eTVSM.

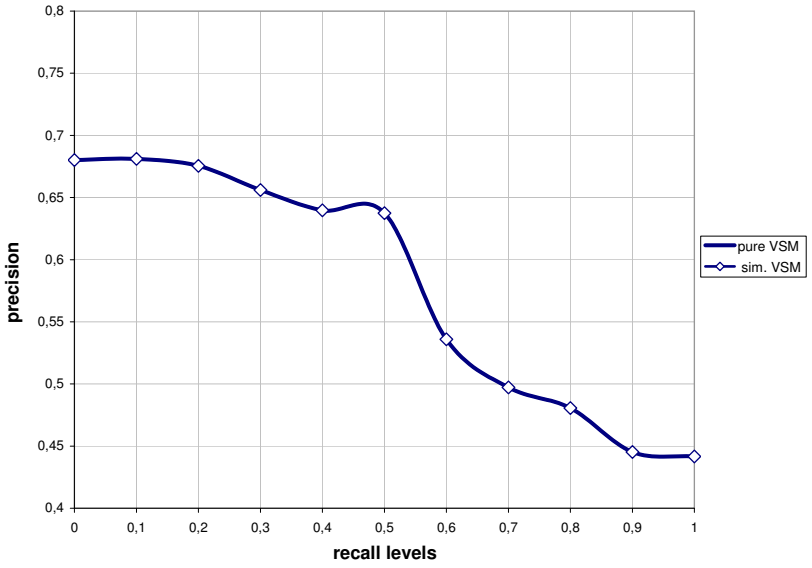

Figure 7.3: VSM simulation precision at standard recall levels plot.

7.3 Synonymy eTVSM

We have taken the name "synonymy eTVSM" for an eTVSM which operates on an ontology designed to represent synonymy term relations. The eTVSM ontology total synonymy modeling pattern is presented in Figure 7.4. We expect to use external synonymy knowledge for an automatic ontology construction. Of course, we expect new evaluation results to outperform the ones for VSM we have obtained so far, since the resolution of synonyms should help in detecting new similarities between terms which haven't been recognized before.

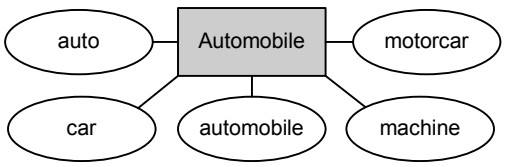

Figure 7.4: eTVSM ontology total synonymy modeling.

Synonymy is the fact that many equivalent or closely related meanings can be conveyed by distinct words. This means any strategy that simply uses string matching to select documents which contain terms also found in the user's query is bound to miss many relevant documents. These are the documents that use another terms with the similar meaning as the terms specified by the user. Synonyms (alternative names for the same concept) are well-known feature of natural languages. Together with spelling differences synonym variations cause many problems in Information Retrieval. If an

Information Retrieval system is not capable of synonymy recognition it will most probably see no similarity in document containing terms like *car* and *auto*. Standard approaches to solve this problem are query expansion mechanisms [9, 31, 43, 20]. A query expansion-enabled interface will take as input a given search term, look for synonyms in the controlled vocabulary, and return documents that match either the search term or any of its synonyms. In our case, we will use an somehow similar approach; conveniently eTVSM already incorporates a container for synonymy vocabulary which can be expressed as a part of the ontology used by the eTVSM. Intuitively, integration of synonymy into Information Retrieval should increase model effectiveness. Later, in this section, we will check our guesses by performing concrete evaluations.

7.3.1 WordNet as Source of Semantics

We have already shown how it is possible to simulate a VSM with an eTVSM in section 7.2. This can be achieved by applying a specially modeled "dummy" ontology. Such ontology assumes all terms to be orthogonal—inter-independent. However, the eTVSM potential is hidden in its ability of representing term relations and to model term meaning relations as used by humans. For a small sized ontology, like one shown before in Figure 3.14 concept relations can be modeled manually. Similar examples of an ontology modeling can be found in [36]. But, such approach does not suit our needs. It is not feasible to model semantic relations manually for some thousands of terms (which are potentially present in Time test collection or in any real world scenario) in a short time. Moreover, the complexity multiplies with the need of ontology re-modeling for eTVSM re-evaluation with applying different ontology modeling approaches.

An alternative to manually modelling an ontology is the reuse of existing ontologies or lexicons. WordNet is a semantic lexicon for the English language. It groups English words into sets of synonyms called synsets, and it provides short, general definitions, and records the various semantic relations between these synonym sets. WordNet was created and is being maintained at the Cognitive Science Laboratory of Princeton University under the direction of psychology professor George A. Miller. Development began in 1985. Since then WordNet has grown to include 117597 synsets which include a total of 207016 word-sense pairs. WordNet can be interpreted and used as a lexical ontology in the computer science sense. In our scenario, we want to reuse this accumulated knowledge. We aim at an automatic domain ontology construction by reusing the WordNet. This can be achieved by extracting terms and their semantic relations from the WordNet lexicon and loading them into an ontology suitable for the eTVSM.

The WordNet database and software tools have been released under a BSD style license and can be downloaded at [4] and used freely. The database can also be browsed online [7]. There exist many related projects, primarily artificial-intelligence solutions and access APIs. One of these APIs is of particular interest to us. This is the PostgreSQL API proposed within WordNet SQL Builder project [5]. We have used WordNet version 2.1 and loaded it into a PostgreSQL database schema to unify the domain ontology and WordNet storage. To allow an automatic ontology construction, a transformation script was written which mapped WordNet concepts to ontology concepts. In Figure 7.5 we provide the WordNet schema extract proposed by WordNet SQL Builder project which is relevant to our needs.

The `synset` entity is used to store WordNet synsets. The `semlinkref` entity is used to define semantic links between any pair of two synsets. The semantic link type

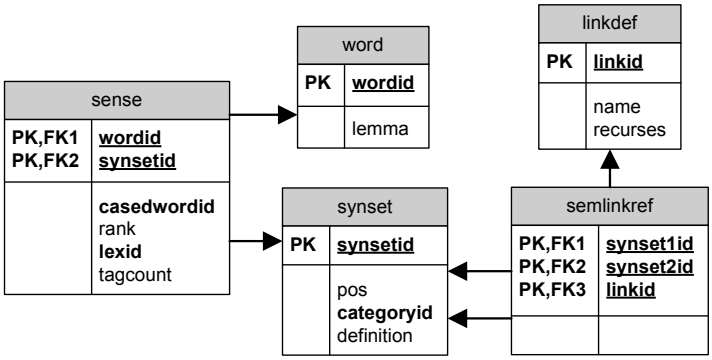

Figure 7.5: WordNet SQL Builder schema extract.

is defined in the `linkdef` entity. The `word` entity is used to store terms—these are terms in the sense of the eTVSM as presented in section 3.1.3. These terms are ordered to appropriate synsets through the `sense` entity. Each term can appear in multiple synsets, as well as each synset can include multiple terms. `rank` property of `sense` entity is used to assign a rank to a term-synset relation. E.g. term *car* is assigned to five synsets (has five senses). The most highly ranked sense is "*a motor vehicle with four wheels; usually propelled by an internal combustion engine*" which is assigned a `tagcount` of 598. The next ranked sense (`tagcount` of 24) for the term *car* is "*a wheeled vehicle adapted to the rails of railroad*". `tagcount` is a measure of term sence usage frequency. Synset descriptions are defined in the `definition` attribute of the `synset` entity. Finally, the `pos` property of a `synset` entity is used to define a part of the speech for the terms which are grouped in the synset.

7.3.2 Synonymy eTVSM Ontology Modeling Approach

In this section we want to present a possible principle of automatic domain ontology construction which exploits the synonymy semantic relations given by the WordNet. This approach has caused our Information Retrieval model effectiveness measurements to improve as compared to the VSM evaluation. The starting point for our synonymy eTVSM approach is the "dummy" ontology as already presented in section 7.2.1. The prior approach did not assume any semantic relations between terms. We will now start to narrow this gap by automatically assigning synonymy relations from the WordNet between terms recognized while the evaluation presented in section 7.2 when we were simulating VSM with eTVSM. The following pseudocode for adding new terms from the target test collection into the synonymy ontology has to be applied on each term recognized in the document base:

```
1: IF term (T) EXISTS IN eTVSM ontology RETURN
2: ELSE
3:   IF term NOT IN WordNet
4:     create dummy eTVSM ontology for term (T)
5:   ELSE
6:     select most highly ranked synset/sense (S) for term (T)
7:     create linked topic and interpretation for synset (S)
8:     create term (T)
```

```
9:      link term (T) to interpretation (S)
```

In case when a term was not found in the WordNet database (line 3) the "dummy" ontology consisting of linked topic, interpretation and term (like in the case of the VSM simulation) has to be be created (line 4). Otherwise, we create a new term and link it to a new or existing most highly ranked WordNet synset equivalent interpretation. This procedure has to be repeated for each term in the following priority order: noun, verb, adjective and then adverb. This assures that we check all parts of the speech in the order of most common WordNet synsets. This is required because some terms can represent different parts of the speech, e.g. *likely* can be either adjective or adverb. First, we look for the most content bearing parts of the text—nouns, further moving to the term senses that have less content weight. Such procedure assures that each term is present only once in the resulting eTVSM ontology with exactly one sense (one link to interpretation). Moreover, in each case we take only the most highly ranked synset (most common term sense). Finally, all these measures result in an eTVSM ontology free from semantic non-determinism which extracts most of the synonymy knowledge from WordNet.

7.3.3 Evaluation of Synonymy eTVSM

The preparation phase of synonymy eTVSM evaluation as compared to VSM simulation differs only in a model preparation activity, in particular the eTVSM ontology construction. Figure 7.6 presents the precision at standard recall levels plot for the synonymy eTVSM evaluation (*syn eTVSM*) superposed on the same type plot for the VSM evaluation (*VSM*).

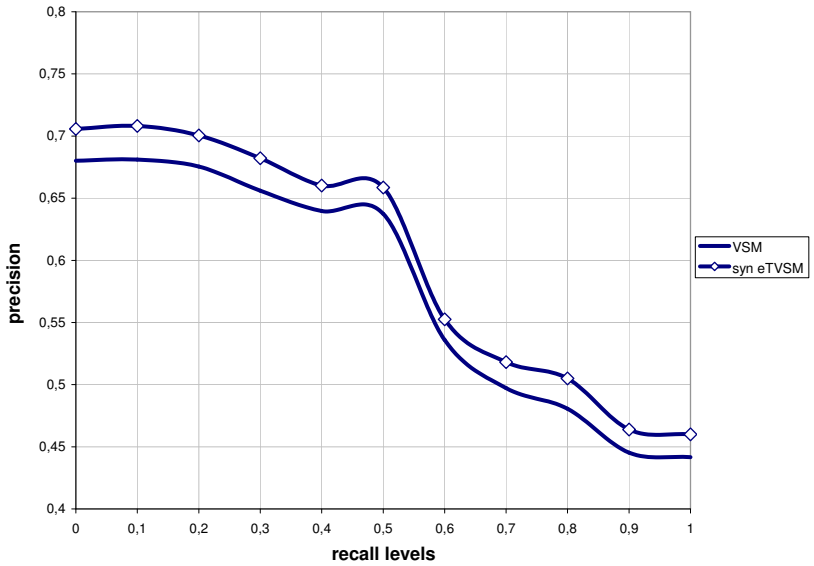

Figure 7.6: Synonymy eTVSM precision at standard recall levels plot.

Visually the synonymy eTVSM has a higher precision at all eleven standard recall levels. Furthermore, we will perform a comparison of the VSM and the synonymy

eTVSM following the procedure proposed in section 5.6.2. For the measurements, upon which to perform comparison, we have chosen precision and recall measurements. These measurements were collected for each query from the test collection at a cut-off rank equal to the number of relevant documents proposed by the test collection for this query. In such a setup the number of relevant documents and the number of documents retrieved is always equal. Thus, precision is equal to recall, resulting in a single measurement value for each model-query pair. Measurements were collected for both models.

According to the comparison procedure we have constructed confidence intervals for a difference in compared measurement values. We have decided to construct confidence intervals for $\mu = \mu_1 - \mu_2$ where μ_1 is the mean measurement value for synonymy eTVSM and μ_2 is the mean measurement value for VSM for different confidence levels α. The computational results are provided in Table 7.1. The intervals were obtained based on computed estimators $\hat{z} = 0.0177$ and $S^2 = 0.0285$. According to the applied comparison procedure we can conclude that synonymy eTVSM performs better than the VSM with respect to the compared measurement value when confidence interval is a positive interval not containing zero. In Table 7.1, you can find that such a state is achieved when confidence level α is raised to 0.35.

α	$t_{82,1-\frac{\alpha}{2}}$	I_α
0.01	2.6371	$[-0.0312, 0.0665]$
0.05	1.9893	$[-0.0192, 0.0545]$
0.10	1.6636	$[-0.0131, 0.0485]$
0.15	1.4531	$[-0.0092, 0.0446]$
0.20	1.2920	$[-0.0062, 0.0416]$
0.25	1.1586	$[-0.0038, 0.0391]$
0.30	1.0430	$[-0.0016, 0.0370]$
0.35	**0.9400**	[**0.0003, 0.0351**]

α—confidence level, t—Student-t distribution quantile, I—confidence interval.

Table 7.1: VSM and synonymy eTVSM comparison—confidence intervals.

The statistical generalization concludes that in a sequence of similar comparisons of the VSM and the synonymy eTVSM on similar test collections the synonymy eTVSM will perform better with respect to compared measurement value on average in 2 out of 3 cases. In 1 out of 3 cases it will be not possible to give statistically grounded preference to the synonymy eTVSM over the VSM.

7.4 eTVSM with Semi-automated Ontology

The previously presented Synonymy eTVSM assumes that we define sets of terms; all the terms inside such set are considered to be completely interchangeable in a semantic sense. In synonymy eTVSM all the terms from one such set get connected to one corresponding interpretation and further to a topic. So, terms can be either considered totally similar or totally not similar. However, eTVSM allows expressing intermediate similarity levels between terms by setting relations between topics on the topic map level. This opens new possibilities for expressing semantics to further improve Infor-

mation Retrieval effectiveness of eTVSM. This is, however, a hypothesis that should be checked.

7.4.1 Why Semi-automated?

Of course, a completely automatic approach to eTVSM ontology construction which assures high model effectiveness is a desired solution. This would mean that an eTVSM based system can be deployed and configured to right away deliver highly effective results. These results would exploit term semantic relations obtained from WordNet and are expected to compete or even outperform any existing Information Retrieval system. We have performed numerous evaluations which included different topic map configuration patterns. Just to give an idea, we have tried to load hyponymy and meronymy relations both and each separately, we have used the domain term relation presented in WordNet, we have tried the approach presented in [36] for automatic eTVSM ontology loading from WordNet. The results of our attempts are summarized in Figure 7.7. Here, with dashed lines, we have presented the general guiding lines for the precision at standard recall levels plots we have obtained during our evaluations.

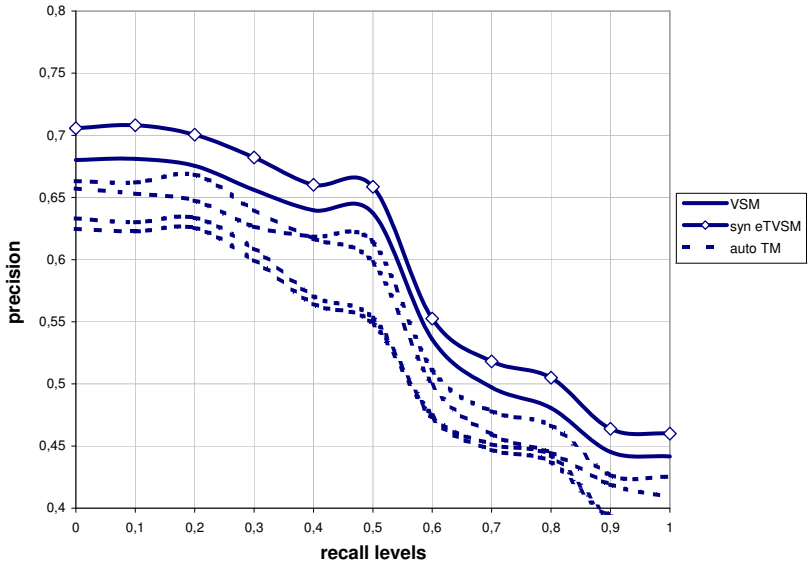

Figure 7.7: Guiding lines for precision at standard recall levels plot in case of eTVSM with automated ontology evaluations.

It is visually distinguishable that our totally automated ontology construction approaches result in a model that performed worse than synonymy eTVSM or even VSM. The conclusion, which needs to be done is that it is hard to find a full automated approach that provides good results. Moreover, even if we find such an approach its benefits are doubtful for the use on other test collection or in general case. The reasons for this, as we can identify them, are:

- WordNet is a general purpose ontology. Specific domain knowledge is not present in WordNet; therefore, it has no chance to get into a domain ontology. E.g. the

Time collection which is composed from the articles from the Time magazine of the period of Cold War. To be more precise its articles cover the period of 1950s'–1960s'. WordNet, however, does not represent historical knowledge of that time in sufficient level of details. E.g. a query like "*signing of the test ban treaty.*" which is query number 60 in Time test collection demonstrates this problem. The term *test ban* is present in WordNet. It is also possible to discover through hyponymy relation that *test ban* is a kind of a *ban*, *prohibition*, *refusal*, *denial*, etc. It is, however, not possible to figure out that this ban has to do with a ban on testing of nuclear weapons and what countries were involved in this treaty. Such information would be quite useful for retrieving relevant documents in this particular case;

- WordNet does not include all terms. E.g. a term like *Ngo Dihn Diem*—"*first President of the Republic of Vietnam*" is not present in WordNet, but is quite often within Time test collection documents and queries. The manual study of Time test collection suggests of introducing synonymy relation between terms *Ngo Dihn Diem* and *President Diem*. Furthermore, any relation of *Ngo Dihn Diem* to *Vietnam* is also missing.

These observations led us to the idea to come up with a list of heuristic recommendations for constructing eTVSM ontology automatically or, if prior is not always possible, manually. In such conditions domain ontology construction can be seen as the symbioses of two: automatic loading term relations from WordNet and later human semantic enrichment which addresses described earlier problems. Because of this human intrusion, we will address the resulting ontology as the one obtained through a semi-automated procedure.

7.4.2 Semi-automated eTVSM Ontology Modeling Approach

After deciding to allow human intrusion into the domain ontology construction process we will go again a step back and take the ontology obtained on the synonymy eTVSM step as the starting point for a new ontology. We will now state a list of hypotheses we consider relevant for improving the effectiveness of eTVSM through semantic term relation enrichment of our initial ontology:

- *Most of the semantic knowledge, valuable for Information Retrieval model effectiveness, is concentrated in queries.* The primary reason for this hypothesis is the fact that queries are the initiators of retrieval process. We, users of an Information Retrieval system, formulate our information need in form of queries. We do this by encoding in queries all the content and semantic information that describes our needs. In regular Information Retrieval scenario queries are short documents, significantly shorter than documents we intend to retrieve. Thus, by reducing human activities only to operations on queries we significantly decrease required human effort in ontology construction assistance, and at the same time maximize these effort effectiveness;

- *All compound terms that occur in queries should be recognized.* Compound terms were already discussed in section 3.1.3. Failing to recognize such terms as, e.g., *test ban*—"*a ban on the testing of nuclear weapons that is mutually agreed to by countries that possess nuclear weapons*" or *Ngo Dihn Diem*—"*first president of the Republic of Vietnam*" from Time test collection might greatly

distort intended query semantics which consequently will deliver documents relevant to the misjudged user intention;

- *Each recognized term should be a part of a synonymy group.* Benefits of synonymy were already described and grounded in section 7.3. If there exist total synonymy terms for the terms that appear in queries—they should be modeled;

- *Semantic relations between terms from the same query should be modeled.* If there exist any semantic relations that were presented in section 2.2 between any pair of terms from the query they should be modeled. This should potentially increase the chance of retrieving documents that include similar semantic relations—which is an intended behavior;

- *Specific domain knowledge for terms from queries should be modeled.* Some terms might intend to define not a general purpose meaning, but a meaning specific to some domain. E.g., term *hot line* in WordNet is described as "*a direct telephone line between two officials*". However, within Time test collection the intended meaning for term *hot line* is the concrete "*direct telephone line between the White House and the Kremlin*". The knowledge about concrete official users might be valuable for this concrete test collection. Similar, the term *Ngo Dihn Diem* might be modeled to possess relations with *Vietnam* and *President* terms.

The above presented are the most important factors that we believe should be included into an eTVSM ontology to ensure high retrieval effectiveness. These should be treated as suggested guidelines to eTVSM ontology construction. If it is possible this approach is preferred to be performed automatically, if not, it might be partially automated and further finalized through human assistance.

7.4.3 Evaluation of eTVSM with Semi-automated Ontology

In the previous section we have produced hypotheses that should improve eTVSM effectiveness. In this section we want to derive practical justification or rejection of these hypotheses by performing evaluations. The preparation phase for eTVSM evaluation will now consist of two stages. First, we have to perform preparation we have accomplished for the synonymy eTVSM (see section 7.3.2), afterwards we will enrich the obtained ontology by following the guidelines presented in section 7.4.2. In order to demonstrate the application of these guidelines in action we present some concrete examples from Time test collection. We will concentrate our attention only on issues relevant to stated hypotheses:

1. **Query** 60: "*Signing of the test ban treaty*."

   The proposed eTVSM ontology constructed according to descrbred principles for query 60 is shown in Figure 7.8.

   The particularity of such a query is that after stemming and stopword removal it consists of four words: *sign*, *test*, *ban* and *treaty*. And, that the most content bearing part of the query is encoded in a single compound term—*test ban*. Failing to recognize this term leads to query misunderstanding. Here, we have also assigned a semantic relation between a pair of terms from query. It specifies that *test ban* is an instance of a *treaty*. Term *treaty* is assigned synonym *pact* obtained from WordNet.

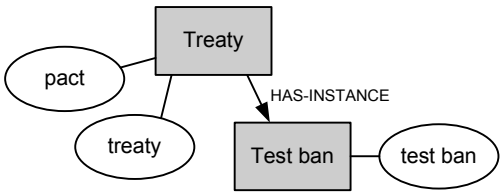

Figure 7.8: Query 60 from Time test collection enrichment ontology extract.

2. **Query** 2: "*Efforts of ambassador Henry Cabot Lodge to get Viet Nam's President Diem to change his policies of political repression*."

   Proposed eTVSM ontology constructed according to descrbred principles for query 2 is shown in Figure 7.9.

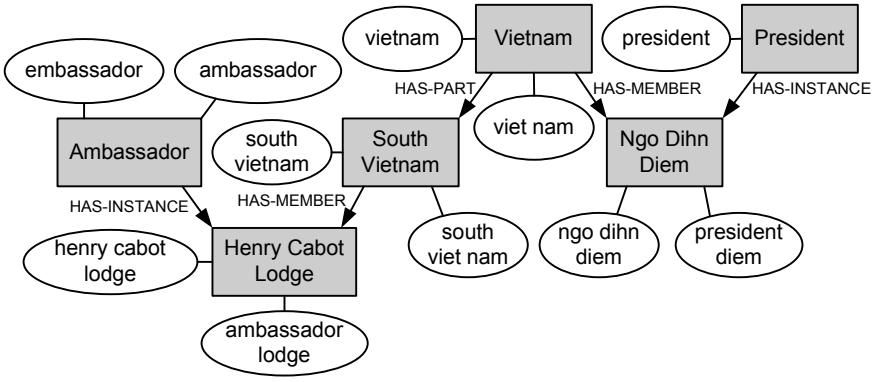

Figure 7.9: Query 2 from Time test collection enrichment ontology extract.

The compound terms that should get recognized in this query are: *Henry Cabot Lodge*, *Viet Nam*, *President Diem*. Further, we should apply domain knowledge to represent that *Henry Cabot Lodge* was the ambassador in *South Viet Nam* and *Ngo Dihn Diem* was the *President* in *Viet Nam*. To extend relations between terms in the query we might relate *South Viet Nam* to *Viet Nam* as its part. For relations modeling in a query like this it is quite likely that the human participation would be required.

Recall-precision Plot

For our evaluation we have obtained eTVSM ontology by first performing automated synonymy eTVSM ontology construction. We have then added all additional relations manually. Nevertheless, it is possible to perform portion of the task we have performed automatically by loading corresponding WordNet concepts. We have followed to the proposed guidelines and performed ontology construction similar to example queries 2 and 60 we have presented from Time test collection. It must be mentioned, that as far as proposed guidelines can not be seen as strict instructions; there is a notion of heuristics in our procedure. However, we believe that any rational work in proposed

direction should deliver improvement in results. Figure 7.10 graphically presents recall and precision measurements we have obtained at this stage of evaluation.

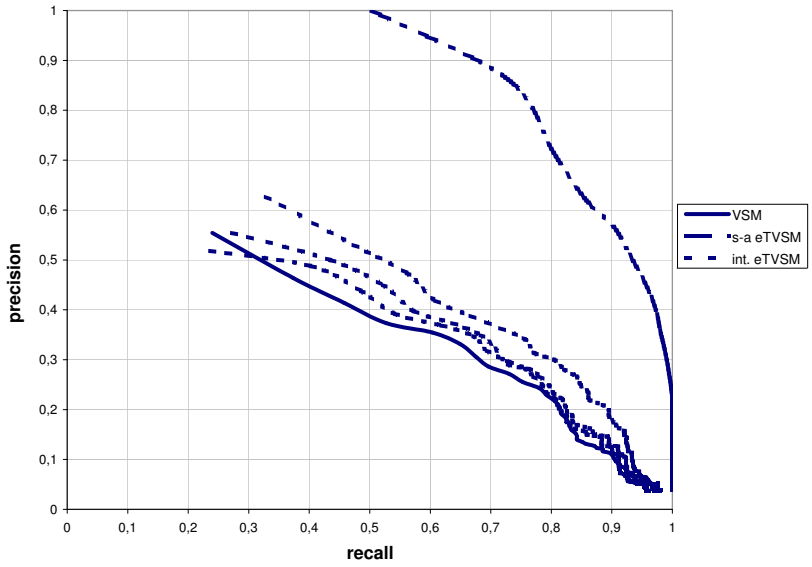

Figure 7.10: eTVSM with a semi-automated ontology recall-precision plot.

With a solid line, we have plotted measurements obtained while VSM evaluation (*VSM*). This line can be seen as a benchmark for further comparisons. At the early stages of eTVSM ontology construction we have performed evaluation to check how eTVSM behaves. We have performed eTVSM re-evaluation after each 2–3 queries we have used to derive term relations. Measurements for some initial intermidiate evaluations are shown with dashed lines (*int. eTVSM*). The behavior of recall-precision plot with respect to number of processed queries assured us that we are on the right way. We have presented the recall-precision plot for our final evaluation with eTVSM ontology after performing processing of approximately half of Time test collection queries (111 additional term relations were added manually for about 16000 terms in total). It is shown with dashed line of variable-dash line lengths (*s-a eTVSM*—closest to the right-top corner). Regardless of quite considerable visual improvement in recall-precision plot, we believe that not actual numbers are of real value, but a general tendency of model effectiveness behavior which tends to the $(1, 1)$ recall-precision measurement values combination.

Precision at Standard Recall Levels Plot

We have constructed the precision at standard recall levels plot for our final eTVSM evaluation. You can see it in Figure 7.11 superposed on plots for prior evaluations: VSM simulation (*VSM*) and synonymy eTVSM (*syn eTVSM*). It is shown with line connecting solid-filled circles that represent actual precision values (*s-a eTVSM*). As you can see, eTVSM with ontology configuration derived in this section delivers better precision at all standard recall levels.

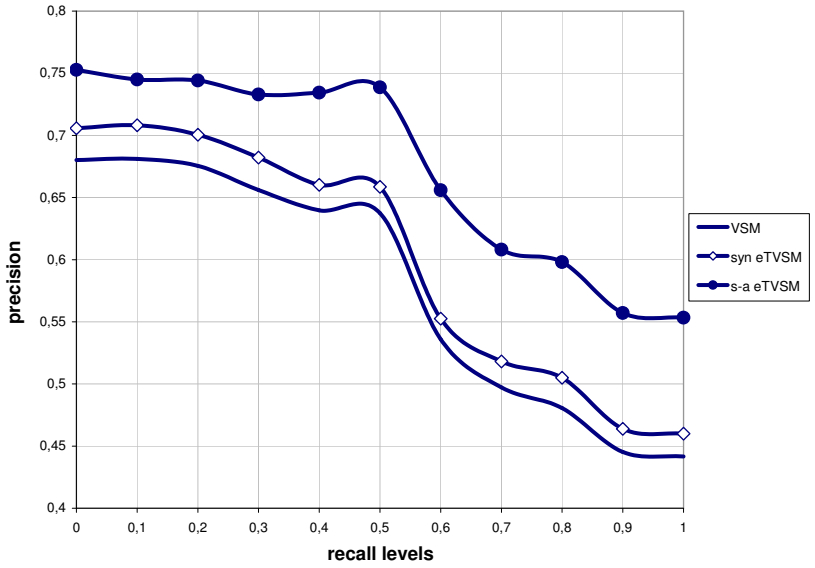

Figure 7.11: eTVSM with a semi-automated ontology precision at standard recall levels plot.

Comparisons

α	$t_{82,1-\frac{\alpha}{2}}$	I_α
0.00050	$3,6251$	$[-0.0059, 0.2567]$
0.00075	3.5021	$[-0.0014, 0.2522]$
0.00100	**3.4132**	[**0.0018, 0.2490**]
0.00125	3.3432	$[0.0043, 0.2465]$
0.00150	3.2853	$[0.0064, 0.2444]$

α—confidence level, t—Student-t distribution quantile, I—confidence interval.

Table 7.2: VSM and semi-automated eTVSM comparison—confidence intervals.

We will perform comparisons of semi-automated eTVSM with prior performed evaluations of VSM and synonymy eTVSM. First we will perform the comparison procedure described in section 5.6.2. We compare the semi-automated eTVSM with the VSM. Again, as in section 7.3.3, for a comparison measurement we will take recall and precision measured for each query at a cut-off rank equal to the number of relevant documents to this query proposed by the Time test collection. Thus, recall is equal to precision for each query and we have one measurement value for each query-model pair. We will construct confidence intervals for $\mu = \mu_1 - \mu_2$ where μ_1 is the mean measurement value for semi-automated eTVSM and μ_2 is the mean measurement value for VSM. Intervals were obtained based on computed estimators $\hat{z} = 0.1254$ and $S^2 = 0.1089$. Computational results for confidence intervals are given in Table 7.2.

The comparison setup concludes that semi-automated eTVSM performs better than VSM with respect to compared measurement parameter when confidence interval is a positive interval not containing zero. In Table 7.2 you can see that such condition holds when confidence level α reaches value of 0.001.

We have performed the same procedure for the same measurement parameter to compare semi-automated eTVSM with synonymy eTVSM. Confidence intervals for $\mu = \mu_1 - \mu_2$ were constructed, where μ_1 is the mean measurement value for semi-automated eTVSM and μ_2 is the mean measurement value for synonymy eTVSM. Intervals were obtained based on computed estimators $\hat{z} = 0.1077$ and $S^2 = 0.0836$. Computational results for confidence intervals are given in Table 7.3. In this comparison setup we can conclude that semi-automated eTVSM performs better than synonymy eTVSM with respect to compared measurement parameter when confidence interval is a positive interval not containing zero. In this comparison the confidence level of 0.001 still includes zero and has negative segment. However, at 0.00125 confidence level α the statement of semi-automated eTVSM superiority over synonymy eTVSM becomes statistically grounded.

α	$t_{82,1-\frac{\alpha}{2}}$	I_α
0.00050	$3,6251$	$[-0.0073, 0.2228]$
0.00075	3.5021	$[-0.0034, 0.2189]$
0.00100	3.4132	$[-0.0006, 0.2161]$
0.00125	**3.3432**	[**0.0016, 0.2138**]
0.00150	3.2853	$[0.0035, 0.2120]$

α—confidence level, t—Student-t distribution quantile, I—confidence interval.

Table 7.3: Synonymy eTVSM and semi-automated eTVSM comparison—confidence intervals.

In both comparisons performed the significance level α is low enough to conclude that semi-automated eTVSM outperforms VSM and synonymy eTVSM. Next, we perform a statistical test and check the level of difference in the measured parameter, that we can statistically reason on semi-automated eTVSM superiority. The test description was given in section 5.6.3. Results of the test are provided in Table set 7.4.

These results can be interpreted as follows: Suppose, you want to check the significance level of measured parameter for semi-automated eTVSM to be better as for VSM for $d_0 = 0.03$. The corresponding test statistics value T from Table 7.4.a is 1.6832. Further, you must find an appropriate t value in Table 7.4.c for which holds $T > t$. In our case this t value is 1.6542. The corresponding confidence level α is, therefore, 0.05. This means that semi-automated eTVSM outperforms VSM in compared parameter in 0.03 with a confidence level of 0.05. Following same procedure, we might conclude, that semi-automated eTVSM outperforms synonymy eTVSM in compared parameter in 0.03 with a confidence level α of 0.09.

The study of difference-confidence level pairs also speaks for superiority of semi-automated eTVSM over VSM and synonymy eTVSM, even in the test conditions of Time test collection that has only 83 queries. However, the problem of statistical significance justification of obtained results is opened for all test collections, as they usually possess limited number of queries, resulting in a small sets of tested measurement values.

a)

d_0	T
0.001	2.1948
0.005	2.1242
0.01	2.0360
0.02	1.8596
0.03	1.6832
0.04	1.5068
0.05	1.3304
0.06	1.1540
0.07	0.9776
0.08	0.8012
0.09	0.6247
0.10	0.4483

b)

d_0	T
0.001	1.8570
0.005	1.7874
0.01	1.7004
0.02	1.5264
0.03	1.3524
0.04	1.1784
0.05	1.0045
0.06	0.8305
0.07	0.6565
0.08	0.4825
0.09	0.3085
0.10	0.1345

c)

α	$t_{164,1-\alpha}$
0.01	2.3493
0.02	2.0702
0.03	1.8939
0.04	1.7616
0.05	1.6542
0.06	1.5629
0.07	1.4830
0.08	1.4115
0.09	1.3465
0.10	1.2867
0.15	1.0397
0.20	0.8438

d_0—tested difference level, T—test statistics,
α—confidence level, t—Student-t distribution quantile.

Table 7.4: Results of the test: a) test statistics for semi-automated eTVSM and VSM test, b) test statistics for semi-automated eTVSM and synonymy eTVSM test, c) confidence levels and corresponding Student-t distribution quantiles.

Summary

The proposed approach has delivered a significant improvement in eTVSM evaluation results. Nevertheless we have to rely on human assistance, we attempt to keep it as small as possible. Potential human operations are reduced to query interpretations. To further benefits of proposed approach one might address non local phenomenon of acquired improvements. Along with the re-evaluations of eTVSM for intermediate ontology configurations we have observed not only improvements in measurements for queries been processed, but also for other queries from test collection. For the concrete test collection (Time collection) it makes sense to go through all queries and perform proposed procedure. But what to do in a real-world scenario, with ever changing document collection and endless stream of various incoming queries? For such cases we propose the following behavior: The starting eTVSM ontology should be configured following the principles presented in synonymy eTVSM description (see section 7.3.2). Afterwards, one might apply ontology enrichment with term relations acquired from most common queries that were sent to the system. To make this approach feasible, one must keep track of queries sent to the system and their frequencies.

Additionally the proposed approach suits well for the Information Retrieval scenario with user feedback. Imagine that users of your system are allowed to send back their experience of working with your system in a form of documents they intend to see retrieved or retrieved with a higher rank position to a given query. One might enrich eTVSM ontology to contain term relations between query and document terms for such most frequent user suggestions.

7.5 Latent Semantic Analysis

So far we have performed evaluations and comparisons of the eTVSM and the VSM. On a very high level we explain superiority of eTVSM over VSM in terms of eTVSM

ability to actually "understand" documents and queries. eTVSM tries to model human perception in parts through representing human perceptional units and relations between them in the eTVSM ontology. Thus, an ontology configuration becomes some sort of a simplified human perceptual model we believe to be sufficient for our scope of Information Retrieval problem. In this section we want to discuss another Information Retrieval model that applies similar view to the Information Retrieval problem—the Latent Semantic Analysis (LSA). The LSA approach claims as one able to model human conceptual knowledge. Applied mechanisms allow comparing documents at the level of their topical similarity.

LSA is a well studied approach with widely available evaluation results for various test collections. We would like to perform comparisons of former obtained in our work evaluation results for eTVSM with same measurements obtained in LSA evaluations. Furthermore, we will give a brief introduction to LSA with presenting principles of LSA operations it takes to fulfill its tasks. We will then provide LSA evaluation results we have found in the literature for Time test collection. Finally, we would like to provide conceptual differences, as we see them, between eTVSM and LSA.

7.5.1 Introduction to Latent Semantic Analysis

LSA is a fully automatic mathematical/statistical technique for extracting and inferring relations of expected contextual usage of words in passages of discourse. It is not a traditional natural language processing or artificial intelligence program; it uses no humanly constructed dictionaries, knowledge bases, semantic networks, grammars, syntactic parsers, or morphologies, or the like, and takes as its input only raw text parsed into words defined as unique character strings and separated into meaningful passages or samples such as sentences or paragraphs. LSA is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations. LSA was applied and assessed in several ways. These are:

- predictor of query-document topic similarity judgments;
- simulation of agreed upon word-word relations and of human vocabulary test synonym judgments;
- simulation of human choices on subject-matter multiple choice tests;
- predictor of text coherence and resulting comprehension;
- simulation of word-word and passage-word relations found in lexical priming experiments;
- predictor of subjective ratings of text properties, i.e. grades assigned to essays;
- predictor of appropriate matches of instructional text to learners;
- mimic synonym, antonym, singular-plural and compound-component word relations, aspects of some classical word sorting studies, to simulate aspects of imputed human representation of single digits, and, in pilot studies, to replicate semantic categorical clustering of words found in certain neuropsychological deficits [37].

What is interesting in respect to our work is LSA application in Information Retrieval field. Some researchers have called attention to the analogy between Information Retrieval and human semantic memory processes. One way of expressing their commonality is to think of a searcher as having in mind a certain meaning, which he or she expresses in words, and the system as trying to find a text with the same meaning. System's success will then be the correct query meaning interpretation and further retrieval of documents with a similar meaning from a document collection. It was shown that LSA copes with this task better than systems that look for term matches in queries and documents. Its superiority can often be traced to its ability to correctly match queries to documents of similar topical meaning when query and document use different words.

Some first tests of LSA were performed against standard test collections, most of them were presented in Table 5.1 on page 39. These are test collections of documents for which representative queries have been obtained and knowledgeable humans have more or less exhaustively examined the whole database and judged which abstracts are and are not relevant to the topic described in each query statement. For such experiments LSA performance ranged from equivalent to the best prior methods to up 30% improvement as compared to them.

7.5.2 Latent Semantic Indexing

In the context of its application to Information Retrieval, LSA is called Latent Semantic Indexing (LSI). Prior to start comparison of eTVSM and LSI evaluation results we would like to present the algorithmic formalism used by LSI [38]. The first step is to represent the text as a matrix in which each row stands for a unique word and each column stands for a text passage or other context. Each cell contains the frequency with which the word of its row appears in the passage denoted by its column. The cell entries are then subjected to a preliminary transformation. It is basically the term weighting scheme in which each cell frequency is weighted by a function that expresses both the word's importance in the particular passage and the degree to which the word type carries information in the domain of discourse in general.

Let X be such a matrix that element (i, j) describes the occurrence of term i in document j, as simplest—term occurrence in the document. Then X looks like this:

$$\begin{array}{rc} & \begin{matrix} d_j \\ \downarrow \end{matrix} \\ t_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \end{array}$$

Each row in such a matrix will be a term vector giving its relation to each document:

$$t_i^T = \begin{bmatrix} x_{i,1} & \dots & x_{i,n} \end{bmatrix}$$

Similar, a column in this matrix will be a document vector, specifying its relation to each term:

$$d_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix}$$

Now, the dot product $t_i^T t_p$ between two term vectors gives the correlation between the terms over the documents. The matrix product XX^T contains all these dot products. Element (i, p) (which is equal to element (p, i)) contains the dot product $t_i^T t_p$ $(= t_p^T t_i)$. Likewise, the matrix $X^T X$ contains the dot products between all the document vectors, giving their correlation over the terms: $d_j^T d_q = d_q^T d_j$.

LSI applies singular value decomposition (SVD) to the matrix. This is a form of factor analysis, or more properly the mathematical generalization of which factor analysis is a special case. In SVD, a rectangular matrix is decomposed into the product of three other matrices. One component matrix describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values such that when the three components are matrix-multiplied, the original matrix is reconstructed. There is a mathematical proof that any matrix can be so decomposed perfectly, using no more factors than the smallest dimension of the original matrix. Now assume that our matrix X was subject to SVD to give U and V orthonormal matrices and Σ—a diagonal matrix.

$$X = U\Sigma V^T$$

The matrix products giving us the term and document correlations then become:

$$\begin{aligned} XX^T &= (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V^{T^T}\Sigma^T U^T) \\ &= U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T \\ X^T X &= (U\Sigma V^T)^T(U\Sigma V^T) = (V^{T^T}\Sigma^T U^T)(U\Sigma V^T) \\ &= V\Sigma U^T U\Sigma V^T = V\Sigma^T\Sigma V^T \end{aligned}$$

Since $\Sigma\Sigma^T$ and $\Sigma^T\Sigma$ are diagonal we see that U must contain the eigenvectors of XX^T, while V must be the eigenvectors of $X^T X$. Now the decomposition looks like this:

$$\begin{array}{ccccc} & U & & \Sigma & V^T \\ & & & & (\hat{d}_j) \\ & & & & \downarrow \\ (\hat{t}_i^T) \rightarrow & \begin{bmatrix} \begin{bmatrix} \\ u_1 \\ \\ \end{bmatrix} & \cdots & \begin{bmatrix} \\ u_l \\ \\ \end{bmatrix} \end{bmatrix} & \cdot \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_l \end{bmatrix} & \cdot \begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix} \end{array}$$

The values $\sigma_1, \ldots, \sigma_l$ are called the singular values, and $u_1, \ldots, u_l$ and $v_1, \ldots, v_l$ the left and right singular vectors.

When fewer than the necessary number of singular values are used, the reconstructed matrix is a least-squares best fit. One can reduce the dimensionality of the solution simply by deleting coefficients in the diagonal matrix, ordinarily starting with the smallest leaving k largest singular values. (In practice, for computational reasons, for very large test collections only a limited number of dimensions, currently a few thousand, can be constructed.) The amazing thing about this approximation is that not only does it have a minimal error, but it translates the term and document vectors into a concept space. The vector $\hat{t}_i$ then has k entries, each giving the occurrence of term i in one of the k concepts. Likewise, the vector $\hat{d}_j$ gives the relation between document j and each concept. This approximation can be written as:

$$X_k = U_k \Sigma_k V_k^T$$

You can now do the following:

- See how related documents j and q are in the concept space by comparing the vectors $\hat{d}_j$ and $\hat{d}_q$ (typically by cosine similarity). This gives you a clustering of the documents;
- Comparing terms i and p by comparing the vectors $\hat{t}_i$ and $\hat{t}_p$, giving you a clustering of the terms in the concept space;
- Given a query, view this as a mini document, and compare it to your documents in the concept space.

To do the latter, you must first translate your query into the concept space. It is then intuitive that you must use the same transformation that you use on your documents:

$$d_j = U_k \Sigma_k \hat{d}_j$$

$$\hat{d}_j = \Sigma_k^{-1} U_k^T d_j$$

This means that if you have a query vector q, you must do the translation $\hat{q} = \Sigma_k^{-1} U_k^T q$ before you compare it with the document vectors in the concept space. The comparison of the query vector with all document vectors will conclude the document collection search for relevant documents.

By reducing dimensionality only to the most significant dimensions LSI is able to abstract from unimportant seldom content extracting valuable conceptual knowledge and presenting it in the low-dimensional (appropriate for computational requirements) concept space. Such operation of abstracting from details and concentrating on main topic is concidered by LSI to be similar to human normal approach to world perception.

7.5.3 Comparison of LSI with eTVSM

In order to be able to compare LSI evaluation results with those already obtained for eTVSM we performed a search for existing LSI evaluations on the Time test collection. Further, we obtained an approximate values for comparable measurements on evaluation that has delivered best results for LSI out of what we have found. For this purpose we used evaluation results we obtained from [10]. The visual comparison of such approximate precision at standard recall levels plot for LSI with former eTVSM evaluations can be done in Figure 7.12. Here, the LSI effectiveness approximation, dashed line (*LSI approx*), is superposed on already presented evaluation measurements for eTVSM.

Visual analysis shows that LSI evaluation from [10] performs better than synonymy eTVSM 7.3 (*syn eTVSM*) and worse than semi-automated eTVSM (*s-a eTVSM*) evaluation measurements obtained in section 7.4. However, you might as well find LSI evaluations that do not show superiority over synonymy eTVSM [18]. LSI gives descriptions on how to represent the contextual-usage meaning of terms by precise statistical computations leaving minor space for variations. The only model parameter that can influence effectiveness is the number of dimensions to include in a concept space. In such conditions, the number of possible variations of concept space modeling is

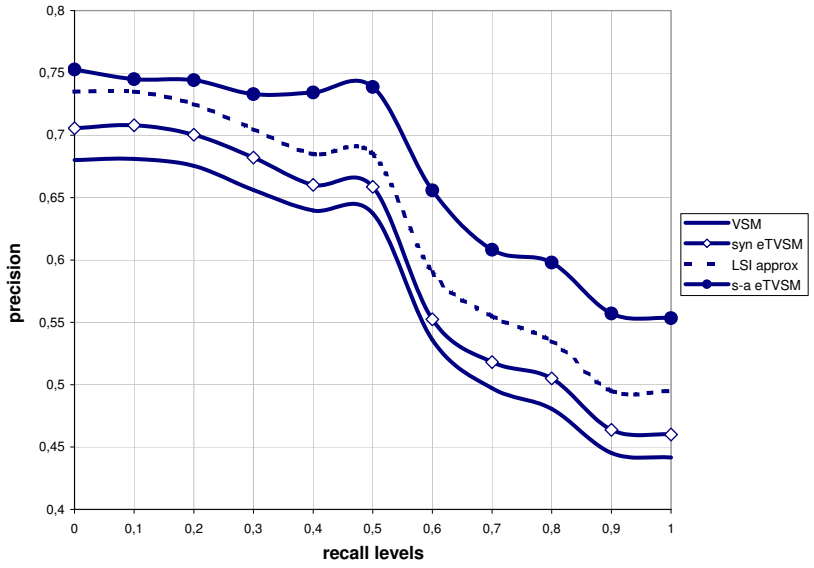

Figure 7.12: LSI precision at standard recall levels plot.

very limited. On the other hand, eTVSM allows an unlimited number of operational spaces each giving the specification of capable human-like understanding of semantic relations between terms. Some general LSI drawbacks are:

- The resulting dimensions might be difficult to interpret. This leads to results which can be justified on the mathematical level, but have no interpretable meaning in natural language;
- LSA, in general, assumes that words and documents form a joint Gaussian model (a Poisson distribution is observed). A newer alternative is a probabilistic Latent Semantic Analysis [29] based on a multinomial model. It is reported to give better results than standard LSA.

Chapter 8

Conclusions

In this work we have performed a complete evaluation cycle of a novel Information Retrieval model—the eTVSM. It includes model configuration, actual experiments and further collecting and aggregating of quality measurements. Prior to our evaluations we have presented theoretical background on the topic which included eTVSM description, Information Retrieval model effectiveness measurements, statistical approach to model comparisons and evaluation design for performing Information Retrieval model evaluations.

We have performed evaluations in chapter 7 which show that eTVSM performs better then other approaches like VSM or LSI if a proper domain ontology is provided. The eTVSM can represent semantic term relations to model common linguistic phenomena by representing these relations in a domain ontology. Such semantic term relations are then mapped by the eTVSM onto the vector angles in the operational vector space. Furthermore, the eTVSM proposes a formal procedure for deriving term angles from an eTVSM ontology model which now becomes a central place for semantic term relations modeling. As long as semantic term relations are considered to be expressible through one parameter a vector space is a perfect choice for representing them through angles. A vector space gives an endless space for expressing levels of semantic relations, yet having an intuitive interpretation of terms to be more relevant the smaller the angle is between corresponding term vectors.

One of the drawbacks of the eTVSM is its computational complexity. If to come up with its pure implementation that follows all the theoretical prescriptions presented in chapter 3, the system that is based on such an Information Retrieval model will be hardly usable. One change in ontology would result in the necessity to reconstruct all document models. Changes in ontology might also result in non-local change in interpretation similarities that are used while obtaining eTVSM document similarities. In section 6.5 we have presented approach to implementing Information Retrieval system based on eTVSM which reduces overall complexity by not requiring synchronous execution of certain activities. It includes simplification assumptions and basically defines a transaction schema for common eTVSM operations. This approach was used for eTVSM implementation in Themis (developed framework for Information Retrieval models) and has proven its benefits while eTVSM evaluations conducted in chapter 7. In general, we believe that with proposed transaction schema eTVSM might be deployed for a real-world scenario as, e.g., a model for the Information Retrieval system in a middle size company's Intranet.

Any Information Retrieval system along with efficient algorithms also requires

effectiveness from incorporated retrieval model. Search results should not only be delivered fast, they also need to be relevant to user queries. As the output of performed evaluations we have came up with an eTVSM ontology lifecycle that starts together with system deployment. It was presented in section 7.4. The basic idea is to start from an automatically constructed ontology which has delivered the best effectiveness of the model—synonymy eTVSM (see section 7.3.2). Then, this basic ontology can get enriched online based on the developed heuristic guidelines presented in section 7.4.2. These heuristic steps should improve chances for users to find relevant information.

A further positive aspect of the eTVSM is that it allows on the fly model reconfigurations through changes in eTVSM ontology. This feature is important for Information Retrieval systems that intend to consider user feedback to improve effectiveness. It is a case when users are allowed to send back their experience of working with a system in a form of documents they want to be retrieved with a higher rank position to a given query.

Bibliography

[1] *National Institute of Informatics (NTCIR Project)*, 2006.
http://research.nii.ac.jp/ntcir/permission/perm-en.html

[2] *Reuters-21578 test collection*, 2006.
http://www.daviddlewis.com/resources/testcollections/reuters21578/

[3] *Text Retrieval Conference (TREC)*, 2006.
http://trec.nist.gov

[4] *WordNet download*, 2006.
http://wordnet.princeton.edu/obtain

[5] *WordNet SQL Builder project*, 2006.
http://wnsqlbuilder.sourceforge.net/

[6] *Reuters Corpora*, 2007.
http://groups.yahoo.com/group/ReutersCorpora/

[7] *WordNet online*, 2007.
http://wordnet.princeton.edu/perl/webwn

[8] BAEZA-YATES, R. A.; RIBEIRO-NETO, B. A.: *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
http://citeseer.ist.psu.edu/baeza-yates99modern.html

[9] BAI, J.; SONG, D.; BRUZA, P.; NIE, J.-Y.; CAO, G.: *Query Expansion Using Term Relationships in Language Models for Information Retrieval*, 2005

[10] BAST, H.; MAJUMDAR, D.: *Why Spectral Retrieval Works*. In *SIGIR Forum*. 2005, p. 11–18

[11] BECKER, J.; KUROPKA, D.: *Topic-based Vector Space Model*. In *Proceedings of the 6th International Conference on Business Information Systems*. Colorado Springs, July 2003, p. 7–12.
http://bpt.hpi.uni-potsdam.de/twiki/bin/view/Public/DominikKuropka

[12] BOOKSTEIN, A.; KLEIN, S. T.; RAITA, T.: *Detecting Content Bearing Words by Serial Clustering*. In *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*. 1995, p. 319–327

[13] BUCKLEY, C.; ALLAN, J.; SALTON, G.: *Automatic Retrieval with Locality Information using SMART*. In *Proceedings of the First Text Retrieval Conference TREC-1*. 1993, p. 59–72

[14] BUNT, H.: *Mass Terms and Model-theoretic Semantics*. Cambridge University Press, New York, 1985

[15] BURKHARDT, H.; DUFOUR, C.: *Handbook of Metaphysics and Ontology*. Philosophia Verlag, München, 1991

[16] CASSANDRAS, C. G.: *Discrete Event Systems — Modeling and Performance Analysis*. Aksen Associates, Boston, 1993

[17] CHEN, P. P.: *The Entity-Relationship Model—Toward a Unified View of Data.*. In *ACM Trans. Database Syst.* 1(1), 1976: p. 9–36.
http://csc.lsu.edu/news/erd.pdf

[18] CHENG, B.: *Towards Understanding Latent Semantic Indexing*

[19] CHURCH, K. W.: *One Term Or Two?*. In *SIGIR*. 1995, p. 310–318

[20] CRESTANI, F.: *A Model for Combining Semantic and Phonetic Term Similarity for Spoken Document and Spoken Query Retrieval*. Fachbericht TR-99-020, Berkeley, CA, 1999

[21] DIRVEN, R.: *Conversion as a Conceptual Metonymy of Basic Event Schemata*. In *Workshop on Metonymy*. Hamburg University, Germany, June 23-24, 1996

[22] DUPRET, G.: *Latent Semantic Indexing with a Variable Number of Orthogonal Factors*

[23] FASS, D.: *Processing Metonymy and Metaphor*. Greenwich, Conn.: Ablex Pub. Corp., London, 1997

[24] FLORIDI, L.: *The Blackwell Guide to the Philosophy of Computing and Information*. Oxford University Press, New York, 2003

[25] FROMKIN, V.; RODMAN, R.; HYAMS, N.: *An Introduction to Language*. Heinle, 2006

[26] GOSSET, W. S.: *The probable error of a mean*. In *Biometrika* (6(1)), 1908: p. 1–25

[27] GROSS, J. L.; YELLEN, J.: *Handbook of Graph Theory*. CRC Press, 2003

[28] HEARST, M.: *Automatic Acquistion of Hyponyms from Large Text Corpora*. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*. Nantes, France, 1995

[29] HOFMANN, T.: *Probabilistic Latent Semantic Analysis*. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*. Stockholm, 1999

[30] HULL, D.: *Stemming Algorithms — A Case Study for Detailed Evaluation*. In *JASIS* (47(1)), 1996: p. 70–84

[31] IMAI, H.; COLLIER, N.; TSUJII, J.: *A Combined Query Expansion Approach for Information Retrieval*, 1999

[32] JONES, K.: *Information Retrieval Experiment*. Butterworth, 1981

[33] JONES, K.: *Synonymy and Semantic Classification*. Edinburgh University Press, Edinburgh, Scotland, 1986

[34] KINI, A. U.: *On thr Effect of Inquirt Term-weighting Scheme on Query-sensitive Similarity Measures*, 2005

[35] KROVETZ, R.; CROFT, W. B.: *Lexical Ambiguity and Information Retrieval*. In *Information Systems* 10(2), 1992: p. 115–141

[36] KUROPKA, D.: *Modelle zur Repräsentation natürlichsprachlicher Dokumente*. Logos Verlag, Berlin, 2003

[37] LAHAM, D.: *Latent Semantic Analysis Approaches to Categorization*. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society* 1997: p. 979

[38] LANDAUER, T. K.; FOLTZ, P. W.; LAHAM, D.: *Introduction to Latent Semantic Analysis*. In *Discourse Processes* 25, 1998: p. 259–284

[39] LEE, D. L.; CHUANG, H.; SEAMONS, K.: *Document Ranking and the Vector-Space Model*. In *IEEE Softw.* 14(2), 1997: p. 67–75

[40] LOVINS, J. B.: *Development of a Stemming Algorithm*. In *Mechanical Translation and Computational Linguistics* (11), 1968: p. 22–31

[41] O'MAHONY, M.: *Sensory Evaluation of Food: Statistical Methods and Procedures*. CRC Press, New York, 1986

[42] PAICE, C.: *Method for Evaluation of Stemming Algorithms Based on Error Counting*. In *JASIS* (47(8)), 1996: p. 632–649

[43] PARAPAR, D.; BARREIRO, A.; LOSADA, D. E.: *Query Expansion Using WordNet with a Logical Model of Information Retrieval*

[44] PECK, R.; OLSEN, C.; DEVORE, J. L.: *Introduction to Statistics and Data Analysis (with ThomsonNOW Printed Access Card)*. Duxbury Press, 2007

[45] POLYVYANYY, A.: *Evaluation Design of Information Retrieval System with eTVSM Specific Extensions*, 2006.
`http://bpt.hpi.uni-potsdam.de/twiki/pub/Public/SeminarPublications/ArtemPolyvyanyy.pdf`

[46] POLYVYANYY, A.: *Evaluation of a Novel Information Retrieval Model: eTVSM*, 2007. Master thesis at the Hasso Plattner Institute, University of Potsdam,
`http://kuropka.net/3rd_party_files/Thesis_Artem_Polyvyanyy.pdf`

[47] PORTER, M.: *An Algorithm for Suffix Stripping*. In *Program* (14(3)), 1980: p. 130–137

[48] PORTER, M.: *Snowball: A Language for Stemming Algorithms*, 2001.
`http://www.snowball.tartarus.org/texts/introduction.html`

[49] RAGHAVAN, V. V.; WONG, S. K. M.: *A Critical Analysis of Vector Space Model for Information Retrieval*. In *Journal of the American Society for Information Science* (35(5)), 1986: p. 279–287

[50] ROSSO, P.; FERRETTI, E.; JIMNÉZ, D.; VIDAL, V.: *Text Categorization and Information Retrieval Using Wordnet Senses*, 2004.
http://citeseer.ist.psu.edu/697740.html

[51] SALTON, G.: *Introduction to Modern Information Retrieval (McGraw-Hill Computer Science Series)*. McGraw-Hill Companies, September 1983

[52] SALTON, G.: *Automatic Text Processing — The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989

[53] SALTON, G.; BUCKLEY, C.: *Term Weighting Approaches in Automatic Text Retrieval*. Fachbericht, Ithaca, NY, USA, 1987.
http://portal.acm.org/citation.cfm?id=866292

[54] SALTON, G.; WONG, A.; YANG, C. S.: *A Vector Space Model for Automatic Indexing*. In *Communications of the ACM* (vol. 18, nr. 11), 1975: p. 613–620

[55] SANDERSON, M.: *Reuters Test Collection*. In *BSC IRSG*. 1994

[56] SANDERSON, M.: *Word Sense Disambiguation and Information Retrieval*. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*. Dublin, IE, 1994, p. 49–57

[57] S.N.BERNSTEIN: *The Scientific Legacy of P. L. Chebyshev. First Part: Mathematics*. Academiya Nauk SSSR, Moscow-Leningrad, 1945

[58] SRZEDNICKI, J. T. J.; RICKEY, V. F.: *Lesniewski's Systems: Ontology and Mereology*. Kluwer, 1984

[59] STUMP, G.: *Inflectional Morphology: A Theory of Paradigm Structure (Cambridge Studies in Linguistics)*. Cambridge University Press, New York, 2001

[60] TURTLE, H. R.; CROFT, W. B.: *A comparison of text retrieval models*. In *Comput. J.* 35(3), 1992: p. 279–290

[61] VALIN, V.; ROBERT, D.: *An Introduction to Syntax*. Cambridge University Press, New York, 2001

[62] VAN RIJSBERGEN, C. J.: *Information retrieval*. Butterworths, London, 1979

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
18	978-939-469-58-2	**Proceedings of the Fall 2006 Workshop of the HPI Research School on Service-Oriented Systems Engineering**	Benjamin Hagedorn, Michael Schöbel, Matthias Uflacker, Flavius Copaciu, Nikola Milanovic
17	3-939469-52-1 / 978-939469-52-0	**Visualizing Movement Dynamics in Virtual Urban Environments**	Marc Nienhaus, Bruce Gooch, Jürgen Döllner
16	3-939469-35-1 / 978-3-939469-35-3	**Fundamentals of Service-Oriented Engineering**	Andreas Polze, Stefan Hüttenrauch, Uwe Kylau, Martin Grund, Tobias Queck, Anna Ploskonos, Torben Schreiter, Martin Breest, Sören Haubrock, Paul Bouché
15	3-939469-34-3 / 978-3-939469-34-6	**Concepts and Technology of SAP Web Application Server and Service Oriented Architecture Products**	Bernhard Gröne, Peter Tabeling, Konrad Hübner
14	3-939469-23-8 / 978-3-939469-23-0	**Aspektorientierte Programmierung – Überblick über Techniken und Werkzeuge**	Janin Jeske, Bastian Brehmer, Falko Menge, Stefan Hüttenrauch, Christian Adam, Benjamin Schüler, Wolfgang Schult, Andreas Rasche, Andreas Polze
13	3-939469-13-0 / 978-3-939469-13-1	**A Virtual Machine Architecture for Creating IT-Security Labs**	Ji Hu, Dirk Cordel, Christoph Meinel
12	3-937786-89-9 / 978-3-937786-89-6	**An e-Librarian Service - Natural Language Interface for an Efficient Semantic Search within Multimedia Resources**	Serge Linckels, Christoph Meinel
11	3-937786-81-3	**Requirements for Service Composition**	Prof. Dr. M. Weske, Dominik Kuropka Harald Meyer
10	3-937786-78-3	**Survey on Service Composition**	Prof. Dr. M. Weske, Dominik Kuropka Harald Meyer
9	3-937786-73-2	**Sichere Ausführung nicht vertrauenswürdiger Programme**	Andreas Polze Johannes Nicolai, Andreas Rasche
8	3-937786-72-4	**Resourcenpartitionierung für Grid-Systeme**	Andreas Polze, Matthias Lendholt, Peter Tröger
7	3-937786-56-2	**Visualizing Design and Spatial Assembly of Interactive CSG**	Prof. Dr. Jürgen Döllner, Florian Kirsch, Marc Nienhaus
6	3-937786-54-6	**Konzepte der Softwarevisualisierung für komplexe, objektorientierte Softwaresysteme**	Prof. Dr. Jürgen Döllner, Johannes Bohnet

ISBN 978-3-939469-95-7
ISSN 1613-5652