Exploring multi-word similarity measures for Information Retrieval applications: the T-SRM method

Euthymios Drymonas

Technical University of Crete (TUC)
Department of Electronics and Computer Engineering

Chania 2006

Table of Contents

Abstract

Term extraction relates to identifying the most characteristic or important terms in a corpus. In this work we focus on the problem of extracting multi-word terms from documents. We implemented a state-of-the-art method for term extraction which combines linguistic and statistical information [15]. Methods for computing the similarity between such multi-word terms are also considered. The approach followed combines internal (lexical) and external (contextual) criteria [21]. Lexical similarity is based on computing common constituents (i.e. term heads and modifiers within multi-word terms) while contextual similarity is based on the likelihood of using the same terms in similar concepts along with the terms. We combine these measures to form a hybrid similarity measure for computing the similarity between multi-word terms. The effectiveness of this hybrid measure was evaluated using human relevance judgements. The results of the evaluation revealed that the proposed similarity measure approximates the human notion of similarity up to 73%. Building upon multi-word similarity, we proposed an information retrieval model, the *Term-based Similarity Retrieval Model (T-SRM)*, capable of discovering similarities between documents that contain conceptually similar terms. The *T-SRM* was applied for retrieval on the OhsuMed document collection. The results demonstrated very promising performance improvements over the Vector Space Model, the classic model for information retrieval.

Acknowledgements

I would like to thank Dr. Petrakis for the advice, encouragement and support he provided to me in supervising this thesis. I would like to thank all people in the Intelligent Systems Laboratory of Technical University of Crete, especially Epimenidis Voutsakis and Kelly Zervanou for their technical advise and their invaluable contribution in this thesis.

Chapter 1. Introduction

The increasing amounts of text information in modern application domains such as medicine, digital libraries and the Web, brings new challenges to information management. Information extraction in particular, plays an important role towards understanding better the contents of document collections and can be used for improving the accuracy of processes such as document indexing and retrieval. As a result of this, information extraction has become of primary interest in computational and applied linguistics, language interpretation, as well as in various other disciplines for extracting terminologies from texts. By terminologies we refer to the study of and the field of activity concerned with the collection, description, processing and presentations of terms [17] (i.e. lexical items belonging to specialised areas of usage). Such a task used to be particularly laborious in the past, as term extraction was carried manually by human experts. The introduction of computers to language analysis facilitated the automation of the term extraction process. As a result, a number of term extraction methods have been developed to assist terminological work.

Because terms can also be related to existing knowledge and to each other, the notion of *term similarity* has also been defined and considered in different ways: terms may have functional, structural, lexical or other similarities. Establishing relations between extracted terms from a corpus is indispensable for improving information extraction, document categorisation and information retrieval [51].

The purpose of *information retrieval* is to assist users in locating the information they are looking for. Information retrieval is currently being applied in a variety of application domains from database systems to web information search engines. The main idea is to locate documents that contain terms that users specify in queries.

Retrieval, by classical information retrieval models (e.g. Vector Space, Probabilistic, Boolean), is based on plain lexicographic term matching between terms (e.g. a query and a document term are considered similar if they are lexicographically the same). However, plain lexicographic analysis and matching is not generally sufficient to determine if two terms are similar and consequently whether two documents. Two terms can be lexicographically different although they have the same meaning (e.g. they are synonyms). The lack of common terms in two documents does not necessarily mean that the documents are irrelevant. Similarly, relevant documents may contain conceptually similar but not necessarily the same terms. Conceptually similar terms may be expressed in different words in the documents and the queries, and direct comparison between them is not effective (e.g. the Vector Space Model will not recognize synonyms or semantically similar terms).

In this work we apply Automatic Term Extraction in combination with term similarity measures for the purpose of improving the effectiveness if information retrieval. We examined term extraction techniques, we experimented with similarity measures between multi-word terms and we explored machine learning algorithms computing the relative importance of various term similarity criteria (i.e. lexical and contextual) for defining a hybrid similarity measure, combining different similarity measures. Building upon term similarity, we proposed the *Term Similarity Retrieval Model (T-SRM),* which is capable of computing the similarity between documents containing similar multi-word terms (not just single word terms as in classical retrieval models).

Our work is organised as follows: Chapter 2 discusses the background and related work. Chapter 3 talks about the methodology followed while chapter 4 discusses the Implementation part. chapter 5 talks about the Experiments and Evaluation and finally chapter 6 denotes the future work.

Chapter 2. Related Work & Background

This work aims of exploring similarity measures between multi-word terms and their application in Information Retrieval. In this chapter we discuss problems and approaches in the two related research areas. That is Automatic Term Extraction (ATE) and Information Retrieval (IR). Firstly, we identify the main problems in ATE (i.e. variation and ambiguity). We provide a definition for *term* and we introduce the notion of *multi-word term*. We also present various automatic term extraction approaches (statistical, linguistic, machine learning, hybrid) and we discuss multi-word term similarity measures. Then we discuss the main objectives of IR along with classical IR approaches.

2.1 Automatic Term Extraction (ATE)

The terms represent important information related to a corpus, as they linguistically represent the concepts in documents, express the semantic content of texts and characterize the documents semantically. Term Extraction is considered to be a difficult task and it is usually carried out by human experts. However this process tends to be slow and subjective and does not scale-up well for large document collections.

2.1.1 *What is a term ?*

According to ISO704 [45], terms are words or multi-word expressions, which, contrary to general language words, are deliberately created within a scientific or technical linguistic community not only for concept naming, but also for specialized concept distinction and classification purposes. The automatic identification of terms is of particular importance in information management applications since certain linguistic expressions are bound to convey the informational content load of a

document. In the specific context of term extraction for retrieval purposes, the principal objective of ATE is the identification of discrete content indicators , namely *index terms*. At this point, a distinction should be made between the notion of *domain* term and the *index* term. *Domain terms* are deliberately created within a scientific or technical linguistic community for specialized concept distinction and classification purposes, as defined by ISO704 [45]. *Index terms* are key concepts, words of phrases, which semantically label and categorize the content of a document for information management purposes, such as retrieval. For this reason, although terms (domain terms) may be discovered in an indexing process, neither all domain terms are useful index terms, nor all index terms are domain terms. For example, a valid domain term appearing very frequently in a document collection is useless for the retrieval of a specific document.

In the rest of this work, the notion of term refers mainly to index terms, though in the ATE approach used in our method (C-NC Value, as we will see in the next chapters), the design objective is domain term extraction, rather than indexing. Our purpose is to use the information from this Automatic Term Extraction method for the establishment of term similarities between multi word terms and the application of all the above in our proposed Information Retrieval Model, the *Term-based Similarity Retrieval Model (T-SRM)*.

2.1.2 *Problems in term identification*

In theory, terms must be mono-referential (i.e. one and only one term should refer to only one). In practice, we face phenomena of semantic ambiguity, such as Polysemy (i.e. when a term refers to many different concepts) and synonymy (i.e. when many terms reflect the same concept).

Statistical measures (e.g. frequency of occurrence) are widely used for the identification and evaluation of the terms in document collections. However, the frequency does not always constitute a sufficient criterion for term identification, given that many words may appear often within specialized texts without being terms and conversely many domain terms may have low frequencies.

2.1.3 *Multi-Word terms : Definition – Problems*

The majority of terms consist of compound units [44]. Most compound terms are nouns, which consist of either two or more nouns, e.g. "*carrier protein*" , or by a noun that follows an adjective, ex. "*nuclear receptor*". Also the existence of nested terms is very common .e.g. [cornea [nerve] cell]. The term "*cornea cell*" is nested within the "*cornea nerve cell*".

The identification of multi-word terms, although it may be fairly an easy task for an expert, it is not equally simple for an automatic term extraction system. The identification of compound and nested terms constitutes one of the main researching challenges in ATE, as it relates to the resolution of the problems of *ambiguity* (the same term corresponding to many concepts) and *variation* (many terms leading to the same concept).

Ambiguity is a general phenomenon of natural languages that greatly has preoccupied the research area of linguistic technology. In the field of automatic term extraction, the phenomenon of ambiguity may be of four types : *morphosyntactic, syntactic, semantic and classification ambiguity*. For example, we have morphosyntactic ambiguity when a word may be interpreted in more than one ways (i.e. as a verb or noun (e.g. "walk"), as an adjective or adverb). Further explanation regarding other types of term ambiguity is given in [44].

Variation appears when a concept is expressed with many synonym terms or with variants of the same term (table 2.1). Variation is widely common phenomenon in terminology, it is estimated that the 37% of terms that appear in a document consist of variations [46]. Variant tracking and identification is of a great importance to term extraction and the construction of thesauri. It is important to know if different term patterns are referring to the same concept or (in general) if they are related to each other (and how).

haemorrhage	v	blood loss	*Lexico-semantic*
Clones of human	v	human clones	*Syntactic*
Down syndrome	v	*Down's* syndrome	*Morphological*
amyloid *beta*-protein	v	amyloid *β*-protein	*Orthographical*

Table 2.1 Variant example types

The FASTR system [46] handles morphological and syntactic variations, while semantic variants are handled via a specialized lexicon (e.g. "WordNet").
In this work, we applied a simple morphological analyser from WordNet Java library, for handling morphological and orthographic variants in our termextraction process.

2.1.4 Automatic Term Extraction Approaches – Term Similarity

An overview of the methods for automatic term extraction and for computing similarities between terms is presented below. Some make use of linguistic knowledge, some others of statistical information while the majority of methods make use of both.

A purely linguistically based tool to term extraction, created by the French Electricity Board for thesauri updating and creation, is *Lexter*. *Lexter* works in two stages. During the first stage, it extracts all word sequences that on linguistic rules could *not* constitute a terminological unit. Studies of the linguistic properties of terms have shown that certain word sequences rarely constitute a term, for example sequences comprising conjugated verbs, pronouns, conjunctions or certain strings of prepositions + determiners. Next, based again on linguistic information on the prevailing term formation patterns of the special language under study, *Lexter* extracts subsets from maximal length noun phrases that most likely constitute a terminological unit. The resulting list is then submitted to an expert for validation. Another approach which makes use both of word repetition and negative knowledge is proposed by Oueslati et al. (1996). Their algorithm extracts repeated word sequences and then uses a stop list and a domain verb list to filter the extracted data. The result is a list of noun-noun combinations. Additionally, *FASTR*, an algorithm proposed by Jacquemin [46], attempts to retrieve both terms and term variants in an attempt to improve recall (i.e. to reveal even more terms).
A state-of-the art method for extracting multi-word terms is *KEA* [14]. *KEA* automatically extracts keyphrases from the full text of documents. The set of all candidate phrases in a document are identified using rudimentary lexical processing, features are computed for each candidate, and machine learning is used to generate a

classifier that determines which candidates should be assigned as keyphrases. Two features are used in the standard algorithm: tf*idf and position of first occurrence. The tf*idf requires a corpus of text from which document frequencies can be calculated; the machine learning phase requires a set of training documents with keyphrases assigned.

Another method for the automatic extraction of multi-word terms is *C-NC/Value* [15]. *C-NC/Value* constitutes a domain-independent method, combining linguistic and statistical information. It enhances the common statistical measure of frequency of occurrence and incorporates information from context words to the extraction of terms.

Based on the work by Milios et al. demonstrating that C/NC-Value outperforms KEA achieving significantly better precision and recall in identifying terms in special text corpora [13], we use the C-NC Value method for the extraction phase, as we will discuss in methodology chapter (chapter *3*).

2.1.5 Term Similarity

Several methods have been developed for the identification of conceptual similarities among terms. For example, Bourigault and Jacquemin [22] used lexical similarities to cluster terms. Their idea is based on adapting the term normalization process proposed within the FASTR framework [46]. A cluster is produced by linking terms that are associated by specific syntactic variation links (namely lexical characteristics and possible term-formation decompositions), which reflect the internal term structures.

Statistical methods (such as co-occurrence frequency counts) have also been used for establishing term similarities. For example, Maynard and Ananiadou [6] and Mima, Ananiadou and Nenadic [47] analysed terms co-occurring in a close proximity to one another as a basis for estimating term similarities. However, term co-occurrences and statistical distributions over larger text units (e.g. documents) may not reveal significant associations for some types of relationships (Hindle [48] Ding, Berleant, Nettleton and Wurtele [49]). Therefore, statistical and shallow-parsing methods have been combined. For example, Hindle [48] suggested a similarity measure among nouns based on mutual information of subject-verb and verb-object co-occurrences.

His main assumption is that a noun appears as subject or object of a restricted set of verbs, and that, consequently, each noun can be characterized by the verbs it co-occurs with. Grefenstette [19] extended this approach by considering other grammatical roles.

2.2 Information retrieval (IR)

Information Retrieval deals with the representation, storage, organization of, and access to information items [4]. The user must express his information need into a *query* which can be processed by the IR system (the search engine). Given a user query, the objective of an IR system is to retrieve information which is related to the query and might be useful to the user. In its most common form , the translation of the user query yields a set of keywords (*index terms*) which summarize the description of the user information need [4]. An index term in IR is simply a (document) word whose semantics helps in representing the document's main themes. For example, let us consider a collection of a thousand documents: a word which appears in all documents is completely useless as an index term, because it does not distinctly represent the particular content of a specific document in the collection. Conversely, a word which only appears in some, but not all documents is quite useful, because it narrows down considerably the space of documents which might be of interest to the user. Thus, it should be clear that distinct index terms have varying relevance when used to describe documents. The assignment of numerical *weights* to each index term of a document allows for the estimation of the varying relevance of the index terms.

2.2.1 ***Building an information retrieval system***

Before the retrieval process can be initiated, it is necessary to define the text database. Text normalization operations are firstly applied to the text[4]. Such operations include *stemming* (the reduction of distinct words into their common grammatical roots) and the elimination of *stopwords* (such as articles and connectives). Text operations reduce the complexity of the document representation and result in a set of *index terms* for each document.

Different index structures might be used to speed up the search, but the most popular

one is the *inverted file*. It can be thought as a data structure that allows fast random access to words stored inside it contained in documents. The concept behind it, is analogous to an index at the end of a book, which lets the reader locate rapidly pages that discuss certain topics.

The user first specifies a query (an expression of his information need) which is then parsed and transformed by the same text operations into a vector of terms. Then the query is processed to obtain the retrieved documents. The retrieved documents are ranked according to their likelihood of relevance to the query. The user then examines the set of ranked documents in the search for useful information.

2.2.2 *Classic Information Retrieval*

In this section we briefly present the three classic models namely, the Boolean, the Vector, and the Probabilistic models.

Boolean Model

The Boolean model is a simple retrieval model based on set theory and Boolean algebra. It considers that index terms are present or absent in a document. As a result, the index term weights are assumed to be all binary, i.e. $w_{i,j} \in \{0,1\}$. A query q is composed of index terms linked by boolean operators such as"*not*", "*and*", " *or*".

The Boolean model suffers from the following drawbacks. First, its retrieval strategy is based on a binary decision criterion (i.e. a document is predicted to be either relevant or non-relevant) without any notion of grading scale. Second, while Boolean expressions have precise semantics, frequently it is not simple to translate an information need into a Boolean expression. Finally because the Boolean model predicts that each document is either *relevant* or *non-relevant*, there is no notion of a partial match to query conditions.

To conclude, the main advantages of the Boolean model are the clean formalism behind the model and its simplicity. The main disadvantages are that exact matching may lead to retrieval of too few or too many documents and that retrieved documents are not ranked by relevance to the query.

Vector Space Model

The vector model [41, 42] recognizes that the use of binary weights is too limiting and proposes a framework in which partial matching is possible and desirable. This is accomplished by assigning *non-binary* weights to index terms in queries and in documents. These term weights are ultimately used to compute the *degree of similarity* between each document stored in the system and the user query. The vector space model sorts the retrieved documents by decreasing order of similarity. Also, retrieves documents that match the query only partially (i.e. a query term may be absent from a document). Similarly, the vector for a document d_j is represented by $d_j = (w_{1;j}, w_{2;j}, ..., w_{t;j})$. A document d_j and a user query q are represented as t-dimensional vectors (figure 2.1). Let $w_{i,q}$ be the weight assisted with the pair $[k_i,q]$, where $w_{i,q} >= 0$. The query vector q , is defined as $q = (w_{1;q}, w_{2;q}, ..., w_{t;q})$ where t is the total number of index terms in the system.

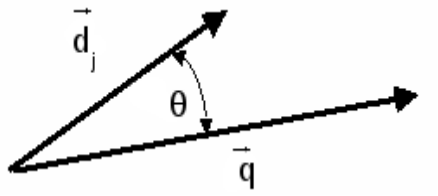


Figure 2.1 The cosine of θ is adopted as $sim(d_j, q)$

The Vector Space Model computes the degree of similarity of the document d_j with regard to the query q as the cosine of the angle between document vectors $\vec{q}$ and $\vec{d}$. This is computed as follows, where $\vec{d}_j \cdot \vec{q}$ is the inner product of the document and query vectors.

$$Sim(d_{i,q}) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_t w_{i,j} \times w_{i,q}}{\sqrt{\sum_t w_{i,j}^2} \times \sqrt{\sum_t w_{i,q}^2}}$$

$Sim(d_j, q)$ varies from 0 to +1. Thus, instead of attempting to predict whether a document is relevant or not, the vector model ranks the documents according to their degree of similarity to the query.

Appropriate weight to documents and queries is computed according to the tf*idf formula. The *tf* (i.e. term frequency) measures the raw frequency of a term k_i inside a document d_j. *Term frequency* is referred to as the *tf factor* and provides a measure of how well a term describes a document content. The inverse of the frequency of a term k_i among the documents in the collection, measures the importance of a term in the whole document collection. This factor is referred to as the *inverse document frequency* or the *idf factor*. The motivation for usage of an idf factor is that terms which appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one.

The *tf*idf weighting* scheme tries to balance these factorsand computes weights as :

$$w_{i,j} = tf * idf \qquad (2.1)$$

The main *advantages* of Vector Space Model (VSM) are (according to [4]) :

i. Its term-weighting scheme improves retrieval performance
ii. Its partial matching strategy allows retrieval of document that approximate the *q* conditions (don't necessarily match all query terms)
iii. Its cosine ranking formula allows for sorting the documents by decreasing similarity with the *q*.

The vector model has the *disadvantage* that index terms are assumed to be mutually independent. (equation 2.1 does not account for index term dependencies). In this work, our proposed model, the *T-SRM* model (as described in chapter 3, *Methodology*) tries to resolve this drawback.

Probabilistic Model

The *Probabilistic Model* introduced by Robertson and Sparck Jones [43] and later became known as the *binary independence retrieval* model. The fundamental idea is that given a user query, there is a set of documents which contains exactly the relevant documents and no other (an ideal answer set). Given the description of this ideal answer set, we would have no problems in retrieving its documents. Thus, we can think of the querying process as a process of specifying the properties of an ideal answer set. The probabilistic model is based on the assumption that given a user query *q* and a document d_j in the collection, the probabilistic model tries to estimate

the probability that the user will find the document d_j interesting (i.e. relevant). Documents in the set *R* are predicted to be *relevant* to the query. Documents not in this set are predicted to be *non-relevant*. This assumption is quite troublesome because it does not state explicitly how to compute the probabilities of relevance [4].

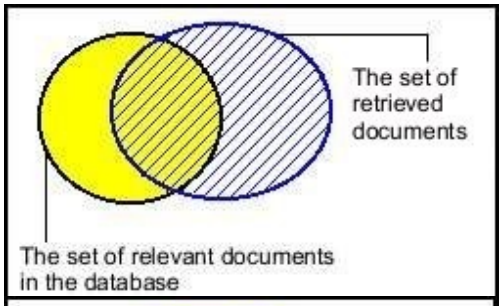
The main advantage of the probabilistic model [4] is that documents are ranked in decreasing order of their probability of being relevant. The disadvantages include: i) the need to guess the initial separation of documents into relevant and non-relevant sets, ii) the fact that the method does not take into account the frequency that an index term occurs in the document (i.e. all weights are binary) iii) the assumption that all index terms are independent.

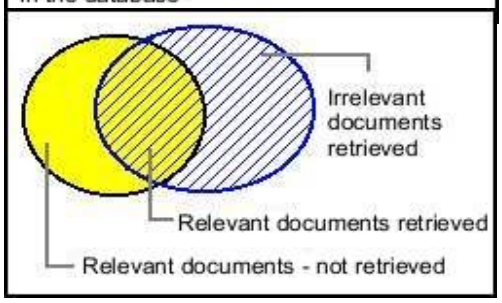
2.2.3 *Comparison of Models*

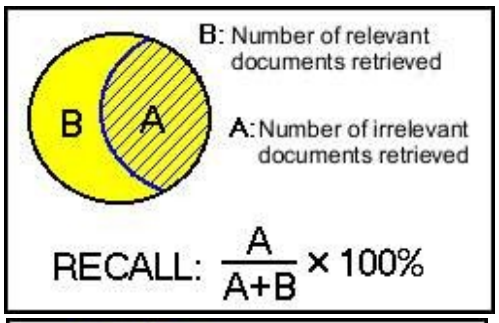
The Boolean model in general is considered to be the weakest from the methods described briefly above [4]. In comparing vector with probabilistic, Salton and Buckley showed that the vector model is expected to outperform the probabilistic model [50]. This also seems to be the most popular thought among researchers, practitioners and the Web community. For this reason, we choose to implement the vector space model (see also chapter *4*) to compare the results with out proposed method, the *Term based Similarity Retrieval Model* (*T-SRM*).

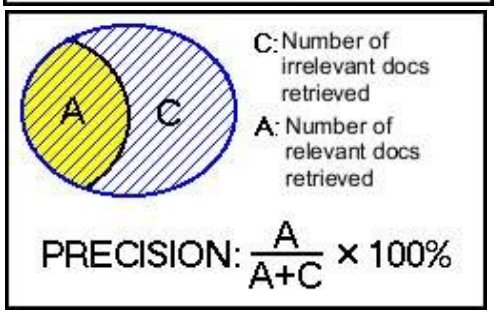
2.2.4 *Performance Evaluation*

The evaluation of the performance of retrieval is usually based on a test reference collection and on an evaluation criterion. The test reference collection consists of a collection of documents, a set of example queries and a set of relevant documents (provided by specialists) for each query. Given a retrieval strategy *S*, the evaluation criterion measures the *similarity* between the set of documents retrieved by *S* and the set of relevant documents provided by the specialists. This provides an estimation of the goodness of the retrieval strategy [4]. In our experiments, we apply the two most common retrieval evaluation measures, referred to as *recall* and *precision*.









As shown in the first two figures on the left, these measures assume:

1. There is a set of documents in the database which is relevant to the search topic
2. Documents are assumed to be either relevant or irrelevant *(these measures do not allow for degrees of relevancy)*.
3. The actual retrieval set may not perfectly match the set of relevant documents.

Recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the database. It is usually expressed as a percentage.

Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is usually expressed as a percentage.

Measuring recall is difficult because it is often difficult to know how many relevant records exist in a database. This would require that the whole database is browsed and each document is judged as relevant or non relevant. This is particularly difficult for large document collections. Often recall is estimated by identifying a pool of relevant records ("pooling method", [27]) and then determining what proportion of the pool retrieved by the search. There are several ways of creating a pool of relevant records: one method is to use all the relevant records found from different searches, another is to manually scan several journals to identify a set of relevant answers. Of course there are alternative measures to evaluate a retrieval method. Some of them are the *E*

measure, the harmonic mean, satisfaction etc. [4].

Chapter 3. Methodology

Our proposed method aims towards extracting meaningful multi-word terms from documents. It shows furthermore that it is possible to exploit this result for enhancing the performance of information retrieval in document collections. This is achieved through *T-SRM*, a novel information retrieval method, which uses vectors of multi-word terms as document representations and appropriately defined similarity measures for computing document similarity as a function of the similarity between individual multi-word terms. Our method works in the following steps:

- Corpus selection
- Term Extraction
- Multi-Word Term similarity computation
- Weight computation
- Term-Based Document Similarity

3.1 Considerations in Corpus Selection

Corpus selection is an essential prerequisite for testing our method. Our corpus should be in text format and representative of a domain. Also, should be a collection potentially used in IR tasks. Moreover, in order to benchmark our method against other methods, we should use a standard reference corpus. TREC[1] meets our requirements. TREC, The Text Retrieval Conference, has purpose to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. Each of these collections consists of a set of documents, a set of topics (queries), and a corresponding set of relevance judgements (correct answers to the queries).

Another consideration is the corpus size. In particular we need our corpus length to be

1 http://trec.nist.gov/

big enough to help us make reliable judgements using statistical methods.

In the Implementation chapter (section *4.1)* we describe the corpus fulfilling our requirements and its particular characteristics.

The method for multi-word term extraction, referred to as the *C-NC Value* method [15], is discussed in the next section (section *3.2*).

3.2 The C-NC Value Method [15]

The C-Value method is a hybrid domain-independent method combining linguistic and statistical information (with emphasis on the statistical part) for the extraction of multi-word and nested terms (i.e. terms that appear within other longer terms, and may or may not appear by themselves in the corpus). This method takes as input a corpus and produces a list of candidate multi-word terms, ordered by the likelihood of being valid terms, namely their C-Value measure.

The C/NC-Value method comprises of two main parts:

i. The linguistic

ii. The statistical, which consist of:

   (a) C-Value

   (b) NC-Value

   NC-Value is an enhancement to C-Value. It incorporates contextual information aiming at improving the ranking of candidate multi-word term list extracted by C-Value.

3.2.1 Linguistic Processing

The Linguistic Processing applies the following steps:

- *The Part-of-Speech(POS) tagging of the corpus:*

  Part-of-Speech(POS) tagging is the process of assigning a grammatical category tag (such as noun, verb, adjective, adverb or preposition) to each word. In *C/NC-Value*, *POS* is applied prior to linguistic filters that exact noun phrases.

- *The Linguistic Filter:*

  Terms consist mostly of nouns and adjectives [17] and sometimes prepositions [9]. The statistical information , without any linguistic filtering, is not enough to produce useful results. Without any linguistic information, undesirable strings such as "of the", "is a", "etc.", would also be extracted. The linguistic filter is used to extract noun phrases that constitute multi-word terms discarding such undesirable strings.

- *The Stop-list.*

  A stop-list for a corpus in ATR is a list of words which are not expected to occur as term words in that domain. It is used to avoid the extraction of strings that are unlikely to be terms, improving the precision of the output list. The stop list is manually constructed based on domain observation. Figure *3.1* shows a sample list of stop words.

became	because	becoming	before	beforehand	behind
become	becomes	been	being	believe	below
better	best	beside	even	does	every
far	cant	did	day	etc.	fold

Figure 3.1 *Example of stop words*

3.2.2 The Statistical Part

(i) C-Value

The C-value constitutes a measure of the importance of each candidate term extracted in the previous steps. The higher the C-Value measure the more likely it is the candidate term to be a valid term.

The C-Value of a term is computed as follows:

$$C-value(a)=\begin{cases}\log_2|a|*f(a) & a \text{ is not nested}\\ \log_2|a|*(f(a)-\frac{1}{P(T_a)}\sum_{b\in T_a}f(b)) & otherwise\end{cases}$$

where :

- α is the candidate string
- $f(.)$ the frequency of occurrence in corpus
- T_a the set of extracted candidate terms that contain α (longer candidate

terms)

- $P(T_a)$ the number of these longer candidate terms

The negative effect on the candidate string α being a substring of other longer candidate terms is reflected by the negative sign '-' in the formula above. The independence of α from these longer candidate terms is given by $P(T_a)$. The greater this number the bigger its independence(and the opposite) is reflected by having $P(T_a)$ as the denominator of a negatively signed fraction. The measure is built using several statistical characteristics of the candidate string. These are:

1. The total frequency of occurrence of the candidate string in the corpus.
2. The frequency of the candidate string as part of other longer candidate terms.
3. The number of these longer candidate terms.
4. The length of the candidate string (in number of words).

Figure 3.2 *Characteristics of the candidate string used for calculating C-Value*

The higher the number of distinct longer terms that our string appears as nested in , the more certain we can be about its independence (i.e. that the candidate term extracted is a real term). The fact that a longer string appears X times is more important than that of a shorter string appearing again X times[15]

(ii) NC-Value

NC-Value is an enhancement to *C-Value* that is computed based on context information.

Firstly *NC-Value* creates a list of important term context words. Term context words are words that appear in the vicinity of terms in texts. These will be ranked according to their `importance' when appearing with terms. The criterion for the extraction of a word as a term context word is the number of terms it appears with. The higher this number is, the higher the likelihood that the word is `related' to terms (it occurs with other terms in the same corpus).

Each candidate term in the C-Value list appears in the corpus with a set of context words. From these context words, the nouns, adjectives and verbs are retained for each candidate term. *NC-Value* provides a method for the extraction of term context words (words that tend to appear with terms) and incorporates this information (from term context words) into the term extraction process. This above criterion is more formally expressed as[15] :

$$Weight(w) = \frac{t(w)}{n} \quad (figure\ 3.3)$$

where

- *w* is the context word (noun, verb or adjective) to be assigned a weight as a term context word,
- *Weight(w)* the assigned weight to the word w,
- *t(w)* the number of terms the word w appears with,
- *n* the total number of terms considered.

The purpose of the denominator *n* is to express this weight as a probability (the probability that the word *w* might be a term context word). The NC-value measure is then computed as :

$$NC-Value = 0.8CValue(a) + 0.2\sum_{b \in C_a} f_a(b)\,weight(b)$$

where

- *a* is the candidate term,
- C_a is the set of distinct context words of a,
- *b* is a word from C_a,
- $f_a(b)$ is the frequency of *b* as a term context word of *a*,
- *weight(b)* is the weight of *b* as a term context word.

The two factors of NC-value, i.e. C-value and the context information factor, have been assigned the weights 0.8 and 0.2 respectively. These have been chosen among others after experiments and comparisons of the results [15].
We describe thoroughly the *C/NC-Value* algorithms during the implementation section (*chapter 4*).

3.3 Multi-word similarity measures

In this section, we present the method for computing the similarity between multi-word terms extracted by *C-NC Value* (as described in section *3.2*). The multi-word term similarity measures are to be used in combination with our proposed *T-SRM (Term-based Similarity Retrieval Model)* in information retrieval tasks – for retrieving results to user queries.

An approach for computing similarity between multi-word terms is proposed in [21]. It takes into consideration both lexical and contextual criteria which are combined into a unique formula. In this work we apply machine learning for measuring the relative importance of the two criteria.

3.3.1 *Lexical Similarity*

The *Lexical Similarity* between multi-word terms measures the similarity between the words that constitute terms. This idea was exploited by Bourigault and Jacquemin [22] by adapting the term variation process, and by Dagan and Church [18] via "grouping" the list of term candidates according to their heads[2]. These approaches were generalized by considering constituents (head and modifiers) shared by terms. Therefore the rationale behind lexical similarity involves the following hypotheses[21]:

1. Terms sharing a head are assumed to be (in)direct hyponyms of the same term *(e.g. progesterone receptor* and *oestrogen receptor* are both receptors*)*
2. When a term is nested inside another term , we assume that the terms are related *(e.g. retinoic acid receptor and retinoic acid should be associated)*

Lexical similarity between two terms is defined based on based on identifying their respective common subsequences[21]. By comparing all their non-empty sub-sequences, it is possible to give more credit to pairs of terms that share longer nested constituents. An additional credit is given to terms having common heads.

Given a sequence of words *s*, P(s) (according to Nenadic et al. [21]) refers to the set of all sub-sequences in *s*. For example, P(*orphan nuclear receptor*) = {*orphan,*

2 i.e for "web page" and "web snippet" the head is the word "web".

nuclear, receptor, orphan nuclear, nuclear receptor, orphan nuclear receptor}. The lexical similarity between term t_1 and term t_2 (whose heads are denoted by h_1 and h_2 respectively) is computed according to a Dice-like coefficient formula. Dice coefficient is widely used in information retrieval [33]. Building upon the idea of Dice coefficient [33], the lexical similarity between two terms is computed as :

$$LS(t_1, t_2) = \frac{|P(h_1) \cap P(h_2)|}{|P(h_1)| + |P(h_2)|} + \frac{|P(t_1) \cap P(t_2)|}{|P(t_1)| + |P(t_2)|}$$

The numerators in the previous formula denote the number of shared constituents, while the denominators refer to the sums of total numbers of constituents. The following table illustrates some examples.

i	t_i	$P(t_i)$
1	*nuclear receptor*	*{nuclear, receptor, nuclear receptor}*
2	*orphan receptor*	*{orphan, receptor, orphan receptor}*
3	*orphan nuclear receptor*	*{orphan, nuclear,receptor, orphan nuclear, nuclear receptor, orphan nuclear receptor}*
4	*nuclear orphan receptor*	*{nuclear, orphan, receptor, nuclear orphan, orphan receptor, nuclear orphan receptor}*
$LS(t_1,t_2)=0.67$, $LS(t_1,t_3)=0.83$, $LS(t_1,t_4)=0.72$, $LS(t_2,t_3)=0.72$, $LS(t_3,t_4)=0.75$		

Examples of lexical similarity

3.3.2 *Contextual Similarity*

According to Nenadic et. al. [21], contextual similarity is mainly based on the Harris' notion of substitutability: If two terms can substitute each other in similar *contexts,* then they can be deemed similar. For example the term*"ligand inducible transcription factor"* (*following table)* typically appears in a context that can be described as "*belonging to a superfamily of"*.

More specifically, the contexts of terms similar to the term 'ligand-inducible

transcription factor' are:

- *...T3R belongs to the nuclear receptor family of* ***ligand-inducible transcription factors****...*
- *...The retinoid receptors belong to a large superfamily of l****igand-inducible transcription factors****...*
- *...this receptor is a novel member of the superfamily of* ***ligand-inducible transcription factor****....*

So this description follows a certain *context pattern :*

<term> (belong | member of) <modifier> *superfamily of* **ligand-inducible transcription factor**

In place of *<term>* different terms may appear and all these terms are mutually associated. Such context patterns can be used to establish term similarities [21]. As context patterns are treated as term features, the Dice-like coefficient is used to estimate contextual similarity between terms as a function of both common and distinctive features. The Contextual similarity measure is formally defined as :

$$ContextualSim(t_1, t_2) = \frac{|C_{L1} \cap C_{L2}| + |C_{R1} \cap C_{R2}|}{|C_{L1}| + |C_{L2}| + |C_{R1}| + |C_{R2}|}$$

where: $C_{L1}, C_{R1}, C_{L2}, C_{R2}$ are sets of left and right context patterns associated with terms t_1 and t_2 respectively.

3.4 Combination of Similarities using Machine Learning

Neither lexical, nor contextual similarity are sufficient on their own to define term similarity measure between two arbitrary terms**:** Lexical similarity can indicate only restricted types of similarity (hyponymy and meronymy). Regarding contextual similarity, if a term appears infrequently or within very specific context patterns, the number of its context patterns will influence its contextual similarity to other terms. Thus, a combination of these similarities is required.

In this work the relative importance of each similarity measure is computed by machine learning. An existing platform of machine learning algorithms is WEKA[1]. WEKA is self-described in its web page as "a collection of machine learning algorithms for data mining tasks". The algorithms can either be applied directly to a dataset or called from Java source code. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. WEKA is an open source software issued under the GNU General Public License.

We tried several algorithms to combine the similarities. These algorithms along with the results are briefly explained in the implementation chapter (*4.5*). In the next section (*3.6*) we propose the *Term-Based Similarity Retrieval Model (T-SRM)* , an information retrieval model based on the multi-word term similarity method described above.

3.5 The Term-based Similarity Retrieval Model (T-SRM)

Our hypothesis is that conceptual similarity may be learnt directly from our text corpora and that conceptual similarity may be expressed as combination of lexical and contextual similarities. Building upon this idea, we propose the Term-Based Similarity Retrieval Model (*T-SRM*), a novel information retrieval model which s capable of computing the similarity between documents containing conceptually similar but not necessarily lexically similar terms. Notice that classic retrieval models such as the Vector Space Model (section *2*), will not recognize synonyms or conceptually similar terms (e.g. "google search engine", "web search engine").

Our *T-SRM* model suggests computing the similarity between documents based on the similarity of multi-word terms contained in the documents (i.e. based on the lexical and contextual similarity measures as described in section *3.3*). In our *T*-SRM model the queries may also be augmented with conceptually similar terms which are retrieved by applying similarity measure to the terms extracted from the corpus examined. Our model takes all possible term associations between two documents into account and accumulate their similarities. Similarly to VSM, queries and

1 http://www.cs.waikato.ac.nz/ml/weka/

documents should be first analysed and represented by term vectors. Very infrequent or very frequent terms should be eliminated. Each term in this vector should be represented by its weight. The weight of a term should be computed as a function of its frequency of occurrence in the document collection. In particular in our approach the term frequency - inverse document frequency (*tf*idf*) model is used for computing the weight. Typically, the weight d_i of a term *i* in a document is computed as

$$d_i = tf_i \cdot idf_i$$

where tf_i is the frequency of term *i* in the document and idf_i is the inverse document frequency of *i* in the whole document collection.

Then the *T-SRM* computes:

- **Document Similarity:**

  The similarity between an expanded query *q* and a document *d* is computed as:

$$Sim(q, d) = \frac{\sum_i \sum_j q_i d_j sim(i, j)}{\sum_i \sum_j q_i d_j}$$

  where *i* and *j* are terms in the query and the document respectively. Query terms are expanded according to the previous step. All *tf*idf* terms and documents(queries) are normalized by document (query) length. The measure is normalized in range[0,1].

An alternative would be to use *C-Values* for term weighting. Furthermore, the formula by Mihalcea [40] could be applied for computing similarity between documents. According to Mihalcea, The similarity between the input documents T1 and T2 is determined using the following scoring function:

$$Sim(T_1, T_2) = \frac{1}{2}\left(\frac{\sum_{w \in T_1} (maxSim(w, T_2) * idf(w))}{\sum_{w \in T_1} idf(w)} + \frac{\sum_{w \in T_2} (maxSim(w, T_1) * idf(w))}{\sum_{w \in T_2} idf(w)}\right)$$

Mihalcea defines the semantic similarity of two documents T_1 and T_2 using a metric that combines the similarities of each document in turn with respect to the other

document. First, for each word w in the document T_1 tries to identify the word in the segment T_2 that has the highest semantic similarity (maxSim(w,T_2)). Next, the same process is applied to determine the most similar word in T_1 starting with words in T_2.

Chapter 4. Implementation

The implementation of the method is discussed below. The method is implemented in Java and the main steps are :

1. Corpus preprocessing
2. C/NC Value computation
3. Multi-Word Similarity computations
4. Combination of Similarities
5. T-SRM document similarity computation

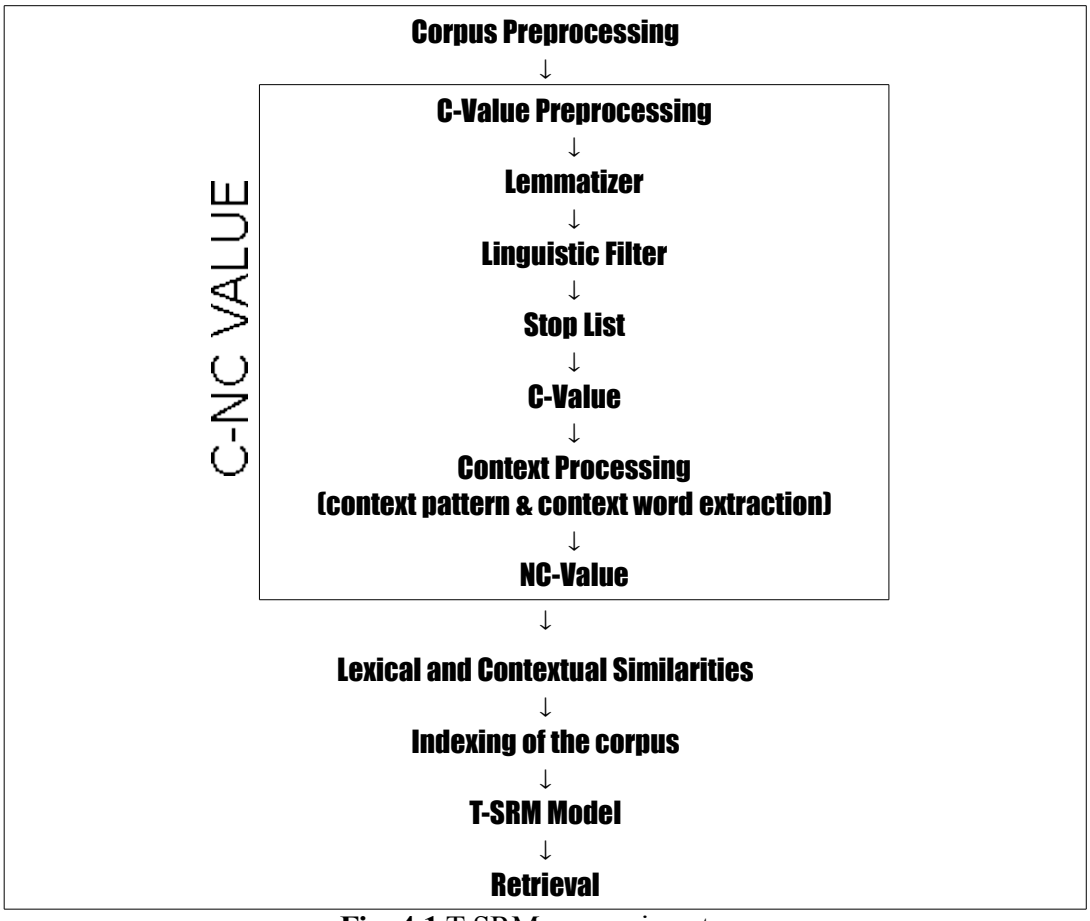


Fig. 4.1 T-SRM processing steps

4.1 Corpus Preprocessing

We test out methods on a standard TREC corpus. OhsuMed collection, due its large size and thorough experimentation, is considered to be a standard reference test collection. OhsuMed is a set of 348,566 references from Medline, the on-line medical information database, consisting of titles and/or abstracts from 270 medical journals over a five-year period (1987-1991). The OhsuMed document collection was compiled by William Hersh and colleagues, for the experiments described in [38], [39].

OhsuMed provides queries and a set a documents that are relevant to each query respectively. All other documents are considered irrelevant. This helped us measuring precision and recall automatically in our IR experiment.

4.2 C-NC Value

The C/NC-Value method is implemented in the following steps:

4.2.1 Preprocessing

Corpus preprocessing applied the following:

- Sentence Splitting
- Tokenizing
- POS tagging

We used initially the tools from GATE[1] for Part-of-Speech Tagging , tokenizing and sentence splitting. However, our first experiments showed that the corresponding collection from OpenNLP[2] was much faster and much more stable. For this reason we decided to use the OpenNLP tools for the preprocessing stage.

The OpenNLP POS tagger that we used is the MX-POST, based on the work of Ratnaparkhi on maximum entropy models[3] for natural language processing [36] .

1 http://gate.ac.uk :: GATE is an infrastructure for developing and deploying software components that process human language.

2 http://www.opennlp.org :: OpenNLP is an organizational center for open source projects related to natural language processing. Hosts a variety of java-based NLP tools which perform sentence detection, tokenization, pos-tagging, chunking and parsing, named-entity detection, and coreference using the OpenNLP Maxent machine learning package.

3 *(quoting from Manning and Schutze)* [37] (page 589): Maximum entropy modeling is a framework for integrating information from many heterogeneous information sources for classification. The data for a classification problem is described as a (potentially large) number of features. These features can be quite complex and allow the experimenter to make use of prior

The tagset (a sample in table 4.1) used is the Penn-TreeBank tagset [16]. After POS Tagging, the linguistic filter that extracts phrases that satisfy the linguistic criterion and frequency threshold is applied next.

POS Tag	***Description***	***Example***
CC	coordinating conjunction	and
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJS	adjective, superlative	greenest
NN	noun, singular or mass	table
NNP	proper noun, singular	John
RB	adverb	however, usually, naturally, here, good
VB	verb, base form	take
VBZ	verb, 3rd person sing. present	takes
PRP	personal pronoun	I, he, it
LS	list marker	1)

Table 4.1 *Description and Examples of sample POS tags*

Table *4.2* shows a sample paragraph before and after the preprocessing respectively.

The original text
This survey focuses on clustering in data mining. Data mining adds to clustering the complications of very large datasets with very many attributes of different types. This imposes unique computational requirements on relevant clustering algorithms. A variety of algorithms have recently emerged that meet these requirements and were successfully applied to real-life data mining problems. They are subject of the survey.
After POS Tagging , Sentence Splitting, Tokenizing
This/DT survey/NN focuses/VBZ on/IN clustering/VBG in/IN data/NNS mining/NN ./. Data/NNP mining/NN adds/VBZ to/TO clustering/VBG the/DT complications/NNS of/IN very/RB large/JJ datasets/NNS with/IN very/RB many/JJ attributes/NNS of/IN different/JJ types/NNS ./. This/DT imposes/VBZ unique/JJ computational/JJ requirements/NNS on/IN relevant/JJ clustering/NN algorithms/NNS ./. A/DT variety/NN of/IN algorithms/NNS have/VBP recently/RB emerged/VBN that/IN meet/VB these/DT requirements/NNS and/CC were/VBD successfully/RB applied/VBN to/TO real-life/JJ data/NNS mining/NN problems/NNS ./. They/PRP are/VBP subject/JJ of/IN the/DT survey/NN ./.

Table 4.2 *Preprocessing stage sample*

knowledge about what types of informations are expected to be important for classification. Each feature corresponds to a constraint on the model. We then compute the maximum entropy model, the model with the maximum entropy of all the models that satisfy the constraints.

Detection of morphological variants[1]

We proposed and implemented an enhancement to C-Value, using the morphological processor from WordNet[2] Java Library[3] (*JWNL*) . *JWNL* is an API for accessing WordNet-style relational dictionaries. It also provides relationship discovery and morphological processing. The *Morphological Processor* attempts to match the form of a word or phrase to its respective lemma, i.e. base form, in WordNet. For example, if one calls lookupBaseFom(POS.VERB, "running"), the lemma "run" should be returned. This enhancement has been thought important because it allows the C/NC Value tool to handle morphological variants of terms , for example “web page”, “web pages”.

Linguistic filtering and stop-word removal

At this point the linguistic filter is applied on each sentence to extract potential multi-word terms. Moreover, if a constituent word in each candidate extracted multi-word term resides in the stop list, then the candidate term is rejected.

The choice of linguistic filter affects the precision and recall of the output list. A `closed' filter which is strict about the strings it permits, will have a positive effect on precision but a negative effect on recall. As an example, consider the “*Noun+*” filter that Dagan and Church used [18]. This filter permits only noun sequences and as a result produces high precision since noun sequences in a corpus are the most likely to be terms. At the same time, it negatively affects recall, since there are many noun compound terms that consist of adjectives and nouns, which are excluded by this filter. We implemented all the filters below and we chose the third one for our experiments, the open filter which results in augmented recall over precision (to reveal more candidate terms):

1. Noun+Noun,
2. (Adj | Noun)+Noun,
3. ((Adj | Noun)+|((Adj | Noun)*(NounPrep)?)(Adj | Noun)*)Noun

After this step, main process of C-Value takes place.

1 The notion of term variants is described in chapter 2
2 http://wordnet.princeton.edu/
3 http://jwordnet.sourceforge.net/ :: JWNL is an API for accessing WordNet-style relational dictionaries. It also provides functionality beyond data access, such as relationship discovery and morphological processing.

4.2.2 C-Value process – The algorithm - Implementation

We describe the steps taken in the C-value method to construct a list of candidate terms from a corpus.

- ***Step 1***

We tag the corpus. As mentioned earlier, we need the tagging process since we will use a linguistic filter to restrict the type of terms to be extracted.

- ***Step 2***

This stage extracts those strings that satisfy the linguistic filter and frequency threshold. The terms will be extracted from among these strings. According to Nenadic et al. [21] the maximum length of the extracted strings depends on:

1. The working domain. In arts for example, terms tend to be shorter than in science and technology.

2. The type of terms we accept. Terms that only consist of nouns for example, very rarely contain more than 5 or 6 words.

The process of finding the maximum length is as follows: We attempt to extract strings of a specific length. If we do not find any strings of this length, we decrease the number by 1 and make a new attempt. We continue in this way until we find a length for which strings exist. At this point, extraction of the candidate strings can take place. Initially, a list of strings of each length is created, (i.e. a list for the bigrams, a list for the trigrams, etc.). The lists contain the strings with their frequency of occurrence. The lists are then filtered through the stop-list and are concatenated. The longest strings appear at the top, and decrease in size as we move down, with the bigrams being at the bottom.

- ***Step 3***

The C-value for each of the candidate strings is computed, starting with the longest ones and finishing with the bigrams. The C-value for the longest terms is given by the top branch of formula described in the C-Value section (section *3.2.2*) and we quote again:

$$C-value(a)=\begin{cases}\log_2|a|*f(a) & a \text{ is not nested}\\ \log_2|a|*(f(a)-\frac{1}{P(T_a)}\sum_{b\in T_a} f(b)) & otherwise\end{cases}$$

We set a C-value threshold, so that only those strings with C-value above this threshold are added onto the list of candidate terms. For the evaluation of C-value for any of the shorter strings, we need two more parameters:

- their frequency as part of longer candidate terms
- the number of these longer candidate terms

To obtain these two parameters, we perform the following:
For every string *a*, that it is extracted as a candidate term and for each substring *b* of *a* , we compute tuples *(f(b), t(b), c(b))*, where
f(b) is the total frequency of b in the corpus,
t(b) is the frequency of b as a nested string of candidate terms,
c(b) is the number of these longer candidate terms.

When this triple is first created, *c(b)* = 1 and *t(b)* equals the frequency of *a*. Each time *b* is found after that, *t(b)* and *c(b)* are updated, while *f(b)*, its total frequency, does not change.
c(b) and *t(b)* are updated in the following manner:
c(b) is increased by 1 every time *b* is found within a longer string a that is extracted as a candidate term. *t(b)* is increased by the frequency of the longer candidate term *a*, *f(a)*, every time b is found as nested. If *n(a)* is the number of times a has appeared as nested, then *t(b)* will be increased by *f(a) – n(a)*.
Now in order to compute C-value for a string *a* which is shorter by one word, we either already have for it a triple *(f(a), t(a), c(a))*, or we do not. If we do not, we calculate the *C-value* from the top branch of the above *C-Value* formula. If we do, we use the bottom branch respectively. In that case, *P(Ta) = c(a)* and $\sum_{b\in T_a} = t(a)$.
After the calculation of C-value for strings of length *l* finishes, we move to the computation of *C-value* for strings of length *l*-1 . This way it is evident whether the string to be processed has been found nested in longer candidate terms.
At the end of this step, a list of candidate terms has been built. The strings of the list

are ranked by their C-value.

Table *4.3* shows a part of the output list after the application of C-Value on our computer science corpus. The higher a term is in this hierarchy , the higher its probability of being a real term [15]

Extracted Term	*C-Value*
web page	1260.2
search engine	881.0
information retrieval	492.333
similarity measure	412.667
web site	409.2
machine learning	405.0
prior specific permission	340.763
natural language processing	339.57
clustering algorithm	337.909
IEEE transactions	325.0
semantic web	306.142
relevant document	305.0
relevant page	283.0
information extraction	262.0
objective function	248.0
knowledge base	241.6
subject descriptor	240.0

Table 4.3 *C-Value output sample*

We then proceed to the next phase,i.e. the *NC-Value* computation.

4.2.3 NC-Value computation

NC-Value is computed to eachcandidate term as follows:

- ***Step 1***

We apply the C-value method to the corpus. The output of this process is a list of candidate terms, ordered by theirC-value.

- ***Step 2***

This involves the extraction of the term context words and their weights. These will be used in the third stage to improve the term classification in the extracted list. In order to extract the term context words, we need a set of top C-Value terms, according to [15]. We use the `top' candidate terms from the C-value list, which are expected to present very high precision on valid terms[4]. We chose the first 1500 terms for running our experiments. These `top' terms produce a list of term context words and assign to each of them a weight following the process described in the previous section.

- ***Step 3***

This involves incorporating of context information acquired from the second stage of the extraction of multi-word terms. The C-value list of candidate terms extracted during stage one is re-ranked using context information, so that the real terms appear closer to the top of the list than they did before, i.e. the collection of real terms at the top of the list increases while the collection of those at the bottom decreases. The re-ranking takes place in the following way: Each candidate term from the C-value list appears in the corpus with a set of context words. From these context words, we retain the nouns, adjectives and verbs for each candidate term. These words may or may not have been met before, during the second stage of the creation of the list with the term context words. In the case where they have been met, they retain their assigned weight. Otherwise, they are assigned zero weight. For each candidate term, we obtain the context factor by summing up: the weights for its term context words, multiplied by their frequency of co-occurrence with this candidate term.

For example, let us assume that the candidate word W appears 10 times with the context word c_1, 20 times with the context word c_2, and 30 times with the context word c_3. Let us assume also, that the weight for c_1 is w_1, the weight for c_2 is w_2, and the weight for c_3 is w_3. Then, the context factor for *W (The definition of weights is given in methodology chapter, section 3.2.2)* is :

$$10\ w_1 + 20\ w_2 + 30\ w_3$$

The above description is the second factor of the NC-value measure which re-ranks

4 Note also that C/NC Value was primarily designed for the detection of domain terms. Our application of C/NC Value is in indexing. Therefore any not valid domain terms are not expected to cause problems in retrieval.

the C-value list of candidate terms. The first factor is the C-value of the candidate terms. The whole NC-value measure is formally described as

$$NC-Value = 0.8\,CValue(a) + 0.2 \sum_{b \in C_a} f_a(b)\, weight(b) \quad (figure\ 4.4)$$

where

- a is the candidate term,
- C_a is the set of distinct context words of a,
- b is a word from C_a,
- $f_a(b)$ is the frequency of b as a term context word of a,
- $weight(b)$ is the weight of b as a term context word.

The two factors of NC-value, i.e. C-value and the context information factor, have been assigned the weights 0.8 and 0.2 respectively. These have been chosen among others after experiments and comparisons of the results. [15]

4.3 Multi-Word Similarity measures

During the implementation of contextual similarity (as described in chapter *3*), we had to consider two types of context pattern constituents: morpho-syntactic (such as noun and verb phrases, prepositions, etc.) and terminological (i.e. term occurrences). Morpho-syntactic constituents like verb and noun phrases can be identified by applying a chunker (which recognize syntactic noun phrases {NPs}, verb phrases {VPs}), while terminological entities can be recognized by an ATR processor *(like C-NC Value as described in section 3.2)*.

We generated all possible "linearly nested" patterns for each given context. In particular, when considering left contexts, contexts of the maximal length (without crossing the sentence boundary) are initially selected, and they are then iteratively trimmed on the left side until the minimal length is reached. Right contexts are treated analogously. The following example illustrates the left linear pattern generation process:

V	PREP	TERM	NP	PREP	(the maximal pattern)
	PREP	TERM	NP	PREP	
		TERM	NP	PREP	
			NP	PREP	(the minimal pattern)

NP stands for 'Noun Phrase', a basic syntactic structure.
During our implementation we experimented with various maximal pattern lengths. Based on results observation, we decided to set the minimum pattern length to 2 and the maximum to 8. We assume that longer contexts are more important for assessing term similarities[23].

Some of the syntactic categories were removed from context patterns, since not all of them are equally significant in providing useful contextual information. Adjectives (that are not part of terms), adverbs and determiners can be removed from context patterns for they rarely bare some specific information [23]. In addition, the so-called "linking words" (e.g. however, moreover, etc.), or more generally, "linking devices" (e.g. verb phrases, such as "result in", "lead to", "entail", etc.) are frequently used in special languages in order to achieve more effective communication [5]. However, these constituents are typically non informative and were eliminated.

During the implementation of contextual similarity, (as described in section *3.3.2*), we needed some syntactic phrase structure information in order to apply various levels of generalizations on the term context. For this reason, we considered using a chunker (a shallow syntax analyser). A chunker applies shallow syntactic parsing analysis. In particular, text chunking consists of dividing a text in syntactically correlated parts of words. A chunker attempts to identify noun phrases and verb phrases in text. For example, the sentence *"He reckons the current account deficit will narrow to only # 1..8 billion in September ."* can be analysed as follows:

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September] .

For the chunker used in our implementation, the OpenNLP chunker, the problem of chunking is viewed as tagging problem[26], where the chunk structure is encoded as tags attach to each word.

In the next section (4.4) we present the methods used to combine the similarity measures (lexical and contextual) to form a final hybrid similarity measure.

4.4 Combination of Similarities

We used the WEKA implementation of machine learning algorithms to produce a linear combination of our similarity measures. The training set is given to WEKA in a specific (*ARFF*) format. The training set for WEKA was consisting of term pairs, their respective similarity measures and their evaluation.

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes . In our implementation, each instance (on a single line) represents a term pair. The training set consisted of a representation of 200 term pairs. The size of the training set was considered adequate for training. The training set had three term similarity features:

(a) lexical similarity measure

(b) contextual similarity measure

(c) human similarity measure

The latter, human similarity measure, was assigned to term pairs by human evaluators on a scale of 0 to 3 where (nominal values):

- 0 : The terms are completely unrelated (No relation between them)
- 1 : The terms are almost unrelated (not completely unrelated)
- 2 : The terms are very similar (not exactly similar)
- 3 : The terms have the same meaning

Some of the WEKA algorithms required nominal values (like the decision trees) while other functions required numeric values.

After carefully browsing the plurality of the available WEKA algorithms we finally chose SMOreg, Least Median Square and the C4.5 Decision tree based on results observation.

- *SMOreg*

  Implements Alex J.Smola and Bernhard Scholkopf sequential minimal optimization algorithm for training a support vector regression using polynomial or RBF kernels. [28,29]

- *Least Median Square*

  Implements a least median squared linear regression utilizing the existing WEKA Linear Regression[5] class to form predictions. The basis of the algorithm is [32].

- *C4.5 decision tree [30]*

  decision trees classify instances by sorting them down the tree from the root node to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the decision tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute. This process is then repeated at the node on this branch and so on until a leaf node is reached. ID3 and C4.5 are algorithms introduced by Quinlan for inducing Classification Models, also called Decision Trees, from data [30].

A web interface was built to test the output predictions from the C4.5 prediction tree (*figure 4.6*).

5 Class for using linear regression for prediction. Uses the Akaike criterion for model selection, and is able to deal with weighted instances.

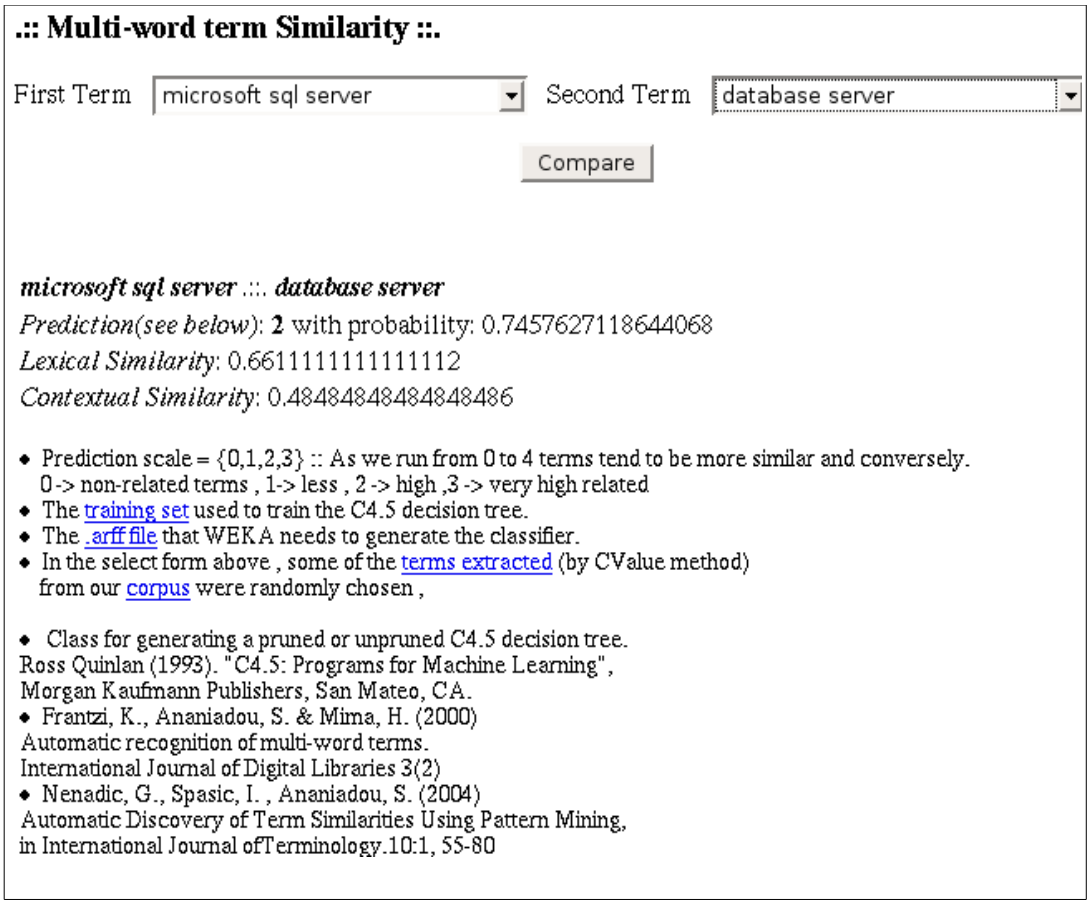


Figure 4.6 *Web Interface testing the C4.5 Decision Tree*

WEKA was trained with the input *ARFF* file as described before (*figure 4.5*). *Contextual* and *Lexical* similarity are calculated and then according to these values the decision tree makes a prediction from scale 0 to 3 along with a probability for being correct.

4.5 Information Retrieval

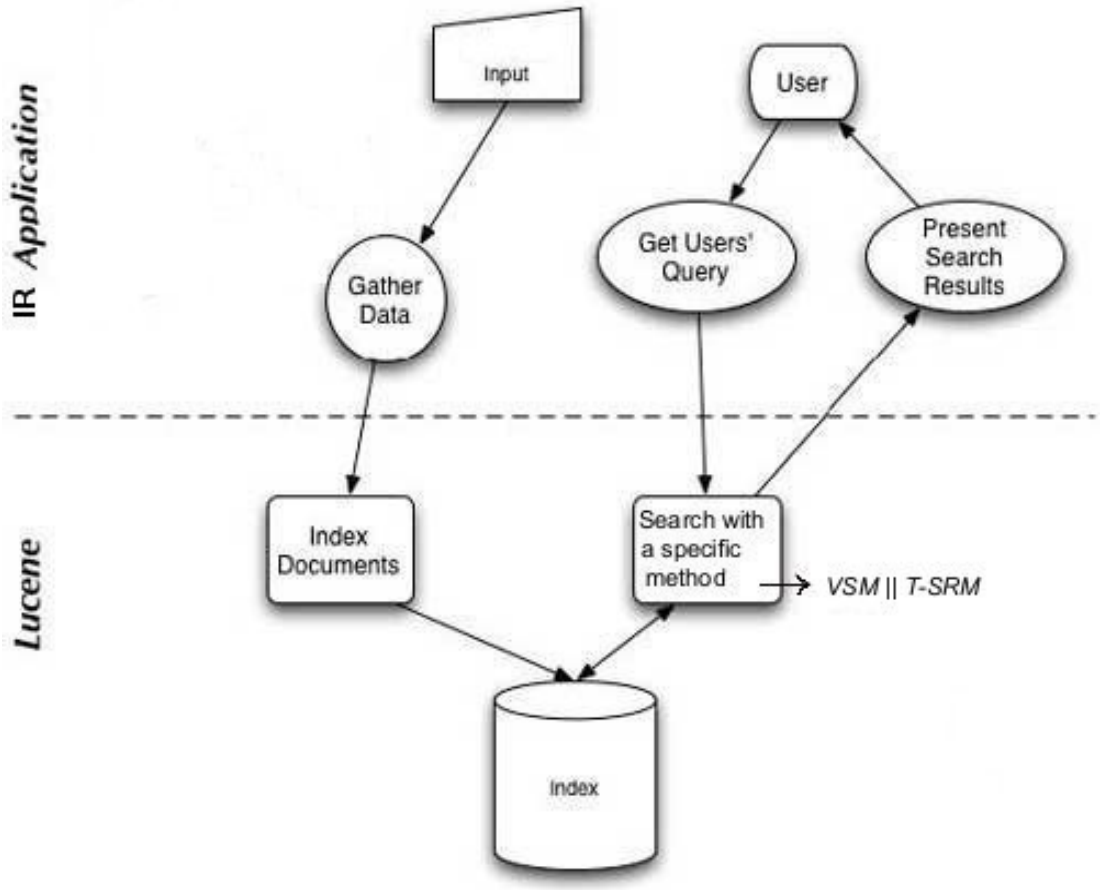
At the heart of all search engines is the concept of *indexing*: processing the original data into a highly efficient cross-reference lookup in order to facilitate rapid searching. Index can be thought as a data structure that allows fast random access to words stored inside it. The concept behind it is analogous to an index at the end of a book, which lets you quickly locate pages that discuss certain topics.

We implemented the indexing of the corpus with Apache Lucene[6]. Lucene is a

6 http://lucene.apache.org

mature ,open-source, high performance scalable Information Retrieval library implemented in Java.

Before text is indexed, it is passed through an Analyser. The *Analyser* is in charge of extracting tokens out of text to be indexed and eliminating the rest. It is an abstract class in Lucene library : We implemented our *Analyser* with the help of the linguistic filter we previously implemented in C-Value section (*4.2*), analysing the data to make it more suitable for searching. To do so, the analyser splits the textual data into tokens (multi-word terms) and then stores them to the index, building term-vectors for each document in the collection.



We implemented the Vector Space Model, as well as the *T*-SRM in Java and we integrated both methods within with Lucene.

The methods we implemented were:

- ✔ classic Vector Space Model (vectors of single word terms)
- ✔ Vector Space Model with vectors of with multi-word terms
- ✔ *T*-SRM
- ✔ *T*-SRM with the C-Value of each multi-word in the document instead of standard *tf***idf* weighting

In the next chapter (*Chapter 5*) we present the evaluation of the above methods in the IR experiments on both corpora. We will also discuss the results of the machine learning process (i.e. the relevant importance of each similarity measure, lexical and contextual respectively). In addition, we will talk about the correlation experiment, tried to compare the output similarity came from WEKA machine learning algorithms in contrast to the human notion of similarity.

Chapter 5. Experimental Results

For the evaluation of our method we carried out the following sets of experiments:

1. Relative importance of lexical and contextual similarity - combination
2. Correlation Experiment
3. Information Retrieval on OhsuMed corpus

5.1 Similarity Relevant Weights Computation

As we note in chapter *4* , none of *lexical* or *contextual* similarities respectively are sufficient on their own to establish term similarities between multi-word terms. With the help of WEKA machine learning algorithms we attempted to combine the two similarity measures.

SMOreg : Final Similarity = *0,7534* * LexicalSim + *0,0723* * ContextualSim - *0,0042*
Least Median Square: Final Similarity = *0,7137* * LexicalSim + *0,322* * ContextualSim + *0,0172*
The method for extracting weights from the resulting C4.5 decision tree: Final Similarity = *0.8* * LexicalSim + *0.2* * ContextualSim

Table 5.1 *Results from lexical&contextual similarity measures into a unique similarity formula*

All methods(described in *4.4)* assigned increased weight on lexical similarity. The combination of *lexical* and *contextual similarity measures,* resulting from each of these, was correlated with human judgements to estimate how close the human notion of term similarity relates to the results.

5.2 Correlation Experiment

Our objective is to discover how close the human notion of similarity (in the same queries) is to the similarity results produced from the methods described above. For the purpose of this experiment, we prepared 50 pairs of multi-word terms. They were randomly selected from the list of terms which C/NC Value extracted from a computer science collection of papers (Lithel corpus[1]). Then, we asked domain experts[2] to provide and estimate for each pair, in a range between 0 (no similarity) and 3 (same meaning). We created a form-based interface on the web and we invited domain experts to enter their evaluation (Table 5.2).

We normalized the results to [0,1] and computed *lexical* and *contextual* similarity for each pair. Then we estimated the final hybrid similarity value for each machine learning algorithm as indicated on the formulas shown in table 5.1.

1 http://www.lithel.ca/Testbed_Corpus/index.html
2 http://www.intelligence.tuc.gr/people

:: Please evaluate the similarity of the above multi-word terms.
Provide with a score between **0** and **3** for each pair:

- Score **0**: The terms are completely unrelated (there is no relation between them)
- Score **1**: The terms are almost unrelated (not completely unrelated)
- Score **2**: The terms are very similar (not exactly similar)
- Score **3**: The terms have the same meaning

	0	*1*	*2*	*3*
1. learning process,training process:	○	○	○	○
2. vector space model,web document:	○	○	○	○
3. google search engine,web search engine	○	○	○	○
4. biomedical terms,mesh terms	○	○	○	○
5. term weighting,term frequency:	○	○	○	○
6. frequent terms,common terms:	○	○	○	○
7. text summarization,automatic text processing:	○	○	○	○
8. web page,web snippet:	○	○	○	○
9. network interface,document relevance:	○	○	○	○
10. average precision,search space:	○	○	○	○
11. text mining,information retrieval:	○	○	○	○
12. xml instance,web search:	○	○	○	○
13. web server,database server:	○	○	○	○
14. candidate web page,web page author:	○	○	○	○
15. web document,web site:	○	○	○	○
16. information extraction engine,information extraction system:	○	○	○	○
17. internet search engine,web search engine:	○	○	○	○
18. web document,cache size:	○	○	○	○
19. vector space model,hypertext links:	○	○	○	○
20. database server,microsoft sql server:	○	○	○	○
21. proxy server,web server:	○	○	○	○
22. web browser interface,web interface:	○	○	○	○
23. training corpus,training phase	○	○	○	○

Table 5.2 . *Interface built to help the evaluation by human domain-experts*

Correlation was computed using the Pearson's correlation function (*equation I*). Suppose we have two variables X and Y , with means $\overline{X}$ and $\overline{Y}$ and standard deviations σ_x and σ_y respectively. The correlation is computed as

$$r = \frac{\sum_{i-1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{(n-1)\sigma_x \sigma_y} \quad (equation\ I)$$

where: X and Y in our experiment stands for the final similarities of each of the methods and of the human evaluation respectively.

Table *5.3* presents a sample part of the results.

Term Pairs	***Human***	***Lexical Sim***	*Context Sim*	***Decision Tree***	***SMO Reg***	***Least Median Square***
learning process,training process	0.71	0.72	0.48	0.64	0.62	0.63
vector space model,web document	0.14	0	0.55	0.17	0.08	0.19
search engine,web search engine	0.67	0.88	0.31	0.71	0.71	0.75
biomedical terms,mesh terms	0.8	0.72	0.26	0.58	0.57	0.61
term weighting,term frequency	0.57	0.72	0.35	0.61	0.59	0.64
frequent terms,common terms	0.67	0.72	0.36	0.61	0.59	0.64
text summarization,automatic text processing	0.43	0.11	0.13	0.12	0.07	0.14
web page,web snippet	0.33	0.72	0.1	0.53	0.53	0.56
network interface,document relevance	0	0	0.37	0.11	0.04	0.14
average precision,search space	0.2	0	0.5	0.15	0.07	0.18
text mining,information retrieval	0.43	0	0.53	0.16	0.08	0.19
. . .	.		. . .	. . .		. . .
xml instance,web search	0	0	0.56	0.17	0.09	0.2
Human correlation	**1**			**0.72**	**0.73**	**0.72**

Table 5.3 *Correlation Experiment Results*

The 1st column denotes the normalized human judgement, while the 2nd and 3rd column denote lexical and contextual similarity for the pair of terms. Respectively, the last 3 columns present the final similarity between the terms, as a combination of lexical and contextual similarity extracted by the machine learning algorithms(*table 5.1*).

It is clear from the correlation score that all chosen methods perform almost equally well. The output weights extracted from all methods were very close (lexical

similarity was given increased weight). Robust regression method [28],[29] returned the best correlation score (*~73%*) with the other methods following very closely.

5.3 OhsuMed corpus IR experiment - Results

The purpose of this experiment is to demonstrate that our method performs better than classic information retrieval models like the Vector Space Model (*VSM*). We experimented with two versions of VSM, one with multi-word term vectors (produced by the C-NC method) and one with single word term vectors as it is typical in IR applications.

We selected 18 queries among Ohsumed's set of queries, to benchmark the information retrieval methods. We chose queries whose title was constituting from 2 words and above. All answers that were not contained in the set that OhsuMed provides, were considered irrelevant.

Figure 5.6 shows a sample OhsuMed query :

<num> Number: MSH282
<title> Anticholesteremic Agents
<desc> Description:
Substances that promote areduction of blood cholesterol levels. (Dorland, 28th ed)

Figure 5.6 *Sample OhsuMed query*

Searching through the instructions and the ReadMe files of the OhsuMed collection, it was not clear whether the relative answers were given based on the title only or the title and the description (i.e. the query was "*Anticholesteremic Agents*" or "*Anticholesteremic Agents, Substances that promote a reduction of blood cholesterol levels*" ?).

We ran *two* experiments on OhsuMed. One experiment using only "titles" as queries and a second one using both "title" and "description" fields as queries. Notice that both types of queries are commonly used in evaluation experiments in the literature. The resulting curve behaviour does not differ significantly among the two experiments, as figures *5.7* , *5.8* denote.

The performance results below indicate that using T-SRM in document representation increases the performance of retrieval in contrast to vector space model using both single word and multi-word term vectors. This may be explained due to the incorporation of conceptual similarity into the T-SRM model. The documents in OhsuMed collection where abstracts, thus the information was dense and the T-SRM took benefit of this. We should also in the future benchmark the performance of our method with full text and more general collections.

The drawback of T-SRM is that our method is noticeably slower than VSM with single word terms. At the same time, because multi-word VSM analyses the query with a more complicated analyser (needs to extract multi-word terms, does not only take single word tokens like simple VSM), this delay is not noticeable comparing T-SRM with multi-word VSM.

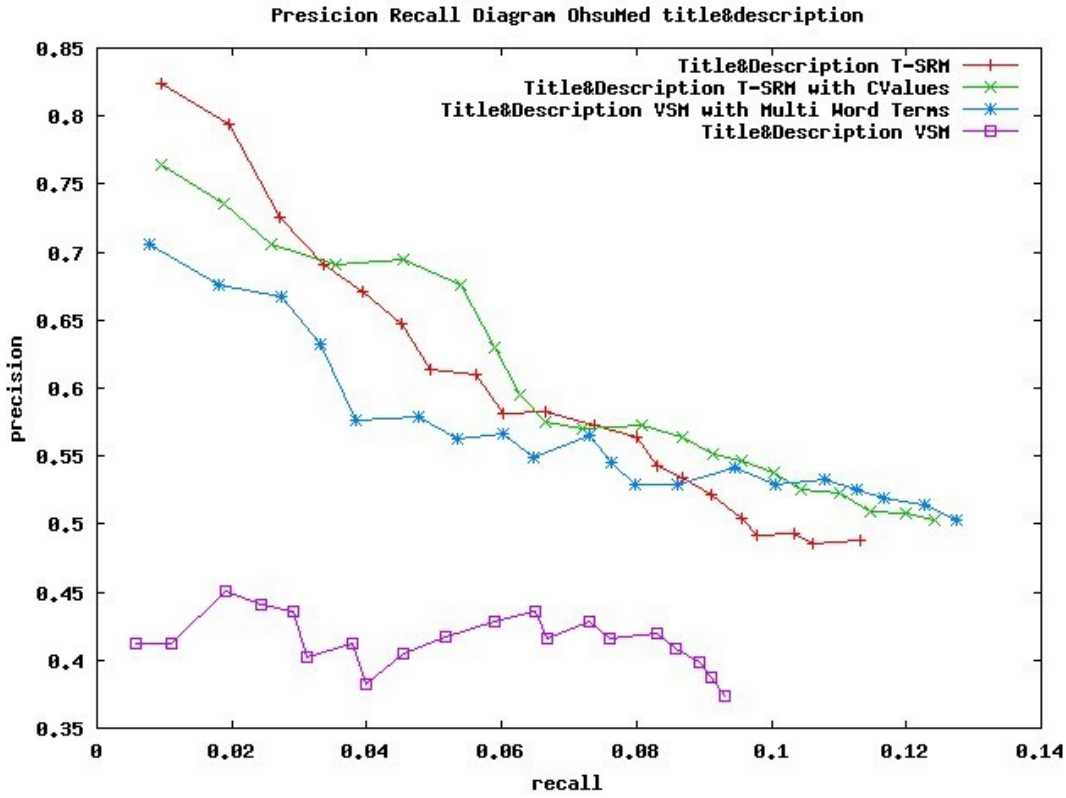


Figure 5.7 *Precision-Recall for queries specifying title with description*

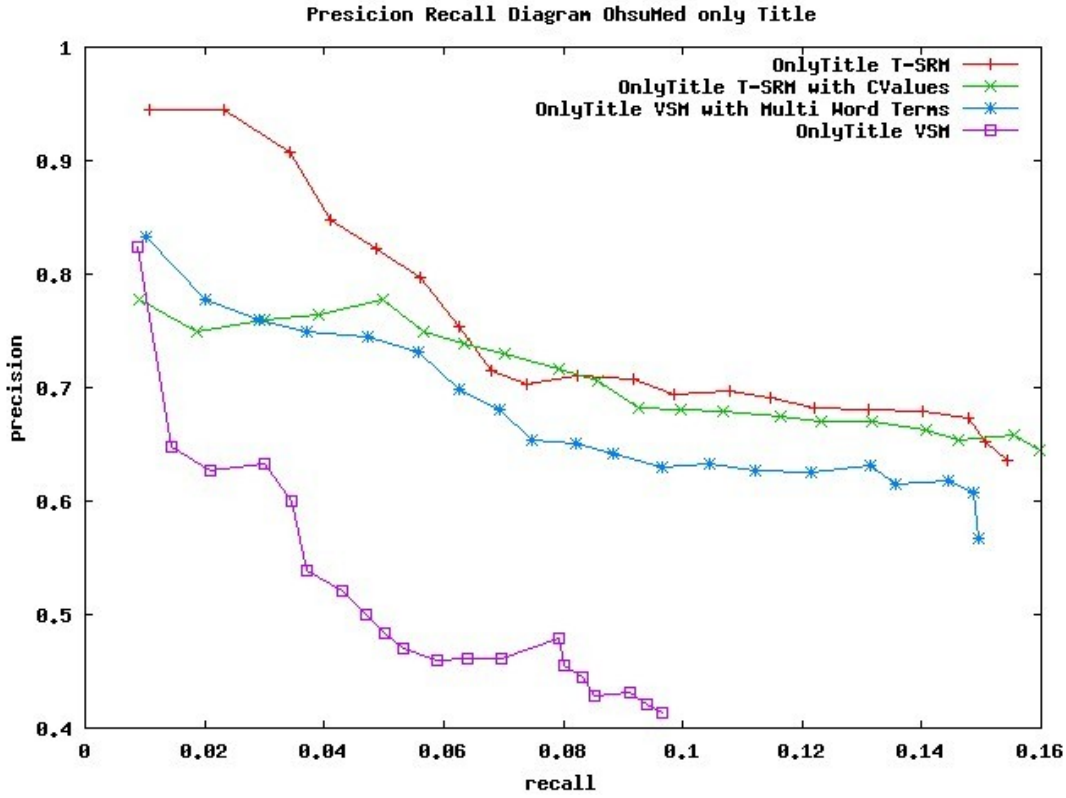


Figure 5.8 *Precision-Recall for queries specifying title only*

Chapter 6. *Conclusions & Future Work*

The contribution of this thesis is twofold: Firstly we showed that it is possible to apply automatic term extraction methods for extracting meaningful representations from documents. Secondly, we showed that is is possible to use these representations for enhancing the performance of retrieval in large document collections.
Our proposed method relies on applying the C/NC-Value term extraction method [15] in conjunction with combined lexical and contextual similarity measures for measuring the similarity between multi-word terms [21]. The experimental results indicate that it is possible to approximate algorithmically the human notion of similarity, reaching correlation up to 73%. Based on this observation we demonstrated that it is possible to exploit this information for improving the performance of information retrieval in applications. We introduce the *Term-based Similarity Retrieval Model (T-SRM). T-SRM* is a novel document similarity model capable of computing similarities when documents contain similar phrases (multi-word terms in general), namely containing not just similar single word terms as it is typical on IR applications and classic retrieval models (like the Vector Space Model). The experimental results indicate that *T-SRM* outperforms the *Vector Space Model (VSM)* for queries on OhsuMed (a standard TREC collection).

As described in chapter 3, the lexical similarity measure relies on the fact that relative multi-word terms often share constituents (heads and modifiers). It will be interesting in the future to explore the semantic relationship between these shared constituents with the integration of *semantic* information from taxonomic ontologies (e.g. WordNet). We will to try make the similarity methods work even better and consequently increase the accuracy of *T-SRM* model retrieval.
Furthermore, additional methods should be used and benchmarked for retrieval, such as the method by Mihalcea et al. [52].

Bibliography – References :

1. Sophia Ananiadou. Towards a Methodology for Automatic Term Recognition. PhD thesis, University of Manchester Institute of Science and Technology, 1988.
2. Sophia Ananiadou. A methodology for automatic term recognition. *In Proceedings of the 15th International Conference on Computational Linguistics, COLING'94,* pages 1034-1038, 1994.
3. Didier Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases.
   In Proceedings of the 14th International Conference on Computational Lingustics, COLING'92, pages 977{981, 1992.
4. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. Modern Information Retrieval. Addison Wesley Longman, 1999.
5. Sager,J.C., Dungworth D.,McDonald P.F : English Special Languages : principles and practice in science and technology. *Oscar Brandsletter Verlag KG* , Wiesbaden , 1980
6. Maynard D, Ananiadou S. 2000b "TRUCKS: a model for automatic multo-word term recognition" *Journal of Natural Language Processing* Vol.8, No. 1, 101-125
7. Kyo Kageura and Bin Umino. Methods of automatic term recognition -a review-. Terminology, 3(2):259-289,1996.
8. Salton G. : Introduction to modern information retrieval.
   Computer Science. McGraw-Hill , 1983
9. Justeson, J.S., Katz,S.M : Technical terminology : some linguistic properties and an algorithm for identification in text.
   Natural Language Engineering 1(1):9-27,1995
10. B.Daille, E.Gaussier, J.M.Lange . Towards automatic extraction of monolingual and bilingual terminology. *In proceedings of Coling 94,* pages 515-521, 1994
11. C.Enguehard and Pantera. Automatic manual acquisition of terminology. *In Journal of Quantitative Linguistics ,* 2(1):27-32 , 1994
12. Dagan I and Itai.A. Word sense disambiguation using a second language monoligual corpus. *Computational Linguistics*, 20:253-596, 1994

13. E.Milios, Y.Zhang, B.He, L.Dong: Automatic term extraction and document similarity in special text corpora. *Proceedings of the 6th conference of the Pacific Association for Computational Linguistics (PACLing'03),* Halifax,Nova Scotia, Canada , August 22-25, 2003 , pages 275-284
14. Frank E., Paynter G.W., Witten I.H., Gutwin C. and Nevill-Manning C.G. (1999) "Domain specific keyphrase extraction" *Proc. Sixteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 668-673.
15. Frantzi, K., Ananiadou, S. & Mima, H. (2000) Automatic recognition of multi-word terms. *International Journal of Digital Libraries 3(2),* Special issue edited by Nikolau, C. & Stephanidis, C. (eds.), 117–132
16. Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz: Building a Large Annotated Corpus of English: *The Penn Treebank, in Computational Linguistic*s, Volume 19, Number 2 (June 1993), pp. 313--330 (Special Issue on Using Large Corpora).
17. Juan C. Sager. A Practical Course in Terminology Processing *John Benjamins Publishing Company*, 1990.
18. Dagan I. , Church K. : Termight: Identifying and translating technical terminology. In: Proc. *7th Conference of the european Chapter of the association for Computational Linguistics,* 1995. EACL'95 , pp 34-40
19. Grefenstette G. :Explorations in Automatic Thesaurus Discovery. *Kluwer Academic Publishers,* 1994
20. Juan C. Sager. Commentary by Prof. Juan Carlos Sager.In Guy Rondeau, editor, Actes Table Ronde sur les Problemes du Decoupage du Terms, Montreal, 26 aouout 1978, pages 39-74, Quebec, 1978. AILA-Comterm,Office de la Langue Francaise.
21. Nenadic, G., Spasic, I. , Ananiadou, S. (2004) Automatic Discovery of Term Similarities Using Pattern Mining, in *International Journal ofTerminology*.10:1, 55-80
22. Bourigault, D. and C. Jacquemin. 1999. "Term extraction + term clustering: an integrated platform for computer-aided terminology." *In Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway, 15-22
23. Maynard D. and S. Ananiadou, 2000a. "Identifying terms by their family and friends." *in Proceedings of COLING 2000*, Luxembourg, 530-536.
24. Collins-Thomson, K., Callan, J.: Query Expansion Using Random Walk Models. *In: ACM Conf. on Information and Knowledge Management (CIKM'05)*, Bremen,

Germany (2005) 704–711

25. Voorhees, E.: Query Expansion Using Lexical-Semantic Relations. In: *ACM SIGIR'94,* Dublin, Ireland (1994) 61–69
26. Ramshaw, LA & Marcus, MP, Text Chunking using Transformation-Based Learning , *ACL Third Workshop on Very Large Corpora*, pp. 82-94, 1995.
27. E.M. Voorhees and D.K. Harmann. Overview of the Seventh Text Retrieval Conference (TREC-7). *In NIST Special Publication* 500-242: The Seventh Text Retrieval Conference (TREC-7), pages 1-23, http://trec.nist.gov/pubs/trec7/t7proceedings.html , 1998.
28. Alex J. Smola, Bernhard Scholkopf (1998). *A Tutorial on Support Vector Regression*. NeuroCOLT2 Technical Report Series – NC2-TR-1998-030.
29. S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, *Improvements to SMO Algorithm for SVM Regression*. Technical Report CD-99-16, Control Division Dept of Mechanical and Production Engineering, National University of Singapore.
30. Ross Quinlan (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA.une, 1995
31. "Text Chunking Using Transformation-Based Learning", Lance Ramshaw & Mitchell Marcus,Proceedings of the Third ACL Workshop on Very Large Corpora, MIT, June 1995
32. "Robust regression and outlier detection Peter J. Rousseeuw, Annick M. Leroy. c1987"
33. Frakes, W., Information Retrieval. Data Structures and Algorithms, ed. W. Frakes and R. Baeza-Yates, Prentice Hall, 1992.
34. Weisstein, Eric W. (1999). *Eric Weisstein's World of Mathematics*. An on-line encyclopedia hosted by Wolfram Inc. (http://mathworld.wolfram.com/)
35. Smadja, Frank (1993). Retrieving collocations from text: Xtract. *Computational Linguistics* **19**(1), 143-177.
36. A. Ratnaparkhi, A maximum entropy model for part-of-speech tagging. In Proceedings of Conference on Empirical Methods in Natural Language Processing, University of Pennsylvania, 1996.
37. Foundations of statistical natural language processing . Christopher D. Manning, Hinrich Schutze. Cambridge, Mass. : MIT Press, c1999.
38. Hersh WR, Buckley C, Leone TJ, Hickam DH, OHSUMED: An interactiveretrieval evaluation and new large test collection for research,

Proceedings of the 17th Annual ACM SIGIR Conference, 1994, 192-201.

39. Hersh WR, Hickam DH, Use of a multi-application computer workstation in a clinical setting, Bulletin of the Medical Library Association, 1994, 82: 382-389.
40. Rada Mihalcea, Courtney Corley, Carlo Strapparava, Corpus-based and Knowledge-based Measures of Text Semantic Similarity, to appear in Proceedings of the American Association for Artificial Intelligence (AAAI 2006), Boston, July 2006.
41. G.Salton. The SMART retrieval system – Experiments in Automatic Document Processing. Prentice Hall Inc., Englewood Cliffs, NJ, 1971
42. G.Salton and M.E. Lesh Computer evaluation of indexing and text processing. Journal of the ACM, 15(1):8-36, January 1968
43. S.E Robertson and K.Sparck Jones. Relevance weighting of search terms. Journal of the American Society for Information Sciences,27(3):129-146, 1976
44. Ananiadou, S. and Zervanou, K. (2004) Techniques and problems in Automatic Term Recognition, *GreekTerminology*: *Research and Applications* (in Greek) Kastaniotis editions, Athens .
45. ISO704. Principles and Methods of Terminology. Technical report, Intern. Organization for Standardization (ISO), Geneva, Switzerland, 1986
46. Jacquemin, C. 2001, *Spotting and Discovering Terms through NLP*. Cambridge MA: MIT Press
47. Mima, H., S. Ananiadou and G. Nenadic. 2001. "ATRACT workbench: an automatic term recognition and clustering of terms." In Matousek, V. et al. (eds), Text, Speech and Dialogue - TSD 2001, LNAI 2166, Springer-Verlag, Berlin, 126-133.
48. Hindle, D. 1990. "Noun classification from predicate-argument structures." In Proceedings of 28thAnnual Meeting of the Association for Computational Linguistic ,Pittsburgh, PA, USA, 268 275.
49. Ding, J., D. Berleant, D. Nettleton and E. Wurtele. 2002. "Mining Medline: abstracts, sentences, orphrases?" in Proceedings of Pacific Symposium on Bioinformatics 2002, Hawaii, USA, 326-337.
50. Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing and Management, 24:513—523.
51. Skuce, D. and I. Meyer. 1991. "Terminology and Knowledge Engineering: Exploring a Symbiotic Relationship," in Proceedings of 6th International Workshop on Knowledge Acquisition for Knowledge-Based Systems, Banff, 29.1-29.21
52. Rada Mihalcea, Courtney Corley, Carlo Strapparava, Corpus-based and

Knowledge-based Measures of Text Semantic Similarity, Proceedings of the American Association for Artificial Intelligence (AAAI 2006), Boston, July 2006.