SEARCHING DOCUMENTS WITH SEMANTICALLY RELATED KEYPHRASES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

İBRAHİM AYGÜL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2010

Approval of the thesis:

SEARCHING DOCUMENTS WITH SEMANTICALLY RELATED KEYPHRASES

submitted by **İBRAHİM AYGÜL** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Nihan Kesim Çiçekli
Supervisor, **Computer Engineering Dept., METU** _____

Assoc. Prof. Dr. İlyas Çiçekli
Co-supervisor, **Computer Engineering Dept., Bilkent University** _____

Examining Committee Members:

Assoc. Prof. Dr. Ferda Nur Alpaslan
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Dept., METU _____

Asst. Prof. Dr. Tolga Can
Computer Engineering Dept., METU _____

Dr. Ruken Çakıcı
Computer Engineering Dept., METU _____

Turgay Yılmaz
HAVELSAN A.Ş. _____

Date: **15.12.2010**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: İBRAHİM AYGÜL

Signature :

ABSTRACT

SEARCHING DOCUMENTS WITH SEMANTICALLY RELATED KEYPHRASES

Aygül, İbrahim
M.Sc., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Nihan Kesim Çiçekli
Co-Supervisor : Assoc. Prof. Dr. İlyas Çiçekli
December 2010, 68 pages

In this thesis, we developed SemKPSearch which is a tool for searching documents by the keyphrases that are semantically related with the given query phrase. By relating the keyphrases semantically, we aim to provide users an extended search and browsing capability over a document collection and to increase the number of related results returned for a keyphrase query. Keyphrases provide a brief summary of the content of documents. They can be either author assigned or automatically extracted from the documents. SemKPSearch uses SemKPIndexes which are generated with the keyphrases of the documents. SemKPIndex is a keyphrase index extended with a *keyphrase to keyphrase index* which stores the semantic relation score between the keyphrases in the document collection. Semantic relation score between keyphrases is calculated using a metric which considers the similarity score between words of the keyphrases. The semantic similarity score between two words is determined with the help of two word-to-word semantic similarity metrics, namely the metric of Wu&Palmer and the metric of Li et al. SemKPSearch is evaluated by the human evaluators which are all computer engineers. For the evaluation, in addition to the author assigned keyphrases, the keyphrases automatically extracted by employing the state-of-the-art algorithm KEA are used to create keyphrase indexes.

Keywords: Keyphrase semantic similarity, Keyphrase based index, Searching and browsing documents , Interctive browsing interface, Keyphrase extraction

ÖZ

ANLAMSAL OLARAK İLİŞKİLİ ANAHTAR KELİME ÖBEKLERİYLE DOKÜMAN ARAMA

Aygül, İbrahim
Yüksek Lisans, Bilgisayar Mühendisliği
Tez Yöneticisi : Doç. Dr. Nihan Kesim Çiçekli
Ortak Tez Yöneticisi : Doç. Dr. İlyas Çiçekli

Aralık 2010, 68 sayfa

Bu tezde, arama terimleriyle anlamsal olarak ilişkilendirilmiş anahtar kelime öbeklerini kullanarak arama yapmayı sağlayan SemKPSearch aracını geliştirdik. Dokümanların kelime öbeklerini anlamsal olarak ilişkilendirmekle kullanıcılara döküman kümesi üzerinde genişletilmiş bir arama yeteneği sağlamayı ve elde edilen alakalı sonuçların artırılmasını hedefledik. Anahtar kelime öbekleri dokümanların içeriklerinin kısa bir özetini sunar. Anahtar kelime öbekleri yazarlar tarafından atanmış veya otomatik olarak üretilmiş olabilir. SemKPSearch dokümanların anahtar kelime öbeklerinden oluşturulmuş olan bir SemKPIndex dizini kullanmaktadır. SemKPIndex içerisinde anahtar kelime öbeklerinden anahtar kelime öbeklerine olan anlamsal ilişki skorunun kaydedildiği bir dizin de içerecek şekilde genişletilmiş bir anahtar kelime öbeği dizinidir. Anahtar kelime öbekleri arasındaki anlamsal ilişki skoru, öbekler içindeki kelimeler arasındaki anlamsal benzerlik kullanılarak hesaplanır. İki kelime arasındaki anlamsal ilişki skorunu hesaplamak için ise Wu ve Palmer kelime benzerliği ölçütü ile Li kelime benzerliği ölçütü, iki farklı kelime tabanlı anlamsal benzerlik ölçüsü olarak kullanılmıştır. SemKPSearch bir grup bilgisayar mühendisi tarafından değerlendirilmiştir. Değerlendirme için yazarların tavsiye ettiği anahtar kelime öbeklerinin yanı sıra, kelime öbeği üretimi için başarılı bir algoritma olan KEA ile otomatik olarak çıkarılmış kelime öbekleri de kullanılarak

oluşturulan kelime öbeği dizinleri kullanılmıştır.

Anahtar Kelimeler: Anahtar kelime öbeklerinin anlamsal benzerlikleri, Anahtar kelime öbeği dizini, Döküman arama ve tarama, Etkileşimli tarama arayüzü, Anahtar kelime öbeği çıkarımı

To my dear wife, Filiz...

ACKNOWLEDGMENTS

I would like to express my appreciation and indebtedness to both my supervisor Assoc. Prof. Dr. Nihan Kesim Çiçekli and my co-supervisor Assoc. Prof. Dr. İlyas Çiçekli for their expert guidance, supportive and constructive approach throughout my masters study and their efforts during supervision of the thesis.

I would like to show my gratitude to my thesis jury members Assoc. Prof. Dr. Ferda Nur Alpaslan, Asst. Prof. Dr. Tolga Can, Dr. Ruken Çakıcı, and Turgay Yılmaz for reviewing and evaluating my thesis.

I would also like to thank my supervisors from TÜBİTAK BİLGEM UEKAE - İLTAREN for their understanding and support during my academic studies.

Special thanks to my colleagues Atakan Şimşek, Celal Çığır, Filiz Aygül, Gülşah Karaduman, Hakan Bağcı, H. Kevser Sunercan, Kezban Başıbüyük and Sercan Gök for attending the evaluation process of this study.

I owe many thanks to my family for their love, trust, and support throughout my life.

Finally, my deepest thanks are to my wife, Filiz, for her endless patience, encouragement and invaluable support during this thesis. I cannot forget her benevolence.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVATIONS

SemKPSearch	Semantically related KeyPhrases aided Search
GUI	Graphical User Interface
KEA	Keyphrase Extraction Algorithm
KIP	Keyphrase Identification Program
TF	Term Frequency
IDF	Inverse Document Frequency
ACM	Association for Computing Machinery

CHAPTER 1

INTRODUCTION

The number of documents available electronically has increased and the use of large document collections such as digital libraries has become widespread. Browsing a document collection and finding the documents of interest turns out to be more difficult. The full-text inverted indexes and ranking algorithms cause standard search engines often return a high number of results, and it is an overwhelming process to find whether a collection covers the useful information.

As Gutwin et al. states, full-text indexing has several problems in browsing a collection [12]. First, although users can retrieve documents containing the words of user's query text, they usually use short topic phrases to explore a collection. The second problem stated by Gutwin et al. is the result set. Standard search engines return a list of documents which is too specific for browsing purposes. And lastly with the nature of browsing, the third problem is the query refinement. Standard engines do not support constituting new queries.

For the problems described above, Gutwin et al. propose a search engine "Keyphind" which is especially designed to help browsing document collections [12]. In Keyphind, keyphrases of the documents are used as the base component for the system. Keyphind uses keyphrase indexes in order to allow users to interact with the document collection at the level of topics and subjects. Keyphrases provide a brief description of a document's content and can be viewed as semantic metadata that summarize documents. Keyphrases are widely used in information retrieval systems (e.g. [11, 9, 17, 15, 24]) and other document browsing systems (e.g. [16, 38]). With the help of the keyphrases of documents in the collection, the user can easily guess the coverage of documents and browse the relevant information.

Keyphrases are usually assigned to documents manually. The authors choose appropriate key-

phrases to summarize their documents. However, most of the documents in a digital library may have no author-assigned keyphrases. Manually attaching keyphrases to documents is a difficult and time consuming process. As a result, in order not to get lost in large data collections, extracting keyphrases automatically becomes essential. In order to address this need, the text mining research community proposes keyphrase extraction or topic detection for the documents in the corpus [32]. Turney [37] states that the keyphrase extraction is the "automatic selection of important, topical phrases from the body of a document". There are several systems developed for automatic keyphrase extraction such as Kea [39], KIP [40], KP-Miner [10] and Extractor [37].

For a given document collection several indexes can be generated based on keyphrases of documents: document-to-keyphrase index, keyphrase-to-document index, word-to-keyphrase index etc. Furthermore, by scoring the semantic similarity of keyphrases and keeping them in a keyphrase-to-keyphrase index, the user can reach documents which contain semantically similar keyphrases with his query text and navigate through the related documents.

There are a considerable amount of semantic similarity measures which give a score for the semantic relation between two words (e.g. [41, 22, 31, 27]). In order to calculate text to text semantic similarity score, there are some measures that use word to word similarity measures [34, 29, 28, 26, 8]. Although these measures are designed especially to score semantic similarities of sentences or long texts , they can be adapted to calculate keyphrase-to-keyphrase semantic similarity score.

In this thesis, we present a keyphrase-based search engine SemKPSearch using SemKPIndex which is similar to the Keyphind index, for browsing a document collection. The user interface and user actions of SemKPSearch are quite different from Keyphind. With the help of keyphrase indexes, the user can browse documents which have semantically related keyphrases with the query text. In this work we extend the keyphrase index with a *keyphrase to keyphrase index* which stores the evaluated semantic similarity score between the keyphrases of the documents in a collection. To calculate similarity scores between keyphrases, we use the text semantic similarity measure given in [8] which employs a word-to-word similarity measure to create a text semantic similarity measure. We propose to use this measure by using Wu & Palmer [41] word-to-word semantic similarity metric and another word-to-word semantic similarity metric described by Li *et al.* [25].

To evaluate SemKPSearch, we used a test corpus that is collected by Krapivin et. al. [19]. The corpus has full-text articles and author assigned keyphrases. Also we used KEA [39] to evaluate the system with automatically extracted keyphrases. We created keyphrase indexes both for author assigned and automatically extracted keyphrases. Besides, to compare two semantic similarities mentioned above, keyphrases are indexed with those metrics. To determine retrieval performance of SemKPSearch, we have evaluated SemKPSearch with Google Desktop search tool which uses full-text index. Evaluation is done by volunteer testers. Testers also graded the semantically related keyphrase suggestions. Evaluation results showed that, with the proposed keyphrase-to-keyphrase semantic similarity metric, SemKPSearch suggests valuable and helpful keyphrases that are semantically related with the query of the tester and the document retrieval performance is better than Google Desktop.

The contributions of this thesis can be listed as follows:

- The notion of the keyphrase semantic similarity is studied.
- A text semantic similarity algorithm is adapted to find the semantic similarity between keyphrases.
- Searching and browsing capabilieites of a search engine which uses keyphrase based indexes are extended by relating the keyphrases semantically.
- To improve the performance of the system, two different word semantic similarity measures are compared.
- To make our system flexible, for the document collections that contain documents without keyphrases, we make use of an automatic keyphrase extraction algorithm.

The rest of the thesis is organized as follows: Chapter 2 gives background information about keyphrase extraction and word semantic similarity metrics. Previous works about text semantic similarity measurements and applications of keyphrases are explained in Chapter 3. Chapter 4 describes the SemKPSearch index structure and generation. Chapter 5 gives the implementation details and the usage of the system. In Chapter 6 evaluation methods and experimental results are presented. Chapter 7 concludes the thesis and discusses the future work.

CHAPTER 2

BACKGROUND INFORMATION

This chapter presents some background information about the concepts and the metrics benefited throughout this thesis. Firstly, information about widely used keyphrase extraction algorithms are given. Next, WordNet and some word-to-word semantic similarity metrics which make use of WordNet are decribed beriefly. Finally, the information retrieval evaluation metrics that are used in evaluation of SemKPSearch are explained.

2.1 Keyphrase Extraction

The usage of keyphrases become more important as the number of documents available electronically increases and the use of large document collections such as digital libraries become widespread. Usually, keyphrases are assigned to documents manually. The authors choose appropriate keyphrases to briefly summarize the documents they have written. However, there are large numbers of documents that have no author-assigned keyphrases. Manually attaching keyphrases to documents is a difficult process. As a result, extracting keyphrases automatically becomes essential. There are several techniques for automatic keyphrase extraction such as KIP [40], Kea [39], KP-Miner [10] and Extractor [37]. In the following, we explain the algorithms KEA and KIP which are widely used in the literature.

2.1.1 KEA

KEA is a simple and effective keyphrase extraction algorithm which uses Naïve Bayes machine learning algorithm for training and extraction purposes [39]. KEA's extraction algorithm has two steps: Training and Extraction. In Figure 2.1, the training and extraction pro-

cesses of KEA are explained briefly.

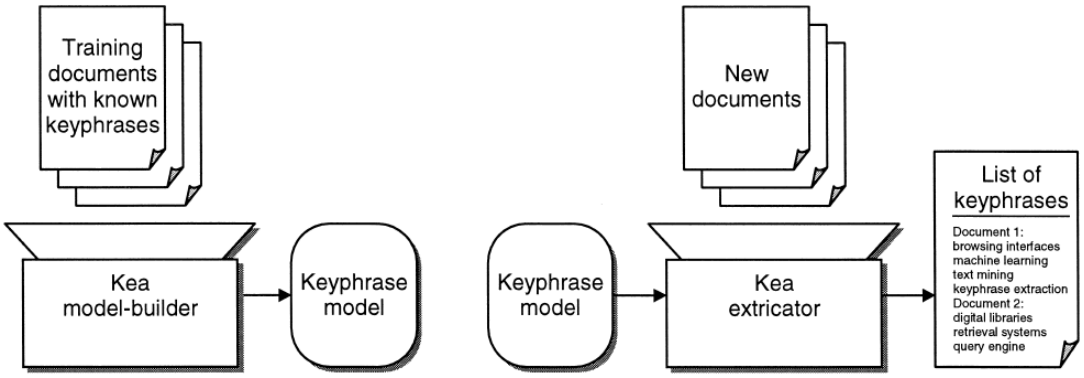

Figure 2.1: KEA keyphrase extraction process

Training Stage:

KEA uses a set of training documents where the author's keyphrases are known. For each training document, candidate phrases are determined by using predefined rules. Then, feature values of each candidate keyphrase are calculated to be used in training and extraction stages. Each phrase is marked as a keyphrase or a non-keyphrase using the author's keyphrases for that document. The machine learning schema generates a model and learns two sets of numeric weights, one for keyphrase examples and one for non-keyphrase examples, from the discretized feature values. There are two features, namely *TFxIDF* and *first occurrence*.

- **TFxIDF:** is a measure of a phrase's frequency in a particular document compared to the frequency of that phrase in general use. Document frequency which indicates the general usage is the number of documents containing the phrase in a large corpus. KEA creates a document frequency file that stores each phrase and the count of the documents in which the phrase appears. The TFxIDF for phrase P in document D is given as [39]:

$$TF \times IDF = \frac{freq(P,D)}{size(D)} \times -\log_2 \frac{df(P)}{N} \tag{2.1}$$

  where

  1. freq(P,D) is the number of times P occurs in D
  2. size(D) is the number of words in D
  3. df(P) is the number of documents containing P in the global corpus

4. N is the size of the global corpus

- **First Occurrence:** is an indicator of how much of the document precedes the phrase's first appearance. It is found by dividing the number of words preceding the phrase's first appearance to the number of words in the document.

Extraction Stage:

To extract keyphrases from a new document, KEA identifies candidate phrases and calculates the feature values of these phrases. Then, the model built during training stage is used to construct a ranked list which contains probabilities of each candidate phrase being a keyphrase. After a post-processing step, the first *n* phrases are returned, where *n* is the number of keyphrases requested.

Evaluation:

The performance of KEA is evaluated by counting the keyphrases that are also chosen by the author of the document. The results of the evaluation show that KEA finds one or two of the five keyphrases assigned by the author. This can be assessed as a good performance because KEA chooses the keyphrases among thousands of candidates.

2.1.2 KIP

KIP [40] is a keyphrase identification program that considers the composition of noun phrases to identify keyphrases. Domain-specific databases which contain expert keyphrases and keywords are used in KIP. It learns how to extract new keyphrases by using these databases.

KIP follows part of speech (POS) tagging, noun phrase extraction and keyphrase extraction steps in order to find out keyphrases of a document. Firstly the documents are loaded into the system and each document is separated into its tokens. Then each word is assigned an initial POS tag by using the WordNet [6] lexical database. A noun phrase is a sequence of two or more words that usually contains useful information. In the noun phrase extraction step, noun phrases are found by selecting the predefined POS tag sequence that are of interest. In the keyphrase extraction step, the noun phrases found in the previous step are scored and ranked by using the domain-specific databases. The keyphrases of the document are found from the ranked noun phrase list.

Evaluation:

Two approaches are heavily used in evaluating automatically extracted keyphrases. In the first approach, standard information retrieval measures, precision and recall, are used. In the second one, domain experts evaluate the extracted keyphrases. In the evaluation of KIP, both methods are utilized. According to the evaluation results of the first method, KIP's recall is 0.70 and precision is 0.44 when the number of extracted keyphrases is 10. The evaluation results of domain experts also show that 94% of the extracted keyphrases are acceptable. The scores of KIP system reveal that it is an effective keyphrase extraction algorithm. Moreover, its ability to learn domain-specific keywords makes it easily applicable to new domains [40].

2.2 WordNet Ontology

WordNet is an electronic lexical database of English that is developed at the Princeton University [6]. WordNet can be seen as a large semantic network which groups English words into sets of synonyms and constructs semantic relations between these synonym sets. Every node in the network represents a real world concept and consists of a set of words that represents this concept. Consequently, each node is a synonym set called *synset*. For example, *car* is a real world concept represented with the synonym set car, auto, automobile, motorcar.

In WordNet, the nodes in the network are connected via links where each link represents a relationship between synsets. The list of relations taken from [30] is given in Table 2.1.

The synsets in WordNet are divided into four categories: nouns, verbs, adverbs, and adjectives. To measure the semantic relatedness, the *Hyponym* relation, i.e. *is-a kind of* or simply *is-a*, is used. This relationship is restricted to nouns and to verbs only. By using the *is-a* relation, the noun and verb synsets are organized into large hierarchies or trees. Each tree has a single root node and the more general concepts are the ancestors of the more specific ones. In Figure 2.2 [30], an example *is-a* hierarchy in WordNet is given. As it can be seen from the figure, the root node of the tree, *entity*, is the most general concept and subsumes the more specific concepts such as *carrot*, *birdcage*, *skin*, etc.

In WordNet, there are 9 hierarchies for nouns and 628 hierarchies for verbs. Verb hierarchies are much shorter and broader with respect to noun hierarchies. This fact results in the higher

Table 2.1: Relations between synsets defined in WordNet.

Relation	**Description**	**Example**
Hypernym	is a generalization of	*furniture* is a hypernym of *chair*
Hyponym	is a kind of	*chair* is a hyponym of *furniture*
Troponym	is a way to	*amble* is a troponym of *walk*
Meronym	is part / substance / member of	*wheel* is a (part) meronym of a *bicycle*
Holonym	contains part	*bicycle* is a holonym of a *wheel*
Antonym	opposite of	*ascend* is an antonym of *descend*
Attribute	attribute of	*heavy* is an attribute of *weight*
Entailment	entails	*ploughing* entails *digging*
Cause	cause to	to *offend* causes to *resent*
Also see	related verb	to *lodge* is related to *reside*
Similar	to similar	to *kill* is similar to *assassinated*
Participle of	is participle of	*stored* (adj) is the participle of "to *store*"
Pertainym of	pertains to	*radial* pertains to *radius*

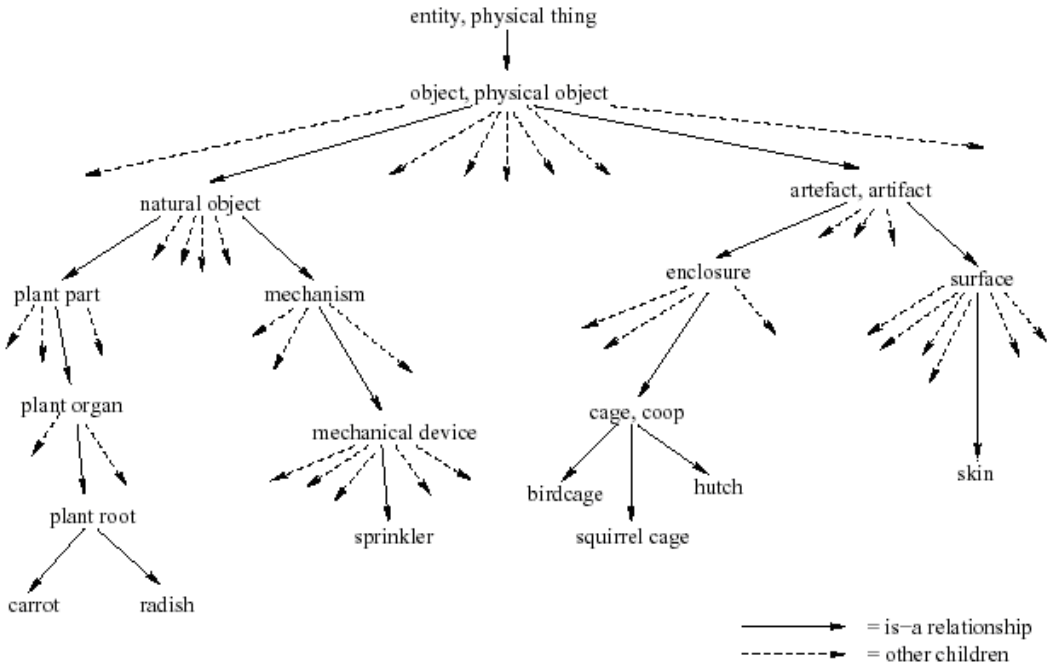

Figure 2.2: An example is-a hierarchy in WordNet

number of verb hierarchies.

There are several application programming interfaces (API) for using WordNet in different environments. By using these API's, one can perform searches, retrievals and even morhpological analysis of words without dealing with complex structures [30]. We used WordNet 3.0 and WordNet.NET [35] library throughout the thesis. Detailed information about the WordNet.NET library is given in Section 5.1.

2.3 Semantic Similarity of Words

There is a large number of measures developed for finding the semantic similarity between words. These measures are successfully applied to natural language tasks such as word sense disambiguation, synonym identification, text summarization, text annotation, information extraction and information retrieval. The lexical database WordNet is heavily used in measuring the semantic similarity because it provides is-a hierarchies for nouns and verbs. While some similarity measures [41, 22] use only the structure and the content of WordNet, some of them [31, 27, 14] use statistical data from large corpus besides the structure of WordNet. Below, we explain semantic similarity metrics that work well with the WordNet hierarchy. All these methods accept two concepts as the input and return a similarity score indicating their relatedness.

2.3.1 Wu & Palmer Similarity Metric

In [41], the depth of two concepts and the depth of the least common subsumer (LCS) in the taxonomy are combined to measure conceptual similarity. It uses the WordNet taxonomy. The formula is given as:

$$Sim_{WP}(c_1, c_2) = \frac{2 * depth(LCS)}{depth(c_1) + depth(c_2)} \tag{2.2}$$

where c_1 and c_2 are two concepts in WordNet, LCS is the least common subsumer of these two concepts, and depth is the distance of the concept to the root of the taxonomy.

2.3.2 Leacock & Chodorow Similarity Metric

Leacock & Chodorow [22] suggests a semantic similarity measure that uses WordNet. The measure considers only the *is-a* hierarchies of nouns in WordNet. As a result, this measure is restricted to find the semantic similarity between nouns. The semantic relatedness is calculated by scaling the distance between the concepts with the maximum dept of the taxonomy.

$$Sim_{LC}(c_1, c_2) = -\log \frac{length(c_1, c_2)}{2 * D} \tag{2.3}$$

where c_1 and c_2 are two concepts in WordNet, length is the shortest path between two concepts using node-counting and D is the maximum dept of the taxonomy.

2.3.3 Resnik Similarity Metric

In [31], Resnik introduces the idea of *information content (IC)*. Information content of a concept indicates the specifity or the generality of that concept. For example, a concept occuring in lower levels of a is-a hierarchy like *carrot* will have high information content. However, a more general concept in a is-a hierarchy like *object* will have much lower information content. The formula of IC is given as:

$$IC = -\log P(c) \tag{2.4}$$

and the probability of occurence of a concept is calculated by using its frequency in the corpus. Hence, statistical data from large corpora is needed to estimate the information content. The probabilty is defined as:

$$P(c) = \frac{freq(c)}{freq(root)} \tag{2.5}$$

Resnik [31] formulates the semantic similarity measure as the information content of the *least common subsumer (LCS)* of two concepts.

$$Sim_{Res}(c_1, c_2) = IC(LCS(c_1, c_2)) \tag{2.6}$$

LCS of two concepts is the lowest concept in the is-a hierarchy that subsumes both concepts.

2.3.4 Lin Similarity Metric

A measure based on the Resnik's similarity measure is introduced in [27]. The Resnik's measure is normalized by including the information content of the two concepts.

$$Sim_{Lin}(c_1, c_2) = \frac{2 * IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)} \tag{2.7}$$

If either of $IC(c_1)$ or $IC(c_2)$ is zero, than similarity is defined as zero. It has a lower bound of 0 and an upper bound of 1.

2.4 Information Retrieval Evaluation Metrics

In order to measure the performance of information retrieval systems, several methods are proposed. In this section, some important evaluation metrics are explained. The common feature of these metrics is that they all assume a ground truth notion of relevancy: a document is relevant or non-relevant to a search term, or a document is assigned a degree of relevance.

2.4.1 Precision

Precision is a metric for measuring the accuracy of the results. It is the proportion of the number of relevant documents among the retrieved documents to the number of all retrieved documents. The formula of precision is given as [3]:

$$Precision = \frac{|\{Relevant\ documents\} \cap \{Retrieved\ documents\}|}{|\{Retrieved\ documents\}|} \tag{2.8}$$

Precision cares all retrieved documents. *Precision at n* or $P@n$ is used to evaluate the precision at a given cut-off rank, taking only the topmost results retrieved by the system into account.

2.4.2 Mean Reciprocal Rank (MRR)

Mean reciprocal rank (MRR) is a statistic value that is used to evaluate an information retrieval system. To use this metric, the IR system must return a set of possible results ordered by correctness probability for a query. The reciprocal rank of a query result can be found by taking the multiplicative inverse of the order of first correct answer. For a query data set,

mean reciprocal rank is defined as the average of sum of reciprocal rank for each query in the dataset. The formula of MRR is given as [4]:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{Q} \frac{1}{rank_i} \tag{2.9}$$

where Q is the query dataset, and $rank_i$ is the rank of the first correct answer to the i^th query.

2.4.3 Discounted Cumulative Gain (DCG)

Discounted cumulative gain (DCG) is used to measure the usefulness, or gain, of an information retrieval system. Each result in the query result list must have a graded relevance score. The results with higher relevancy are more useful when appeared on the top positions of a result list. DCG is based on this assumption. As a result, it penalizes the highly relevant documents appearing at the bottom positions of a result set. The formula of DCG for a particular rank position p is given as [1]:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2 (1 + i)} \tag{2.10}$$

where rel_i is the relevance of the i^th result in the result list of a query, i is the order of the document in the result list, and p is the order of the last document included in the DCG calculation.

For different queries, result list can vary in length. Therefore, the DCG alone is not capable of evaluating the system performance. For this reason, the DCG should be normalized across queries. The normalized cumulative gain, *nDCG* is formulated as:

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{2.11}$$

where $IDCG_p$ is the ideal DCG for rank p, that is computed by sorting the documents of a result list by relevance score. To measure the overall performance of an IR system, the average of nDCG values for all queries is calculated.

CHAPTER 3

RELATED WORK

This chapter gives explanations about applications of keyphrases similar to SemKPSearch. Then we give berief information about text semantic similarity metrics which constructs a base for the keyphrase semantic similarity metric proposed in this thesis.

3.1 Applications of Keyphrases

Keyphrases provide a brief description of a document's content and can be viewed as the semantic metadata that summarize documents. They can be used in many applications such as information retrieval [11, 9, 17, 15, 24], document classification and clustering [42, 21, 13, 7], collection browsing [12, 16, 38], and thesaurus construction [18]. Some of the studies are summarized in the following.

In [24], Li *et al.* incorporate the document keyphrases in the search results. Two indexes are used for this purpose: document-to-keyphrase index and keyphrase-to-document index. For each search result, the keyphrases of that document are listed below the result and act as a hyperlink. When the user clicks on a keyphrase, the documents having this keyphrase will be retrieved. By using this solution, the user can predict the document content easily and navigate through the related documents by using the keyphrase information.

Phrasier [17, 15] is an interactive system for querying, browsing and relating documents in a digital library. Phrasier uses KEA to extract the keyphrases of the documents in the collection. There are two types of indexes: document-to-keyphrase index and keyphrase-to-document index. When the user selects a file, it identifies the keyphrases by using these indexes and highlights them in the text. Also, a link anchor is inserted into the keyphrase text which

points to the set of related documents. The keyphrases, frequency of keyphrases in the current text and the number of documents that keyphrase appears in the collection are displayed to the user. The user can retrieve similar documents by selecting the whole text or selecting a set of keyphrases. Phrasier uses keyphrases rather than the full text to determine document similarities. It uses the cosine similarity method to measure the degree of overlap between the sets of keyphrases for documents. Instead, in our work we try to find semantic similarity between the keyphrases and we suggest the related keyphrases to the user to reach relevant documents.

Keyphind [12] is a system developed for querying, browsing and exploring large document collections. For each document in the collection, keyphrases are automatically extracted by using KEA. Then, four indexes are generated: a phrase list, a word-to-phrase index, a phrase-to-document index, and a document-to-phrase index. When the user enters a word or a phrase to search, the keyphrases containing the query terms and the number of documents corresponding to each keyphrase are displayed to the user. By selecting the keyphrase of interest, the user can view the documents for which the keyphrase is assigned and look at the content of these documents. Also, the keyphrases related to these documents, named co-occurrence phrases, are listed in order to provide the user with the related query topics. The user can filter the search results by forming a query from both the keyphrase AND a co-occurring phrase. Figure 3.1 shows the user interface of Keyphind system during a search for "text".

3.2 Measuring Text Semantic Similarity

Semantic similarity measures of text segments are used in a variety of applications such as natural language processing, information retrieval, text classification and text summarization [29]. The classic method of finding the similarity of two text segments is using a lexical matching method and producing a similarity score based on this method. The methods one step further from the classic method use different approaches such as stemming, stop-word removal, part-of-speech tagging and longest subsequence matching. However, these lexical similarity methods cannot always find the semantic similarity of texts. For example, it is obvious that the two text segments "*I have a car*" and "*I own a vehicle*" are semantically similar, but lexical similarity metrics are inadequate to catch the similarity between these

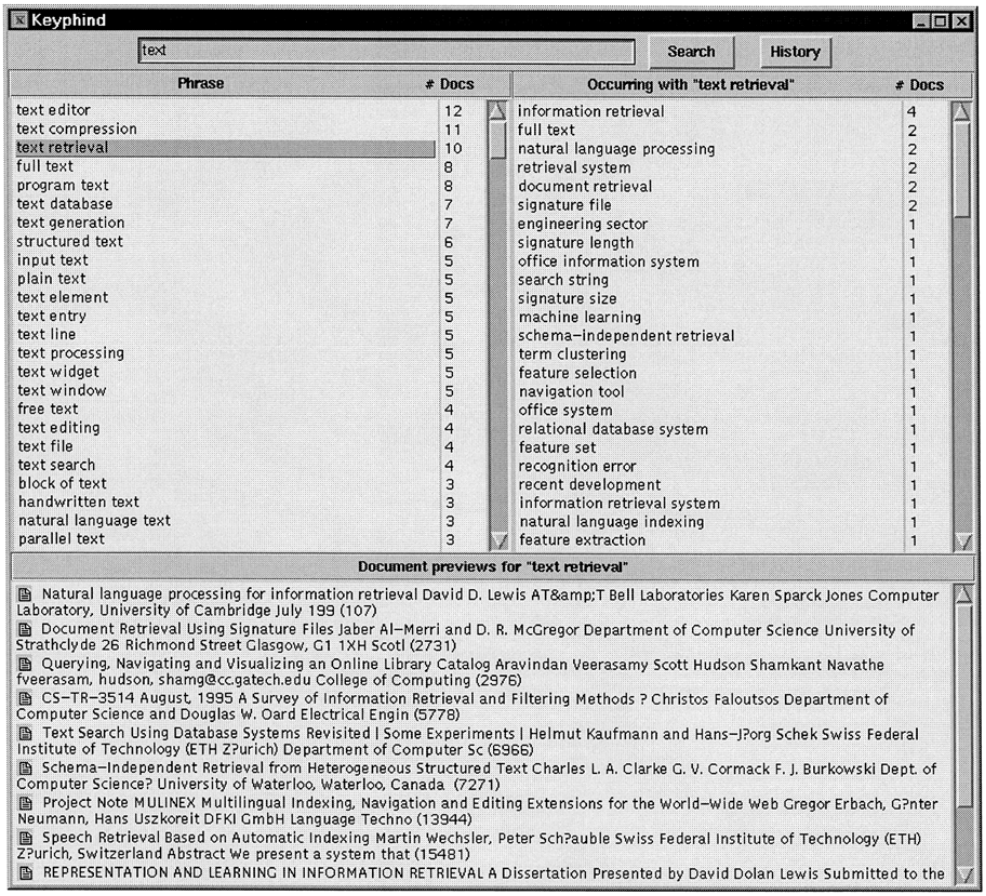

Figure 3.1: Keyphind user interface. Basic query results are seen on upper left panel. On the bottom panel document previews are displayed. On the upper right panel co-occuring phrases with the selected phrase are shown.

texts. There are several word-to-word semantic similarity measures which can be utilized in finding the text semantic similarity. Some of these measures are explained in Section 2.3. In the following sections, two text semantic similarity measures which we draw benefit from in this study are given.

3.2.1 Text Semantic Similarity Measure of Mihalcea and Corley

Mihalcea and Corley [29] suggest a method for calculating the text semantic similarity. They define the semantic similarity of text segments as a function of the semantic similarity of the component words. To calculate the similarity score of texts, they integrate metrics of word-to-word similarity and word specifity. The reason why the word specifiy is included in the similarity measurement is that if a semantic matching is identified between two specific words such as *carrot* and *radish*, then higher weight should be given to this similarity and if the similarity of two general concepts such as *get* and *become* is the issue, then the weight should be lower. To determine the specifity of a word, the *inverse document frequency (IDF)* is used. The formula of IDF is given below [5]:

$$idf_i = \log \frac{|D|}{|d : t_i \in d|} \tag{3.1}$$

where |D| is the total number of documents in the corpus and $|d : t_i \in d|$ is the number of documents where the term t_i appears. If the term does not occur in the corpus, then this will lead to a division-by-zero. As a result, it is common to use $1 + |d : t_i \in d|$ in the denominator.

For two input text segments T_1 and T_2, Mihalcea and Corley suggest the following formula:

$$Sim(T_1, T_2) = \frac{1}{2}\left(\frac{\sum_{w \in T_1} (maxSim(w, T_2) * idf(w))}{\sum_{w \in T_1} idf(w)} + \frac{\sum_{w \in T_2} (maxSim(w, T_1) * idf(w))}{\sum_{w \in T_2} idf(w)}\right) \tag{3.2}$$

According to the formula given above, for each word w in T_1, the word in T_2 that has the highest similarity with w is found by using a word-to-word similarity measure. Then the sum of the word similarities that are weighted with *idf* of corresponding word is calculated. The result of the summation is normalied with the length of T_1. These steps are repeated for text segment T_2. Finally, the semantic similarity of two sentences is measured as the average of the sum of the resulting similarity scores of T_1 and T_2.

To evaluate the success of the proposed text semantic similarity algorithm, they use it to determine if two text segments are paraphrases of each other. They use the Microsoft paraphrase corpus. For each candidate paraphrase pair in the test set, the semantic similarity is calculated by using the formula 3.2. If the candidate pair exceeds a threshold value of 0.5, then it is identified as paraphrase pair. They evaluate the accuracy of the results in terms of the number of the correctly identified paraphrase and non-paraphrase pairs in the test data set. They calculate precision, recall, and F-measure. Also, Mihalcea and Corley evaluate their text similarity measure by using different word-to-word similarity measures while computing the $maxSim(w, T_2)$ and $maxSim(w, T_1)$. They use six knowledge-based measures: Wu & Palmer [41], Lin [27], Resnik [31], Lesk [23], Leacock & Chodorow [22], and Jiang & Conrath [14] similarity metrics. Furthermore, two corpus-based methods are used in evaluation: Pointwise Mutual Information (PMI-IR) [36], and Latent Semantic Analysis (LSA) [20]. Among these similarity measures, the PMI-IR measure gives the best results with an F-measure of 81.0%. By combining all of the metrics mentioned above and simply taking average of them, an F-measure of 81.3% is reached.

In this thesis, we try to adapt text semantic similarity metric given in Formula 3.2 to measure keyphrase similarity. We use the formula with two word-to-word similarity metrics: One with Wu&Palmer and the other with the word similarity measure of Li et al. described in Section 3.2.2.

3.2.2 Sentence Similarity Measure of Li et al.

Li *et al.* [25] propose a method for measuring the semantic similarity between sentences. They calculate the sentence similarity by combining word semantic similarity and word order similarity of sentences. Both of these similarities use the following word semantic similarity metric:

$$s(w_1, w_2) = f_1(l) * f_2(h) \tag{3.3}$$

For a semantic net hierarchy like WordNet, this word similarity measure defines the semantic similarity between two words, w_1 and w_2, as a function of l and h, where l is the shortest path length between w_1 and w_2 and h is the depth of the least common subsumer of w_1 and w_2 in the semantic net. While the path length between two words is increasing, the similarity between these words should decrease. As a result, Li *et al.* define $f_1(l)$ as a monotonically

decreasing function of l. On the other hand, words at upper levels of a hierarchy are more general and less similar according to the words at lower levels. A LCS in deeper levels should effect to a more valuable score. Hence, $f_2(h)$ should be a monotonically increasing function with respect to depth h. With these restrictions the following formula is proposed:

$$Sim_{Li}(w_1, w_2) = e^{-\alpha l} . \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{3.4}$$

where $\alpha \in [0, 1], \beta \in (0, 1]$ are constants to scale the effect of path length and depth.

Given two sentences T_1 and T_2. For each sentence, a semantic vector is constructed by using lexical information and the information content derived from a corpus. The cosine similarity between two semantic vectors gives the semantic similarity S_s of two sentences.

While measuring semantic similarity between two sentences, effect of the word order of sentences should also be taken into account. A word order vector for each sentence is created and used in the calculation of word order similarity. Word order similarity S_r is an efficient metric that measures how similar the word order in two sentences is. Finally, the overall similarity is defined as:

$$S(T_1, T_2) = \delta S_s + (1 - \delta) S_r \tag{3.5}$$

where $\delta \in (0.5, 1]$ is used to adjust the contributions of semantic and word order similarity to the sentence similarity.

Since sentence similarity is out of the scope of this thesis, we prefer to give only a brief summary of the method. More information can be found in [25].

In this thesis, we used the word-to-word similarity of Li's method. We did not use the overall sentence similarity algorithm because the word order similarity does not make sense in the context of keyphrases.

CHAPTER 4

SEARCHING WITH SEMANTICALLY RELATED KEYPHRASES

In this thesis, a search and browsing interface "SemKPSearch" is developed for querying documents in a digital library using their keyphrases. A keyphrased based index, SemKPIndex, is created for a document collection and SemKPSearch uses SemKPIndex for querying and browsing collection in a user friendly interface. In SemKPSearch, browsing also aided by suggesting keyphrases that are semantically related with the given query. As the documents in the collection are indexed by their keyphrases, semantically related keyphrases are indexed with a score which is calculated by employing a semantic similarity metric. In this work, we propose two semantic similarity metrics to calculate a semantic similariy score between keyphrases.

This chapter introduces the proposed keyphrase based search and browsing system. In Section 4.1 an overall description of the system is given. The structure of SemKPSearch index and index generation process are explained in detail in Section 4.2 and Section 4.3. Then Section 4.4 concludes the chapter by explaining methods of finding semantic similarity between keyphrases.

4.1 Overall Description of SemKPSearch System

The overall diagram of SemKPSearch system is shown in Figure 4.1. The steps of preparing the index and querying with keyphrases over a document collection are as follows:

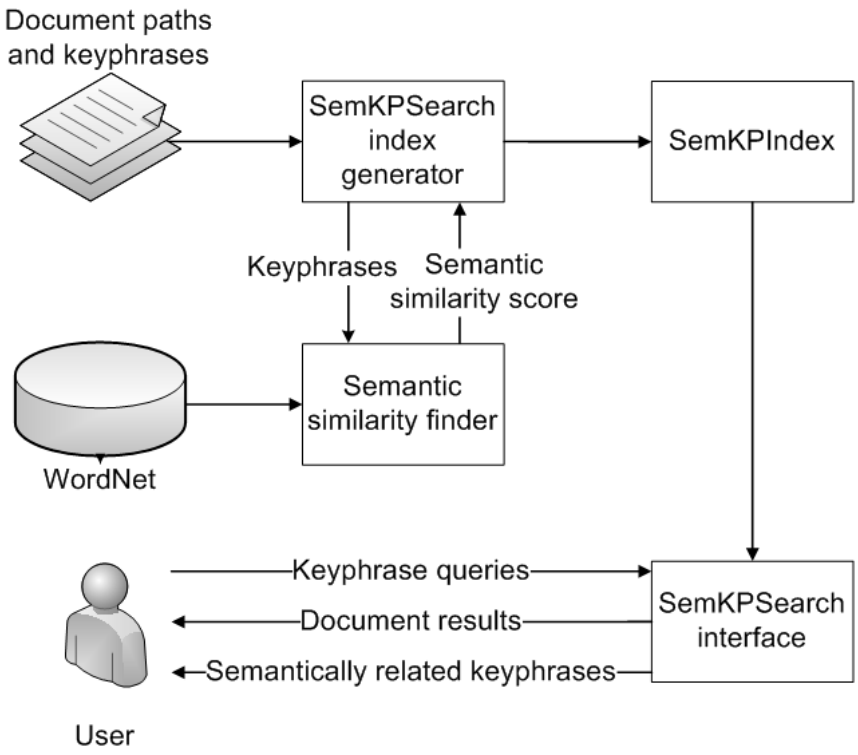

Figure 4.1: Overall diagram of SemKPSearch system

- A document collection with their keyphrases is the main input to SemKPSearch. If the documents in the collection do not have author assigned keyphrases, KEA [39] is employed to extract keyphrases. Automatic keyphrases extraction with KEA is explained in Section 2.1.1.

- Keyphrases, words of keyphrases, paths and titles of documents are indexed together in SemKPIndex.

- Each indexed keyphrase is compared to all other keyphrases and a similarity score is calculated with a semantic similarity metric, and then semantically related keyphrases are stored in SemKPIndex.

- SemKPIndex is written on the disk, so the users can use it later to query the document collection.

- Using SemKPIndex on the SemKPSearch interface, the users query the document collection with topic-like keyphrases. The interface returns a set of document results that contains query term among their keyphrases.

- Besides the documents that contain query term in their keyphrases, the SemKPSearch suggests semantically related keyphrases using SemKPIndex. The users can expand search results by using these suggested keyphrases.

The rest of the chapter explains the structure and generation of the SemKPIndex and the proposed semantic similarity metrics for keyphrases in detail. The implementation and usage details of the SemKPSearch interface are explained in Chapter 5.

4.2 SemKPIndex Structure

SemKPIndex is composed of five indexes; namely a *keyphrase list*, a *word to keyphrase index*, a *document to keyphrase index*, a *keyphrase to document index* and finally a *keyphrase to keyphrase index*. The structure of SemKPIndex is illustrated in Figure 4.2. In the remaining of this section the structures of these indexes are discussed. The generation of the indexes is explained in Section 4.3.

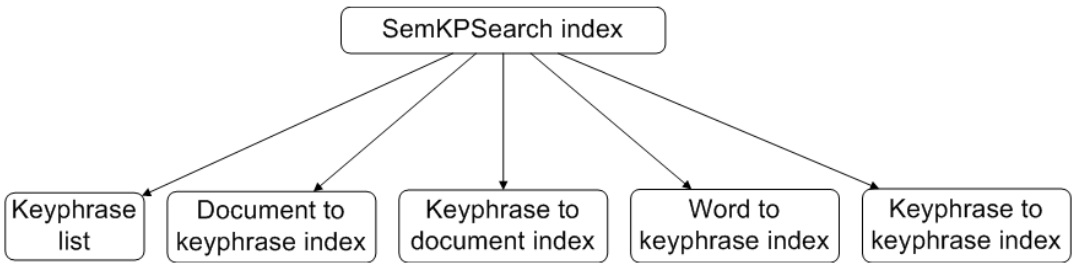

Figure 4.2: The structure of SemKPIndex.

SemKPSearch index structure is very similar to the structure of Keyphind index [12]. In our work there is an additional *keyphrase to keyphrase index* which holds semantically related keyphrases. The sub-indexes of SemKPIndex are given as follows:

1. **Keyphrase list** is a list of all keyphrases that are given with the documents in the collection. This list is used as a reference to keyphrases in other indexes. All keyphrases are kept as lowercase and unique. For example keyphrases “Semantic Similarity” and “semantic similarity” will refer to the same keyphrase. However the keyphrases with non-alphanumeric characters are not handled as the same keyphrases; e.g. “first-order

logic" and "first order logic" are kept as two different keyphrases. This index is used as a suggestion list that guides the user with possible keyphrasea as the user enters the query terms.

2. **Document to keyphrase index** contains information for each document in the collection. Each entry in this index stores the document path, title and the keyphrases of the document. Each keyphrase is kept with a relation score that shows the importance of the keyphrase for the owner document. If no relation score is given for keyphrases, it is automatically calculated during index generation (See Section 4.3). An example entry for this index is given below:

```
C:\docs\587060.txt -> { Using redundancies to find errors,
                        { {error detection, 1.0} ,
                          {extensible compilation, 0.5} } }
```

The *Document to keyphrase index* is used to improve the search result by showing each document with its keyphrases and to order the documents in the search result. For instance, if there are two documents containing the queried keyphrase, then the relation scores are taken into account and the documents are ordered accordingly.

3. **Word to keyphrase index** contains all words in all of the keyphrases. Each entry corresponds to the keyphrases containing the entry word. The following example can be an entry for this index:

```
similarity -> { {similarity} ,
                {semantic similarity} ,
                {similarity measurement} ,
                {similarity retrieval} }
```

This index is needed to show the user more results and more keyphrases to extend the search. For example when the user searches "similarity", in addition to the documents that contain the keyphrase "similarity", the documents containing the keyphrases "semantic similarity", "similarity measurement", "similarity retrieval" will be retrived by the help of this index.

4. **Keyphrase to document index** is a mapping from all keyphrases to the paths of the owner documents. It is somehow the inverse of the *document to keyphrase index*. This

index is used to retrieve the documents that have a given keyphrase among its keyphrases.

5. **Keyphrase to keyphrase index** provides the main contribution in the study. The aim of creating this index is to aid the users in their searches by suggesting semantically related keyphrases with the query terms. The index keeps the semantic relation between keyphrases in the *keyphrase list*. During the index generation, a semantic relation score is calculated between each keyphrase in the system (See Section 4.4). The relations that exceed a predefined threshold value are stored in this index. Each entry is a mapping from a keyphrase to its semantically related keyphrase list. The following entry can be an example of this index:

```
face recognition -> { {face recognition algorithm, 0.930} ,
                      {shape recognition, 0.836} ,
                      {identification system, 0.827} ,
                      {process expressions, 0.815} }
```

The *keyphrase to keyphrase index* gives the user a chance to see the semantically related keyphrases with the search terms. It also helps to extend the search results with the suggested semantically related keyphrases.

4.3 Generating SemKPIndex

SemKPSearch expects a collection of documents and their keyphrases in seperate files as inputs to the SemKPIndex generation process. The keyphrases can be assigned by the authors or automatically extracted from the documents. The documents with their keyphrases are indexed one by one during index generation.

For each document, keyphrases of the document are added to the *keyphrase list*. Then by using these keyphrases, other indexes are created.

After adding keyphrases of a document to *keyphrase list*, a new entry is added to *document to keyphrase index*. The path of the document is stored in the entry with the title of the document[1]. The entry corresponds to the keyphrases of the document. Also a relation score

[1] Title is read from the document. In our experiments title of the document was in the first line of the file, so we did not try to search the title in the given document

is kept for the keyphrases. Keyphrase extraction algorithms generally give keyphrases with their relation scores to the document. If some relation score is given with the keyphrases of the document, this information is stored in the index. Otherwise; for example for the author assigned keyphrases, keyphrases are scored during the index generation as described below:

Calculating Keyphrase Relations to Documents:

If there is no relation score with the keyphrases of a document, a relation score is calculated using the following function that uses the order of each keyphrase:

$$f(i) = 1 - \frac{i}{n} \tag{4.1}$$

where i is the zero based order of appearance of the keyphrase in the keyphrases of the document and n is the number of keyphrases associated with the document. This function gives a score distribution in the range (0, 1]. Using this formula we assume the keyphrases are given by the relevance order. For the higher values of n, the last keyphrase in the list will be much less related with the document. For example, there are two documents, one with 4 and the other with 6 keyphrases. Assume that they have the same keyphrase *"supervised learning"* in the last position of their keyphrases. In the search results for *"supervised learning"* query, the document with 4 keyphrases will appear before the document with 6 keyphrases. This makes sense, because as the number of keyphrases increases, the last keyphrase become less related with the document.

After creating *document to keyphrase index* entry, keyphrases of the document are added to the *keyphrase to document index*. Each entry in this index points to a document list. So if a keyphrase was indexed before, just the path of the document is added to the list corresponding to the keyphrase.

Index generation continues with adding each word of keyphrases to the *word to keyphrase index*. Here each word entry in the index, points to a keyphrase list that gives a reference to keyphrases in which the word occurs. When a word of a keyphrase is already indexed, just the keyphrase is added to the list of the word if the list does not contain it.

After the creation of four indexes above, each keyphrase in the *keyphrase list* is compared with the remaining keyphrases. A semantic relation score is calculated for each keyphrase

pair. To reduce the noise data, this score is compared with a preset threshold value[2]. If the score exceeds the threshold, this score is stored in a *keyphrase to keyphrase index* entry which maps a keyphrase to keyphrase list. If keyphrases to be compared are the same keyphrases, no calculation is done. Calculating the semantic similarity between keyphrases with the proposed metric is explained in detail in the Section 4.4.

After processing all keyphrases, each keyphrase list entry in the *keyphrase to keyphrase index* is evaluated. Each list is sorted according to the relation score. Top 30 keyphrases are kept and others are removed from the list. This reduces the size of the index and increases the usability. Actually even with a threshold value, the semantic similarity calculation may give some irrelevant relations between keyphrases. So we consider only top 30 of the related keyphrases in the list to keep in the index. The user can use additional search options to see some more related keyphrases using the search interface. This task and the other querying tasks are explained in Section 5.2.

4.4 Semantic Similarity Between Two Keyphrases

One of the main contributions in this thesis is expanding a keyphrase based index with a *keyphrase to keyphrase index* which holds semantically related keyphrases information. As described earlier, to generate this index, we propose semantic similarity metrics that calculate a semantic similariy score for two keyphrases.

There are several methods to find the semantic similarity between two texts (e.g. [8, 25, 26, 29]). Keyphrases of the documents generally have more than one word but they are not sentences. So the similarity between two keyphrases is based on the similarity of their words. In [8] Corley and Mihalcea introduce a metric that combines the word-to-word similarity metrics into a text-to-text semantic similarity metric (See Section 3.2.1). This metric is rewritten in Formula 4.2 for keyphrase similarity.

$$KPSim(K_1, K_2) = \frac{1}{2}\left(\frac{\sum_{w \in K_1} (MaxSim(w, K_2) * idf(w))}{\sum_{w \in K_1} idf(w)} + \frac{\sum_{w \in K_2} (MaxSim(w, K_1) * idf(w))}{\sum_{w \in K_2} idf(w)}\right) \tag{4.2}$$

[2] This threshold value depends on the semantic similarity metric used. In Chapter 6.3.3 different threshold values are suggested for different metrics by using the evaluation data.

Here K_1 and K_2 are two keyphrases to be compared; $MaxSim(w, K)$ is a function that gives the maximum similarity score between a given word w and words of keyphrase K using a word-to-word semantic similarity measure; and $idf(w)$ is the *inverse document frequency* of the word w.

In our work, we propose to calculate the semantic similarity between keyphrases using Corley and Mihalcea metric in two ways. One with the combination of Wu & Palmer [41] word-to-word similarity metric, Sim_{WP}, given in Formula 2.2 (See Section 2.3.1), and the other with the following word similarity measure, Sim_{Li}, in Formula 4.3, introduced by Li *et al.* [25] (See Section 3.2.2).

$$Sim_{Li}(w_1, w_2) = e^{-\alpha l} . \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{4.3}$$

where w_1 and w_2 are two words to be compared, l is the shortest path length between w_1 and w_2, h is the depth of the least common subsumer of w_1 and w_2 in WordNet. For WordNet, optimal values for α and β constants are reported as 0.2 and 0.6 respectively in [25].

To find the semantic similarity between two keyphrases by using the similarity metrics above, the first step is to create a similarity matrix. All words of one keyphrase are compared to each word of the other keyphrase and for each comparison, a score is found with Sim_{WP} or Sim_{Li}. Keyphrases are short texts that are not even a sentence. So it is not feasible to detect part of speech information from a bunch of words. Besides, keyphrases of documents generally consist of nouns or verbs. Thus, for word comparisons, words are compared as nouns and verbs and whichever is higher, it becomes the similarity score. For example, to begin comparison of two keyphrases; we will create a similarity matrix as given below:

$$M(K_1, K_2) = \begin{bmatrix} Sim(w1_1, w2_1) & \dots & Sim(w1_1, w2_j) & \dots & Sim(w1_1, w2_n) \\ \vdots & \ddots & & & \vdots \\ Sim(w1_i, w2_1) & \dots & Sim(w1_i, w2_j) & \dots & Sim(w1_i, w2_n) \\ \vdots & & & \ddots & \vdots \\ Sim(w1_m, w2_1) & \dots & Sim(w1_m, w2_j) & \dots & Sim(w1_m, w2_n) \end{bmatrix} \tag{4.4}$$

where K_1 and K_2 are two keyphrases with m and n words respectively, $w1_i \in K_1$ and $w2_j \in K_2$.

In the next step, with the help of this similarty matrix, $\sum_{w \in K_1}(MaxSim(w, K_2) * idf(w))$ and $\sum_{w \in K_1} idf(w)$ are calculated using the maximum value in each row and $idf(w)$ value of the corresponding word in K_1. Similarly $\sum_{w \in K_2}(MaxSim(w, K_1) * idf(w))$ and $\sum_{w \in K_2} idf(w)$ are calculated using the maximum value in each column and $idf(w)$ value of the corresponding word in K_2.

Putting altogether in Function 4.2, the semantic similarity score $KPSim(K_1, K_2) \in [0, 1]$ is calculated for the given keyphrases K_1 and K_2.

CHAPTER 5

IMPLEMENTATION DETAILS AND USER INTERFACE

In this chapter, details of SemKPSearch implementation and the usage of the system are presented. Section 5.1 describes the main components that SemKPSearch depends on and Section 5.2 explains the user interfaces and the usage of the system.

5.1 Implementation Details

SemKPSearch is a windows application running on .NET Framework 3.0. It is implemented in Microsoft Visual Studio 2008 with C# programming language. Also, the following libraries are used to implement similarity metrics.

- **WordNet.Net:** The implemented semantic similarity metrics use WordNet taxonomy. To use WordNet in .NET framework, WordNet.Net library is used. WordNet.Net is an open-source .NET Framework library that works for WordNet 3.0 database, developed by Crowe and Simpson [35].

- **WordsMatching:** A library which is created by Dao [34] and distributed with WordNet.Net. It uses WordNet.Net to search given words in WordNet database. The library includes handy data structures that make easy to compare two words. It provides methods to find least common subsumer or path length between words.

- **TF_IDFWeighting:** TF_IDFWeighting is another useful library that is created by Dao [33]. It helps to get term frequency, inverse document frequency and TF-IDF weight for a word in a given corpus. It also provides methods to compute cosine similarity of two documents in the corpus.

Figure 5.1 presents a diagram explaining the dependencies between these libraries.

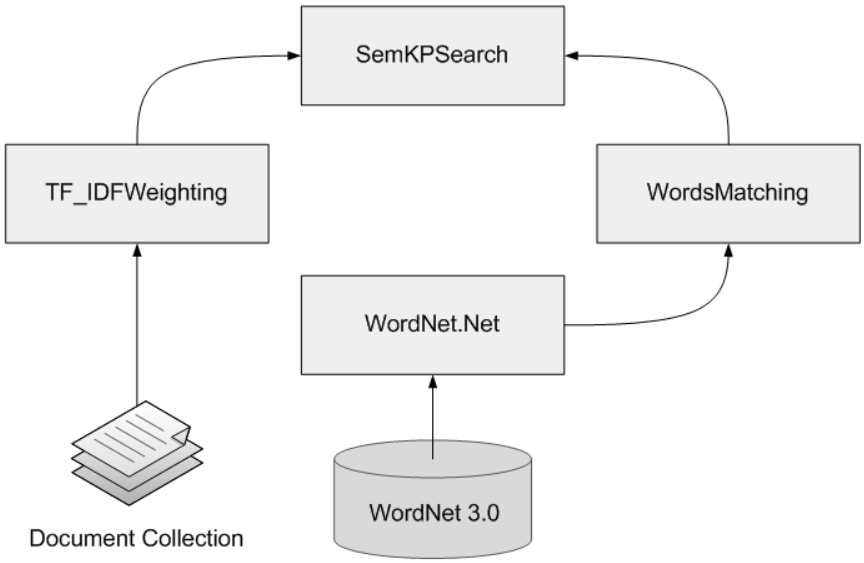

Figure 5.1: The library dependencies in SemKPSearch implementation.

5.2 Using SemKPSearch Interface

The user interface of SemKPSearch enables users to browse documents by querying their keyphrases. SemKPSearch query and browsing interface is shown in Figure 5.2.

SemKPSearch interface provides the following fuctionalities: Suggesting keyphrases stored in the current index as the user is typing the query, caching the queries run by the user, browsing direct results of a keyphrase query, suggesting semantically related keyphrases with the query term, expanding document results and searching incrementally by using semantically related keyphrases. The details of these functionalities are given below:

Keyphrase suggestion as typing:
As the user is writing the query text in the textbox, SemKPSearch suggests the keyphrases that are starting with the same characters of the query text. This ability enables the user to see all keyphrases in the document collection that are indexed in SemKPIndex.

Caching search history:
Every time the user hits the search button, SemKPSearch caches the query in history. After

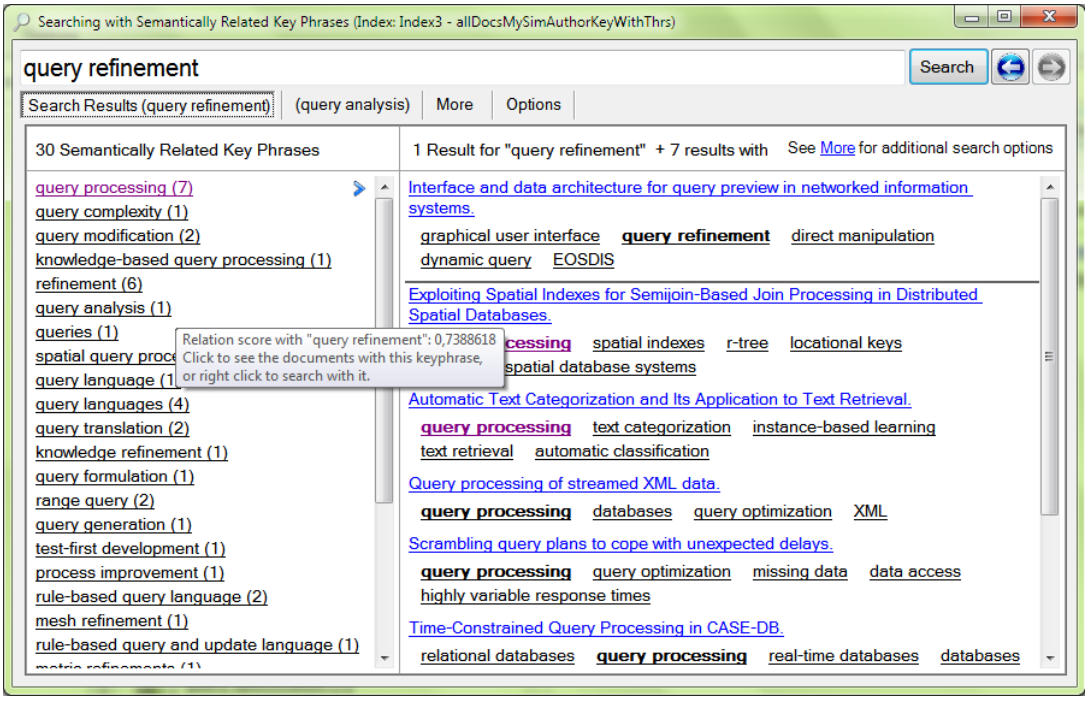

Figure 5.2: SemKPSearch user interface. Keyphrases that are semantically related with the query terms are listed in the left panel of the interface. Search results and their keyphrases are shown on the right panel.

a search, if the user makes a new search, the user can move backward and forward between queries by clicking the “back” and “forward” buttons next to the “Search” button.

Browsing direct results of a keyphrase query:

In SemKPSearch, searching and browsing are based on the keyphrases of the documents instead of their contents. So, when the user enters the query and clicks “Search” button, using *keyphrase to document index* in SemKPIndex, SemKPSearch first returns the documents that have keyphrases matching to the query term. In addition, it uses the *word to keyphrase index* to get keyphrases that contain all of the words of the query term, and then corresponding documents to those keyphrases are shown too. Document results are sorted according to the relation scores of the keyphrases that provide the document to be retrieved. Sorted results are shown in the right panel of the interface as illustrated in Figure 5.2. Documents are shown with their titles and keyphrases which are retrieved from the *document to keyphrase index*. The users can reach to the full text of the document by clicking on the title.

Suggesting semantically related keyphrases:

Together with the search results, keyphrases which are semantically related with the query

term are retrieved from the *keyphrase to keyphrase index* and listed on the left panel as sorted according to the semantic similarity score. If the query text was not indexed in the *keyphrase to keyphrase index* and in the *keyphrase list*, then semantically related keyphrases are calculated on the fly by comparing the query text to all keyphrases. Top 30 of the related keyphrases are stored in the *keyphrase to keyphrase index* and shown to the user. Also by using the *keyphrase to document index*, the number of documents that can be retrieved with a keyphrase is shown on the right of that keyphrase.

Expanding document results:

Using semantically related keyphrases, the user can extend the document results by clicking on the keyphrase with the left mouse button. Documents that can be retrieved by the selected keyphrase are listed below the query results with a seperator bar. Note that, instead of regular search, *word to keyphrase index* is not used, which means the extended results show only the documents containing the selected keyphrase.

Expanding the search using semantically related keyphrases

The user can right click on a semantically related keyphrase or click any keyphrase of a document in the results. Then, on a separate tab a new search with the selected keyphrase is done. In the new tab, the semantically related keyphrases with the selected keyphrase and the corresponding documents are shown. This helps the user to extend the search with a new keyphrase which is semantically related with the query text or which is contained by a document in the result set. If the user wants to come back to the starting point, results for the query text will still be in the first tab. Additional search tabs can be closed from the right click menu on the tabs.

Besides these main abilities SemKPSearch provides more search options on a query. Once a query has been made, the user can look at the “More” tab to see additional search options. As illustrated in Figure 5.3 keyphrases which contain any word in the query text are listed in the left panel. If the user wants to see more keyphrase suggestions that are semantically related with query text, the user can search more in the right panel. This search is done by calculating the semantic similarity between the query text and all keyphrases in the SemKPIndex. The keyphrase similarity metric of the current SemKPIndex is used in this calculation.

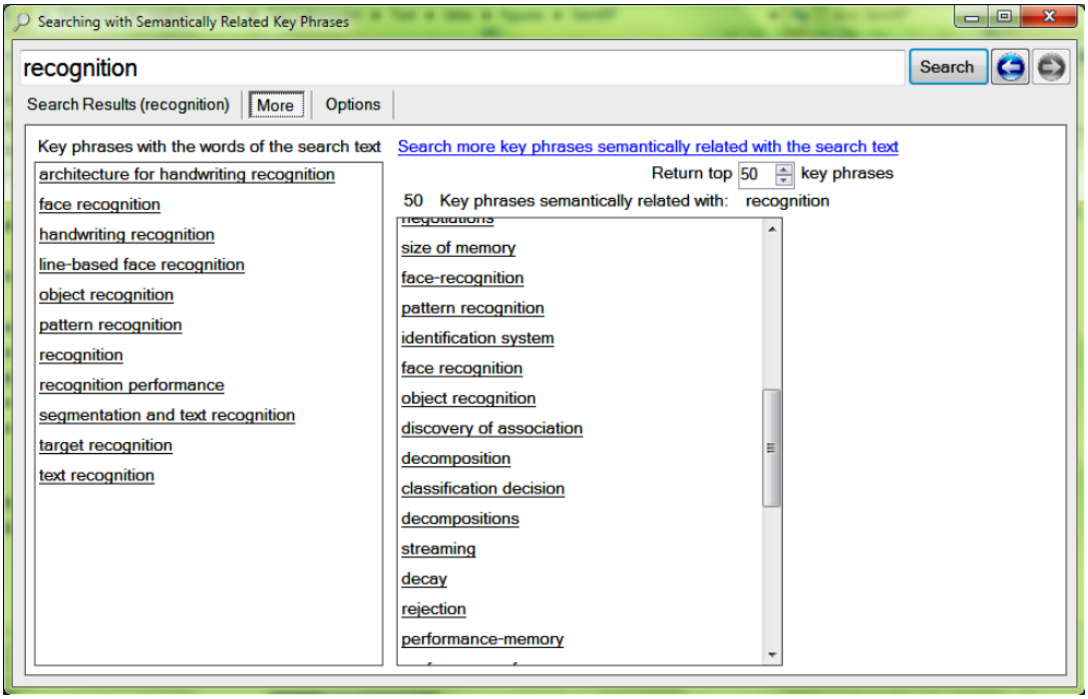

Figure 5.3: Additional search options.

Moreover, using the “Options” tab, a different SemKPIndex file can be loaded or a new SemKPIndex can be generated for a different document collection as long as the documents have their keyphrases.

CHAPTER 6

EVALUATION

6.1 Test Data

In order to evaluate the retrieval performance and the related keyphrase suggestions of SemKP-Search, we used a test corpus that is collected by Krapivin et. al. [19]. The corpus contains 2304 papers from Computer Science domain, which were published by ACM between 2003 and 2005. It has full-text of articles and author assigned keyphrases.

To determine the effect of keyphrase source and similarity metric on keyphrase suggestions, we created four keyphrase indexes. Two of the indexes were created with author assigned keyphrases and the other two were created with KEA extracted keyphrases. Also, two of them used Sim_{WP} and the other two used Sim_{Li} in Formula 4.2 while finding the semantic relation between keyphrases (See Section 4.3 for index generation). In other words, for both author assigned and automatically extracted keyphrases, two indexes were generated with different semantic similarity metrics. The configuraton of these indexes are given below:

- **KEA_SimLi:** This index was created with the keyphrases extracted by KEA. Sim_{Li} was used as the word-to-word similarity metric in Formula 4.2.
- **KEA_SimWP:** This index was created with the keyphrases extracted by KEA by using Sim_{WP} similarity metric.
- **Author_SimLi:** This index was created with the author assigned keyphrases by using Sim_{Li} similarity metric.
- **Author_SimWP:** This index was created with the author assigned keyphrases by using Sim_{WP} similarity metric.

In order to exract keyphrases automatically using KEA, 30 documents were randomly selected from the corpus and their author assigned keyphrases were given to KEA to build the training model (See Section 2.1.1). Then for each document in the corpus, KEA extracted 5 keyphrases which were 2 to 5 words. The reason why we chose keyphrases with at least 2 words is to be able to obtain more precise keyphrases. A one word length keyphrase may be too general.

Besides the SemKPIndexes mentioned above, a full text index over the same corpus was created by Google Desktop. This index was used to compare SemKPSearch with Google Desktop. Google Desktop tool is explained in Section 6.1.1.

6.1.1 Google Desktop

Google Desktop is a desktop search software developed by Google for Linux, Mac OS X, and Microsoft Windows platforms. The software allows you to access the information on your computer such as personal files, music, and photos and information from the web such as user's emails and other Web pages.

When Google Desktop is installed, it creates a full-text index for all files in the computer. Also, the index can be generated for the selected directories only. Google Desktop can index different file types such as text files, pdf and ps files, html files, image and music files, Microsoft Word, Excel, and PowerPoint files [2].

After the index file is created, Google Desktop guarantees that the index stays up to date when new emails are recieved, new web pages are visited, and files are added, deleted, or updated in the local computer. The index file is stored locally on the computer, so it is possible to reach Gmail and web history while being offline.

As soon as indexing is completed, the user can search the files in the computer. Google Desktop interface displays the search results. Each result includes the file name and a brief snippet with the search terms highlighted. The user can display the file content on Google Desktop interface. Also, the search can be restricted to a particular type such as emails, images, videos.

6.2 Evaluation Process

SemKPSearch system evaluation has been done by 8 volunteer testers. Each of them is in computer science area. We gave them an *Evaluation Form* describing the evaluation process and the steps that they should follow while evaluating the system. The *Evaluation Form* is given in Appendix A. We organized the evaluation in two parts. In the following subsections, each part is explained in detail seperately.

6.2.1 Evalution Part 1

In part 1, the evaluators were expected to compare two systems: SemKPSearch and Google Desktop. The aim of this part of the evaluation was to compare a search engine using keyphrase-based index with a search engine using full-text index by measuring the document retrieval performance.

Table 6.1: Keyphrase set used in evaluation part 1.

Keyphrase set 1	**Keyphrase set 2**
clustering algorithm	disk management
information retrieval	recognition
parallel programs	obstacle detection
test cases	sound frequency
data caches	language acquisition
polynomial time approximation	cryptographic algorithms
indirect implications	tree topology
packet routing	file formats
abstract data types	automata theory
description logics	formal languages
data collection	data compression
categorization methods	application development
sensor networks	structured programming
fault detection	complexity analysis
training sets	graph data structure

We prepared two keyphrase sets given in Table 6.1. The difference between the two keyphrase sets is that the keyphrases in *Keyphrase set 1* occurs among the keyphrases of the documents in the collection. However, the keyphrases in *Keyphrase set 2* are not keyphrases of any document. In other words, keyphrases in the first set was indexed in SemKPIndex, the second

set was not. The evaluators selected two keyphrases randomly from each of *Keyphrase set 1* and *Keyphrase set 2*. They searched these four keyphrases by using both SemKPSearch and Google Desktop.

For this part, SemKPSearch was configured with *KEA_SimLi* index. The reason why we chose this index is that it is created from automatically extrated keyphrases. This case is more appropriate for real life applications since most of the documents in a digital library do not have author assigned keyphrases. Also, we believed that the keyphrase semantic similarity metric with SimLi would suggest better results for keyphrase indexing. In Section 6.3, an analysis of evaluation results showed that this assumtion was true.

For both systems, the evaluators judged the relevance of the first 10 documents in the result list. They gave a relevance score between 0 and 4 (namely 0:Irrelevant, 1:Poorly relevant, 2:Partially relevant, 3:Relevant, 4:Completely relevant) to each document, and noted the scores on Table B.1 given in Appendix B.1. During scoring SemKPSearch, if the result set contained less than 10 documents, they expanded the result set by using the suggested keyphrases. Until reaching 10 documents, first they used three suggested keyphrases and evaluated the documents retrieved for these keyphrases. Then up to 9 suggested keyphrases, they used two at a time and scored the retrieved documents.

6.2.2 Evalution Part 2

In part 2, the evaluators were expected to evaluate the keyphrase suggestions of SemKPSearch on different indexes which were created for the same document set. The four indexes used in part 2 evaluation are given in Section 6.1, namely KEA_SimLi, KEA_SimWP, Author_SimLi, and Author_SimWP. One of the aims of evaluation part 2 was to determine which of the keyphrase set, i.e KEA extracted or author assigned, produces more valuable keyphrase suggestions. Another purpose of this evaluation was to find out which of the word similarity metric, the similarity metric of Li et al. or Wu & Palmer, suggests more helpful keyphrases.

For evaluation part 2, we prepared the keyphrase set given in Table 6.2. This is a mixture of keyphrase set 1 and keyphrase set 2 given in Table 6.1. The assessors selected four keyphrases randomly from Table 6.2. They searched each of the keyphrases on four instances of SemKPSearch with the specified indexes and judged the relevance of the first 15 suggested

Table 6.2: Keyphrase set used in evaluation part 2.

Keyphrase set
application development
data caches
polynomial time approximation
disk management
recognition
categorization methods
sound frequency
tree topology
training sets
language acquisition
cryptographic algorithms
automata theory
formal languages
abstract data types
structured programming
complexity analysis
clustering algorithm
sensor networks
fault detection
description logics
parallel programs
test cases
file formats
information retrieval
data compression
graph data structure
obstacle detection
indirect implications
packet routing
data collection

keyphrases for the query term. They gave a relevancy score between 0 and 4 (namely 0:Irrelevant, 1:Poorly relevant, 2:Partially relevant, 3:Relevant, 4:Completely relevant), and noted the scores on Table B.2 given in Appendix B.2. Table B.2 is designed for the results of four keyphrase searches on one index. As a result, in this part, the evaluators filled four tables for four different indexes.

6.3 Analysis of Results

According to the evaluation done with the methodology described in Section 6.2, we collected the data to examine SemKPSearch system. The collected raw data is given in Appendix B. In both of the evaluation parts, each queried keyphrase was evaluated by two evaluators on the average. We weight up the scores gathered from the evaluators in three issues: Keyphrase suggestion success, document retrieval success, and cut-off values for keyphrase similarity metrics.

6.3.1 Keyphrase Suggestion Success

We first discuss the performance of the semantically similar keyphrase suggestion of the system, by calculating the average scores for the first 15 keyphrases given by the evaluators. The results are shown in Table 6.3 and in Figure 6.1. The average scores for the first k keyphrase suggestions where $k \in \{1, 5, 10, 15\}$ is the number of keyphrases to take the average, shows that the Author_SimLi index gets the highest average scores. Also we see that, indexes with SimLi get the same score for the average of 15 evaluated keyphrases.

Table 6.3: Avarage scores calculated for the first k keyphrases.

Index	**Avg@1**	**Avg@5**	**Avg@10**	**Avg@15**
KEA_SimLi	3,34	3,04	2,80	2,48
KEA_SimWP	3,15	2,76	2,22	1,96
Author_SimLi	3,69	3,08	2,81	2,48
Author_SimWP	3,07	2,51	2,10	1,88

Table 6.4 shows the average scores for the keyphrases in the given order. By using these average scores we calculated DCG values for all four indexes. According to the DCG values

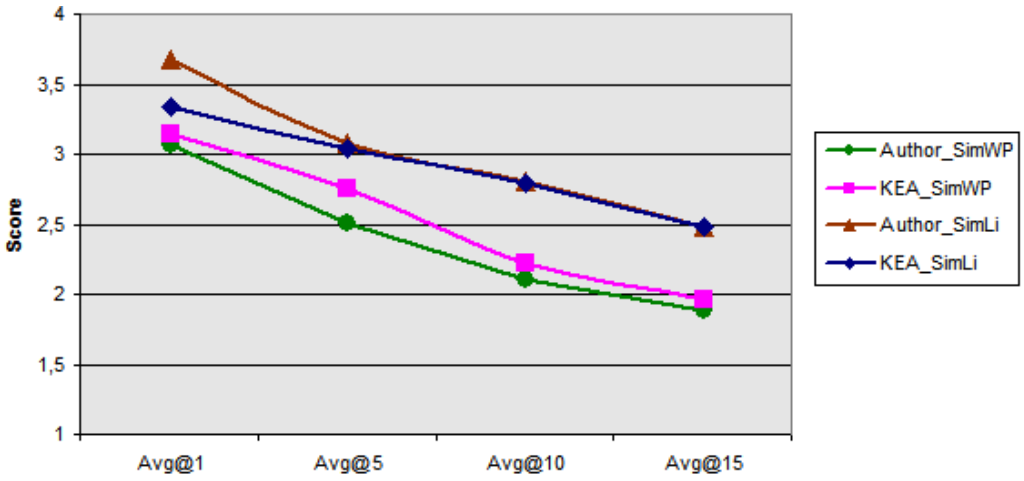

Figure 6.1: Average scores calculated for the first k suggested keyphrases.

in Figure 6.2, Author_SimLi index once again suggests the semantically related keyphrases in the most valuable order among the four indexes. Indexes with SimLi measure measure, suggests keyphrases in more valuable orders. On the other hand, when SimWP measure is used, suggestion with author assigned keyphrases is scored less than the suggestion with KEA assigned keyphrases.

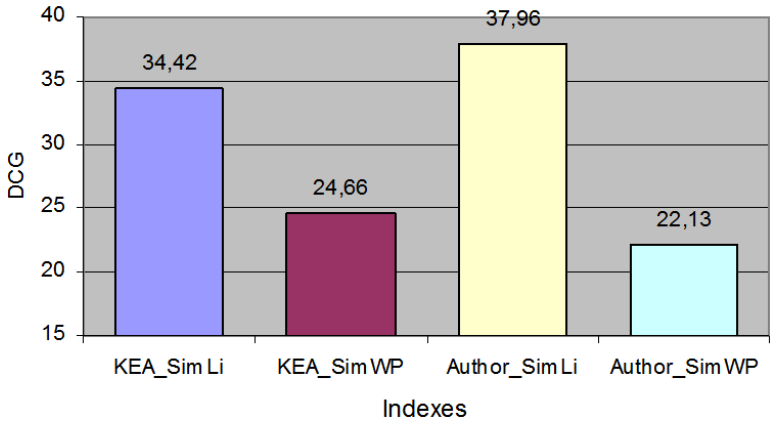

Figure 6.2: DCG values for the suggested keyphrases by using the four SemKPIndexes.

Table 6.4: Average user scores and DCG values for these scores for the first 15 semantically related keyphrase suggestions of four evaluation indexes.

Keyphrase Order	**KEA_SimLi**	**KEA_SimWP**	**Author_SimLi**	**Author_SimWP**
1	3,34	3,15	3,69	3,07
2	3,31	2,81	3,66	2,70
3	2,94	2,73	2,88	2,47
4	2,94	2,54	2,78	1,83
5	2,69	2,58	2,41	2,50
6	2,59	2,27	3,00	2,33
7	2,78	1,96	2,16	1,77
8	2,50	1,92	2,63	1,50
9	2,38	1,19	2,53	1,53
10	2,50	1,04	2,41	1,33
11	2,22	1,35	1,88	1,63
12	1,97	1,27	1,94	1,53
13	1,75	1,50	2,22	1,07
14	1,72	1,62	1,38	1,33
15	1,56	1,46	1,69	1,60
DCG_{15}	34,42	24,66	37,96	22,13

Furthermore, to compare the keyphrase suggestions of the indexes, MRR and precision values were calculated from the assessor scores. Remember we made the evaluators score the keyphrases over 4. So in order to calculate MRR and precision, we mapped the assessor scores to binary data. This mapping is done in two ways:

1. Count the keyphrase as relevant to the query, if its score is 3 or higher; count it as irrelevant otherwise.

2. Count the keyphrase as relevant to the query, if its score is equal to 4; count it as irrelevant otherwise.

According to these mappings MRR, precision for all scored keyphrases and precision for the first 5 suggested keyphrases are shown in Table 6.5.

The values in Table 6.5 tell us that for most of the keyphrase queries Author_SimLi index suggested relevant keyphrases in the earlier orders. KEA_SimLi index follows Author_SimLi index for all examinations. If we compare KEA_SimWP and Author_SimWP, for a more harsh examination by taking score 4 as relevant, Author_SimWP index suggests better than

Table 6.5: MRR, precision and precision@5 values for suggested keyphrases.

Index	MRR_{4-3}	MRR_4	Pre_{4-3}	Pre_4	$P@5_{4-3}$	$P@5_4$
KEA_SimLi	0,88	0,74	0,54	0,31	0,73	0,50
KEA_SimWP	0,85	0,59	0,42	0,22	0,65	0,36
Author_SimLi	0,98	0,80	0,56	0,33	0,76	0,53
Author_SimWP	0,83	0,75	0,34	0,22	0,53	0,39

KEA_SimWP. However if we relax the examination and take the score 3 and above as relevant, KEA_SimWP index suggests better than Author_SimWP index.

According to the results in all three tables, keyphrase suggestion by using keyphrase similarity metric with SimLi, achieves better results than using SimWP. However, using SimLi metric, author assigned keyphrases has higher scores. This is an expected outcome, since author assigned keyphrases may become more meaningful from the automatically extracted kryphrases. Although, Author_SimLi index has better suggestion results, KEA_SimLi index results are competitive with it since both indexes has an average score 2,48 for all evaluated keyphrases in Table 6.3. Considering that in real life applications, since the most of the documents in a collection do not have author assigned keyphrases, we can argue that keyphrase suggestion can be done with automatically extracted keyphrases and the proposed keyphrase semantic similarity metric with SimLi.

6.3.2 Document Retrieval Success

In the first part of the evaluation, SemKPSearch configured with KEA_SimLi index compared to Google Desktop on the same document collection. The document retrieval performance of the two systems were compared with the relation scores for the retrieved documents given by the evaluators. Table 6.6 presents the average scores, MRR values and precision values for both systems. Similarly Figure 6.3 presents the average scores and precision values together. Table 6.6.a shows the evaluation results for the documents returned for a keyphrase query which was indexed by the evaluated SemKPIndex. In other words there was at least one document with the queried keyphrase extracted by KEA in the document collection.

According to Table 6.6.a, the documents retrieved with SemKPSearch get higher average scores than the documents returned by Google Desktop. Since this table is for the evalua-

Table 6.6: Evaluation results to compare document rerieval performance of SemKPSearch and Google Desktop. a) Searching with the keyphrases indexed in SemKPIndex. b) Searching with the phrases not indexed in SemKPIndex.

	SemKPSearch			Google Desktop		
For the first n documents	Average Score	MRR	Precision	Average Score	MRR	Precision
1	3,95	1,00	1,00	3,05	0,70	0,70
3	3,57	1,00	0,83	2,94	0,83	0,67
5	3,32	1,00	0,78	2,74	0,83	0,56
7	3,04	1,00	0,70	2,49	0,83	0,49
10	2,74	1,00	0,62	2,15	0,83	0,40

(a)

.

	SemKPSearch			Google Desktop		
For the first n documents	Average Score	MRR	Precision	Average Score	MRR	Precision
1	2,04	0,43	0,43	2,14	0,29	0,29
3	1,93	0,50	0,33	1,81	0,29	0,25
5	2,01	0,54	0,34	1,86	0,29	0,21
7	1,71	0,54	0,25	1,90	0,31	0,25
10	1,71	0,54	0,21	1,73	0,31	0,22

(b)

tion of the results with the keyphrases indexed in SemKPIndex, one can argue that this is the success of the keyphrase extraction algorithm: direct results in the first orders will get appearently high scores because the search term is directly extracted from the documents as their keyphrases. But with a further analysis of the raw results we see that for all queried keyphrases in the evaluation part one, the number of directly returned documents is 2,4 out of 10 on the average. So, 76% of the evaluated documents are returned by assisting the query with semantically related keyphrases. The average score for the documents that are retrieved by the suggested keyphrases is 2,47. Futhermore, the average score for the first 5 documents that are retrieved by the suggested keyphrases is 2,62. On the other hand, the average score for the last 8 documents out of 10 retrieved by Google Desktop is 1,9. These values reasonably show us that using keyphrases of documents, the document retrieval with SemKPSearch is more successful than Google Desktop.

To calculate MRR and precision values, we counted the documents with a score 3 and above as relevant, and the documents with a score less than 3 is irrelevant. These values on Table 6.6.a are similar to the average scores and SemKPSearch beats Google Desktop. Here we see

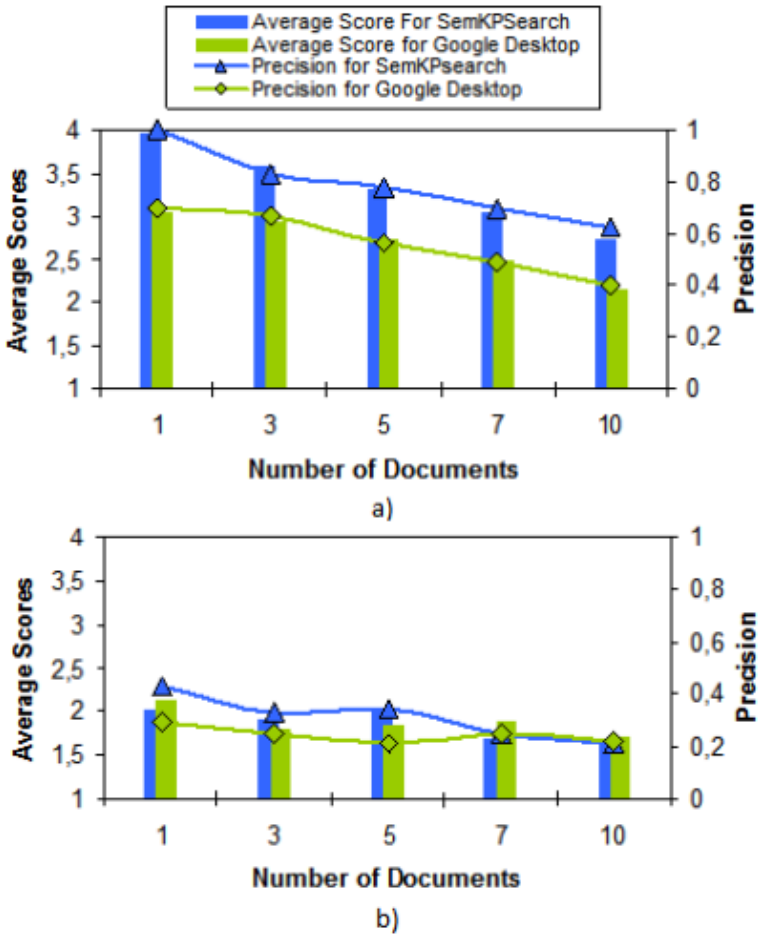

Figure 6.3: Average scores and precision values with respect to number of retrieved documents. a) Searching with indexed keyphrases. b) Searching with non-indexed phrases.

that the MRR value for SemKPSearch is 1, which means that for all queries, SemKPSearch returned a relevant document to the query term at the first place. Actually this result comes from the success of keyphrase extraction KEA. Because the first document has always the query term as its keyphrase extracted by KEA.

In Table 6.6.b, a slightly different result is seen for the documents returned for the phrases not indexed in SemKPIndex. The average scores are a bit lower for the SemKPSearch results. However MRR and precision values shows that for the queries with phrases that are not indexed as a keyphrase of a document, in SemKPSearch, related documents appear on the higher orders and it returns more valuable results up to 7 documents on the average.

By using the scores for each query in evaluation part one, a nDCG value is calculated for the 10 returned documents and average nDCG values are determined for both systems. The results are given in Table 6.7 by grouping again with respect to the query set. According to this table SemKPSearch ranking of the retrieved documents is better than Google Desktop for the queries with indexed keyphrases. However, for the queries with non-indexed phrases, the ranking of the documents returned by Google Desktop is better.

Table 6.7: $nDCG_{10}$ values for overall search results.

	SemKPSearch	Google Desktop
With indexed keyphrases	0,97	0,89
With Non-indexed phrases	0,85	0,93

Consistent with the results mentioned above, the document retrieval performance of SemKPSearch is better than Google Desktop for the queries with the assigned keyphrases of the documents. While the user is typing the query, the user interface of SemKPSearch leads the user to search with indexed keyphrases by showing a suggestion list to the user. This behavior may reduce the deficency of the system on non-indexed phrase queries.

6.3.3 Threshold Values for Keyphrase Similarity Metrics

For the evaluation purposes, we created SemKPIndexes as described in Section 6.1. To generate these indexes we did not use any threshold values. However, to prevent SemKPIndex from being too large, each keyphrase entry is restricted to have a maximum number of 30

semantically related keyphrases in the *keyphrase to keyphrase index*. On the other hand, for a real life application, it is better to use a threshold value for the keyphrase similarity metrics used in SemKPIndex generation. Using a threshold value, useless keyphrase suggestions can be eliminated. For different threshold values, the number of keyphrases that can be suggested and precision values for these suggestions are given in Table 6.8. For precision values, keyphrases with a score 3 and above is counted as relevant. The table is grouped according to the keyphrase semantic similarity metrics of the SemKPIndexes.

Table 6.8: Number of suggested keyphrases and their precision values with respect to the threshold value. a) Number of keyphrases and precision values for the indexes created with SimLi metric. b) Number of keyphrases and precision values for the indexes created with SimWP metric.

	KEA_SimLi		Author_SimLi	
Threshold	# of KP	Precision	# of KP	Precision
0,95	0,75	0,86	0,25	1
0,9	1,59	0,84	1,38	0,93
0,85	3,75	0,83	2,94	0,92
0,8	7,72	0,71	6,75	0,8
0,75	11,38	0,56	10	0,69
0,7	13,28	0,58	13,4	0,6
0,65	14,53	0,55	14,9	0,56

(a)

.

	KEA_SimWP		Author_SimWP	
Threshold	# of KP	Precision	# of KP	Precision
0,95	0,92	0,79	1,1	0,81
0,9	5,42	0,67	5,6	0,64
0,875	7,77	0,59	7,8	0,57
0,85	11,9	0,53	9,93	0,44
0,825	13,7	0,46	11,9	0,39
0,8	14,5	0,43	14,4	0,35
0,775	15	0,42	15	0,34

(b)

According to the information in Table 6.8.a we can advise to use 0,65 threshold value for SemKPIndexes crated with keyphrase semantic similarity metric with SimLi. Because, on the average, the evaluators gave a score 3 and above to more than half of the suggested 15 keyphrases. According to Table 6.8.b, the threshold value 0,85 can be used for the metric with SimWP. Using this threshold value, the number of keyphrases that can be suggested on the average is around 11 and the average precision for these keyphrases is around 50%.

CHAPTER 7

CONCLUSION

In this thesis, we propose SemKPSearch system which is a user friendly search and browsing interface for querying documents in a digital library by their keyphrases. SemKPSearch indexes the documents with their keyphrases in SemKPIndex. SemKPIndex is a keyphrase based index which is constructed from five sub indexes. Namely; the *keyphrase list* for all keyphrases in the index, the *document to keyphrase index* to map the documents to their keyphrases, the *keyphrase to document index* mapping keyphrases to documents, the *word to keyphrase index* to reach the keyphrases by their words, and finally the *keyphrase to keyphrase index* holding the semantic relations between the keyphrases and scores for these relations.

To calculate the semantic similarity between keyphrases, we propose to use a text-to-text semantic similarity metric that is proposed by Corley and Mihalcea [8]. This metric employs a word-to-word semantic similarity measure. By using this metric semantic similarity of the keyphrases is formulated as a function of the similarity of the words of the keyphrases and the specifity of words coming from IDF.

Through the user interface of SemKPSearch, the user can search documents with topic like query phrases. SemKPSearch returns keyphrases that are semantically related to the query text, as well as the documents having keyphrases containing the query text. The user can continue to browse more documents with the suggested semantically related keyphrases or with the keyphrases of the retrieved documents. In this way it is expected that the user can reach the related documents with the query text even if the documents do not contain the query term.

Evaluation of the system is done by the human evaluators. The evaluators judged the quality of the results and the effectiveness of the suggested semantically related keyphrases, by using

the indexes mentioned in Section 6.1. In order to evaluate the performance of retrieving the documents with semantically related keyphrases, SemKPSearch system was compared to Google Desktop which is a full-text index based search engine. The evaluation results showed that the evaluators found the documents retrieved with SemKPSearch more related to the query term than the documents retrieved with Google Desktop. Besides the document retrieval, the semantically related keyphrase suggestions were also evaluated by the assessors. According to the results obtained for related keyphrase suggestions, it is feasible to use the automatically extracted keyphrases and to relate them with the keyphrase semantic similarity that we proposed.

In the future, the semantic similarity measurement between two keyphrases might be extended by calculating the cosine similarity between the document sets corresponding to each of the keyphrases. By improving the semantic similarity measure, false or weak semantic relations in *keyphrase to keyphrase index* might be reduced. Also carrying on an extensive evaluation with a broader document collection and with additional evaluators, might reflect the success of the system more accurately. In such an evaluation, *keyphrase to keyphrase index* might be improved with learning abilities using the false reccomendations marked by the users.

REFERENCES

[1] Discounted cumulative gain.
http://en.wikipedia.org/wiki/discounted_cumulative_gain, last visited on 27 dec. 2010.

[2] Google Desktop - Features.
http://desktop.google.com/features.html, last visited on 07 dec. 2010.

[3] Information retrieval.
http://en.wikipedia.org/wiki/information_retrieval, last visited on 27 dec. 2010.

[4] Mean reciprocal rank.
http://en.wikipedia.org/wiki/mean_reciprocal_rank, last visited on 27 dec. 2010.

[5] tf-idf.
http://en.wikipedia.org/wiki/tf-idf, last visited on 07 dec. 2010.

[6] WordNet - About WordNet. http://wordnet.princeton.edu/, last visited on 24 Aug. 2010.

[7] A. Bernardini, C. Carpineto, and M. D'Amico. Full-Subtopic Retrieval with Keyphrase-Based Search Results Clustering. *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09*, 1, 2009.

[8] C. Corley and R. Mihalcea. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18. Association for Computational Linguistics, 2005.

[9] W.B. Croft, H.R. Turtle, and D.D. Lewis. The use of phrases and structured queries in information retrieval. In *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 32–45. ACM, 1991.

[10] S.R. El-Beltagy. KP-Miner: A Simple System for Effective Keyphrase Extraction. *Innovations in Information Technology, 2006*, pages 1–5, 2006.

[11] J.L. Fagan. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science*, 40(2):115–132, 1989.

[12] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27(1-2):81–104, 1999.

[13] J. Han, T. Kim, and J. Choi. Web document clustering by using automatic keyphrase extraction. In *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, pages 56–59. IEEE Computer Society, 2007.

[14] J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *In Proceedings on International Conference on Research in Computational Linguistics, Taiwan*, 1997.

[15] S. Jones. Design and evaluation of phrasier, an interactive system for linking documents using keyphrases. In *Proceedings of Human-Computer Interaction: INTERACT'99*, pages 483–490, 1999.

[16] S. Jones and G. Paynter. Topic-based browsing within a digital library using keyphrases. In *Proceedings of the fourth ACM conference on Digital libraries*, page 121. ACM, 1999.

[17] S. Jones and M.S. Staveley. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM, 1999.

[18] B. Kosovac, DJ Vanier, and TM Froese. Use of keyphrase extraction software for creation of an AEC/FM thesaurus, 2000.

[19] M. Krapivin, A. Autaeu, and M. Marchese. Large Dataset for Keyphrases Extraction. Technical Report DISI-09-055, DISI, University of Trento, Italy, 2009.

[20] T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2):259–284, 1998.

[21] L.S. Larkey. A patent search and classification system. In *Proceedings of the fourth ACM conference on Digital libraries*, page 187. ACM, 1999.

[22] C. Leacock and M. Chodorow. Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

[23] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *In Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM, 1986.

[24] Q. Li, YB Wu, R.S. Bot, and X. Chen. Incorporating document keyphrases in search results. In *Proceedings of the Americas Conference on Information Systems (AMCIS), New York*, 2004.

[25] Y. Li, Z.A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on knowledge and data engineering*, pages 871–882, 2003.

[26] Y. Li, D. McLean, Z.A. Bandar, J.D. O'Shea, and K. Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, pages 1138–1150, 2006.

[27] D. Lin. An information-theoretic definition of similarity. In *In Proceedings of the 15th International Conference on Machine Learning*, volume 1, pages 296–304, 1998.

[28] Y. Liu, C. Li, P. Zhang, and Z. Xiong. A Query Expansion Algorithm Based on Phrases Semantic Similarity. In *2008 International Symposiums on Information Processing (ISIP)*, pages 31–35, 2008.

[29] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence-Volume 1*, pages 775–780. AAAI Press, 2006.

[30] S. Patwardhan. Incorporating dictionary and corpus information into a context vector measure of semantic relatedness. Master's thesis, University of Minnesota, 2003.

[31] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th international joint conference on Artificial intelligence-Volume 1*, pages 448–453. Morgan Kaufmann Publishers Inc., 1995.

[32] M. Song, I.Y. Song, and X. Hu. KPSpotter: a flexible information gain-based keyphrase extraction system. In *Proceedings of the 5th ACM international workshop on Web information and data management*, page 53. ACM, 2003.

[33] Dao T. Term frequency/inverse document frequency implementation in c#. http://www.codeproject.com/kb/cs/tfidf.aspx, last visited on 07 dec. 2010.

[34] Dao T. and Simpson T. Wordnet-based semantic similarity measurement. http://www.codeproject.com/kb/string/semanticsimilaritywordnet.aspx, last visited on 07 dec. 2010.

[35] Simpson T. and Crowe M. Wordnet.net open source wordnet library for .net. http://opensource.ebswift.com/wordnet.net/, last visited on 07 dec. 2010, 2005.

[36] P. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. *Machine Learning: ECML 2001*, pages 491–502, 2001.

[37] P.D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[38] N. Wacholder, D.K. Evans, and J.L. Klavans. Automatic identification and organization of index terms for interactive browsing. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, page 134. ACM, 2001.

[39] I.H. Witten, G.W. Paynter, E. Frank, C. Gutwin, and C.G. Nevill-Manning. KEA: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, page 255. ACM, 1999.

[40] Y.B. Wu, Q. Li, R.S. Bot, and X. Chen. KIP: a keyphrase identification program with learning functions. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004*, pages 450–454, 2004.

[41] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[42] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks*, 31(11-16):1361–1374, 1999.

APPENDIX A

Evaluation: A usability study of SemKPSearch

The evaluation process of SemKPSearch is divided into two parts. In the first part, you are given two search interfaces: SemKPSearch and Google Desktop. You are asked to compare these two interfaces on the same document set and evaluate the retrieval success by scoring the search results. In the second part, you are given four instances of SemKPSearch and you are expected to examine SemKPSearch on different indexes which are produced for the same document set using different approaches.

Part 1:

In this part, you are asked to search documents for 4 key phrases on SemKPSearch and Google Desktop and evaluate the search results. Below, you will find the instructions about how to use each search interface.

- **SemKPSearch:** SemKPSearch is a searching and browsing tool that enables users to query documents by their author assigned or automatically extracted key phrases, and to expand their queries by suggesting semantically related keyphrases. SemKPSearch indexes are not full text index. Thus user should consider using topic like search terms. When the user searches a phrase, the result set contains the documents that have the search term as its assigned key phrase. The user can expand the result set and reach more documents about the search phrase by clicking on the suggested related key-phrases. Document results are shown by their titles and keyphrases. The user can see the document by clicking on the title. A sample search for "computer graphics" using SemKPSearch can be seen in Figure A.1.

- **Google Desktop:** Google Desktop is an application that enables users to search the files in the computer in a manner similar to searching the web with Google. It provides

full text search on the files. For this evaluation, we configured Google Desktop to index only one folder which contains our evaluation document set. Since Google Desktop searches every word in the query text, please use quotations around the search term in order to gather more relevant and comparable results. Also when a search is done; by default, Google Desktop shows the search results as sorted by date. Click "Sort by relevance" after each search to begin evaluation. To see the content of a document in the result set, the file name can be clicked or simply preview link below the search result can be used. A sample search for "computer graphics" using Google Desktop can be seen in Figure A.2.

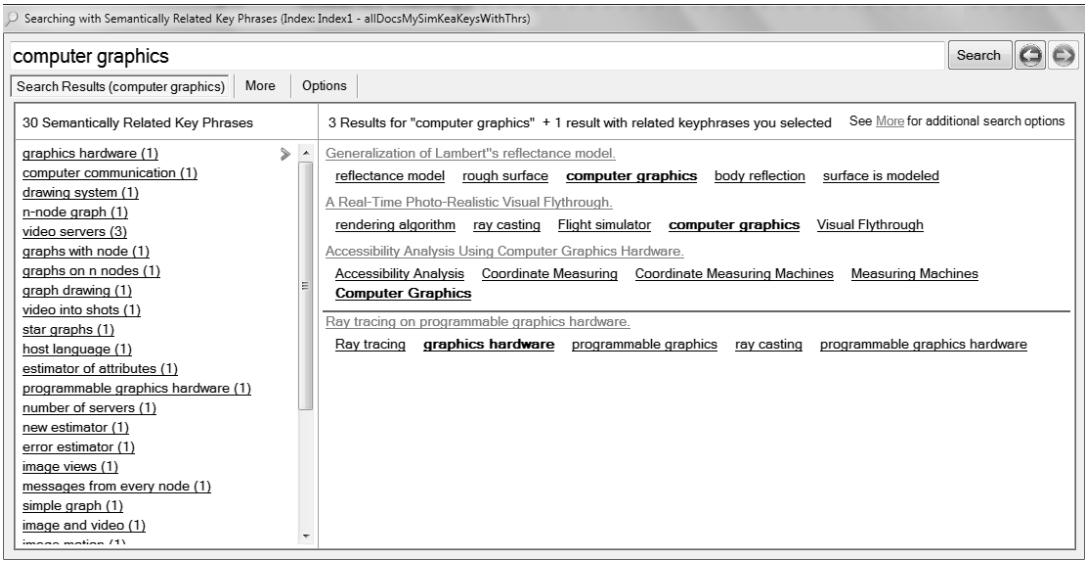

Figure A.1: A sample search with SemKPSearch

Evaluation Task for Part 1:

Open the first worksheet named "Part1" in the given MS Excel file. You will see four tables to fill. For each table, pick a key phase from the list, search them on both SemKPSearch with Index1 and Google Desktop. Judge the relevance of each document in the result set to the query, and give a score between 0 and 4 (namely 0:Irrelevant, 1:Poorly relevant, 2:Partially relevant, 3:Relevant, 4:Completely relevant) to the first 10 results and note the scores on the corresponding table. During scoring SemKPSearch results, if the result set contains less than 10 documents, expand the result set by clicking on the suggested related key phrases in the given order. To expand the result set, first click 3 key phrases and give scores to the newly

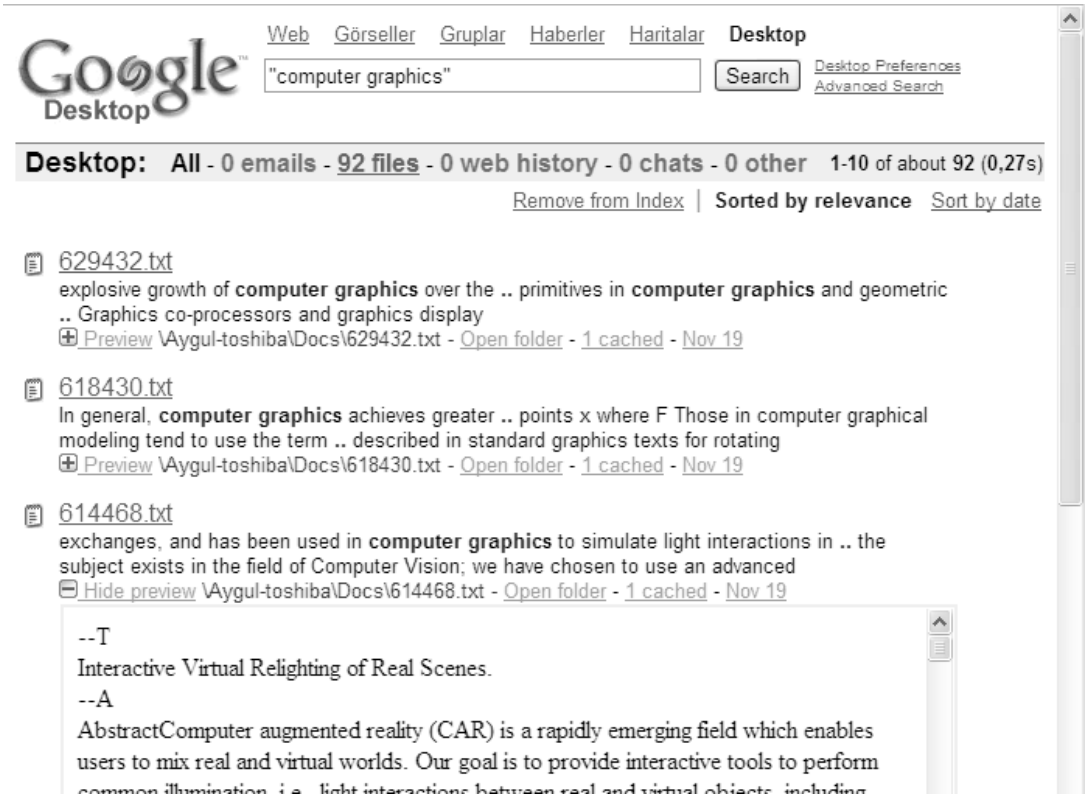

Figure A.2: A sample search using Google Desktop

added documents. If you have not scored 10 documents yet, then continue by expanding 2 key phrases at a time and giving scores to new results. After each expansion, please use the next row in the table to record your scores.

Part 2:

In this part you are expected to examine key phrase suggestions of SemKPSearch on different indexes which are produced for the same document set using different approaches. For the evaluation task, you will be given four instances of SemKPSearch each with a different index.

Evaluation Task for Part 2:

Open the second worksheet named "Part2" in the given MS Excel file. You will see four tables to fill. Each table is for another instance of SemKPSearch with indexes 1-4. In the first table, select four keyphrases that you want to search. Once you select the keyphrases for the first table, keyphrases on the other tables will be automatically filled. For each table, search the selected key phrase on the corresponding SemKPSearch instance, judge the relevance of the first 15 suggested keyphrases to the query term, give a score between 0 and 4 (namely 0:Irrelevant, 1:Poorly relevant, 2:Partially relevant, 3:Relevant, 4:Completely relevant) and note the scores on the table.

After you complete the evaluation, please append your name to the end of the file name and save the MS Excel file. Thank you for your participation.

APPENDIX B

User and System Scores for The Evaluation

Here, we give the scores which the evaluators filled in the evaluation forms.

B.1 Evaluation Form for Part 1

Table B.1: Evaluation form used in part 1 filled with sample data.

Search phrase:	**Score for document in order:**									
clustering algorithm	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4	3	3							
Results aided with 3 key phrases				3	4	4	4			
Results aided with 5 key phrases								4		
Results aided with 7 key phrases									3	4
Results aided with 9 key phrases										
Google Desktop results	2	1	1	3	3	4	1	1	0	3

B.2 Evaluation Form for Part 2

Table B.2: Evaluation form used in part 2 filled with sample data.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
sensor networks	4	4	2	4	4	0	4	4	1	2	4	0	1	2	0
disk management	3	3	3	2	3	3	3	0	4	3	4	1	0	0	1
application development	4	3	3	4	3	2	2	3	4	2	2	3	4	3	0
sound frequency	4	3	3	3	4	3	2	3	3	3	3	1	3	1	1

B.3 Collected Data from Evaluation Part 1

B.3.1 Average Document Retrieval Scores for Indexed Keyphrases

Table B.3: Evaluator scores for *description logics*.

Search phrase:	**Score for document in order:**									
description logics	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4									
Results aided with 3 key phrases		4	2							
Results aided with 5 key phrases				2	1	2	2			
Results aided with 7 key phrases								2	2	1
Results aided with 9 key phrases										
Google Desktop results	4	4	2	1	1					

Table B.4: Evaluator scores for *fault detection*.

Search phrase:	**Score for document in order:**									
fault detection	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4									
Results aided with 3 key phrases		4	2	2						
Results aided with 5 key phrases					1					
Results aided with 7 key phrases						3				
Results aided with 9 key phrases							1	3		
Google Desktop results	4	3	1	2	2	1	1	0	1	0

Table B.5: Evaluator scores for *data caches*.

Search phrase:	**Score for document in order:**									
data caches	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4	4								
Results aided with 3 key phrases			4	3	4					
Results aided with 5 key phrases						3	4			
Results aided with 7 key phrases								4	3	4
Results aided with 9 key phrases										
Google Desktop results	4	4	3	2	4	1	4	4	3	3

Table B.6: Evaluator scores for *test cases*.

Search phrase:	**Score for document in order:**									
test cases	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4	2	4	2						
Results aided with 3 key phrases					4	4	4	4	0	2
Results aided with 5 key phrases										
Results aided with 7 key phrases										
Results aided with 9 key phrases										
Google Desktop results	1	2	3	1	0	0	0	0	0	0

Table B.7: Evaluator scores for *information retrieval*.

Search phrase:	**Score for document in order:**									
information retrieval	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4	4	4							
Results aided with 3 key phrases				4	3	0	0	0		
Results aided with 5 key phrases									0	3
Results aided with 7 key phrases										
Results aided with 9 key phrases										
Google Desktop results	1	3	2	4	3	3	4	2	2	2

Table B.8: Evaluator scores for *clustering algorithm*.

Search phrase:	**Score for document in order:**									
clustering algorithm	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4	3	3,5							
Results aided with 3 key phrases				3	3,5	4	3,5			
Results aided with 5 key phrases								3,5		
Results aided with 7 key phrases									3	3,5
Results aided with 9 key phrases										
Google Desktop results	3	1,5	2,5	2,5	3,5	3,5	1,5	1,5	1,5	2

Table B.9: Evaluator scores for *sensor networks*.

Search phrase:	**Score for document in order:**									
sensor networks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	3,5	3,5	3,5	3,5						
Results aided with 3 key phrases					3,5	1				
Results aided with 5 key phrases										
Results aided with 7 key phrases							0	0,5		
Results aided with 9 key phrases									0,5	0,5
Google Desktop results	2	3	3	3	3	2	2	1	0	0

Table B.10: Evaluator scores for *categorization methods*.

Search phrase:	**Score for document in order:**									
categorization methods	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4									
Results aided with 3 key phrases		2	4	4						
Results aided with 5 key phrases					2	2				
Results aided with 7 key phrases							2	2	4	
Results aided with 9 key phrases										1
Google Desktop results	4	4	3	4	2					

Table B.11: Evaluator scores for *parallel programs*.

Search phrase:	**Score for document in order:**									
parallel programs	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4	4	3,5							
Results aided with 3 key phrases				3,5	3,5	3	2,5	2,5	2	
Results aided with 5 key phrases										1,5
Results aided with 7 key phrases										
Results aided with 9 key phrases										
Google Desktop results	3,5	4	2	2,5	2,5	1,5	2	2,5	3,5	0,5

Table B.12: Evaluator scores for *packet routing*.

Search phrase:	**Score for document in order:**									
packet routing	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results	4	4								
Results aided with 3 key phrases			2,7	3						
Results aided with 5 key phrases					3,3	3				
Results aided with 7 key phrases							3	2		
Results aided with 9 key phrases										
Google Desktop results	4	4	3,7	3,3	2,7	2,3	1	0,7	1	1

B.3.2 Average Document Retrieval Scores for Non-indexed Phrases

Table B.13: Evaluator scores for *application development*.

Search phrase:	**Score for document in order:**									
application development	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results										
Results aided with 3 key phrases	3	3	2							
Results aided with 5 key phrases				3	3	1	2	2	3	1
Results aided with 7 key phrases										
Results aided with 9 key phrases										
Google Desktop results	2	2	2	2	2	2	1	2	1	1

Table B.14: Evaluator scores for *formal languages*.

Search phrase:	**Score for document in order:**									
formal languages	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results										
Results aided with 3 key phrases	1	3	3	3						
Results aided with 5 key phrases					3	0				
Results aided with 7 key phrases							0	2		
Results aided with 9 key phrases									3	3
Google Desktop results	4	3	1	1	3	4	3	0	2	0

Table B.15: Evaluator scores for *disk management*.

Search phrase:	**Score for document in order:**									
disk management	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results										
Results aided with 3 key phrases	3	2	1	2	2	1	2	2	1	
Results aided with 5 key phrases										2
Results aided with 7 key phrases										
Results aided with 9 key phrases										
Google Desktop results	2	1								

Table B.16: Evaluator scores for *file formats*.

Search phrase:	**Score for document in order:**									
file formats	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results										
Results aided with 3 key phrases	3	3	0	1	1	1	0	1	1	1
Results aided with 5 key phrases										
Results aided with 7 key phrases										
Results aided with 9 key phrases										
Google Desktop results	4	3	4	1	1	0	1	1		

Table B.17: Evaluator scores for *tree topology*.

Search phrase:	**Score for document in order:**									
tree topology	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results										
Results aided with 3 key phrases	0,7	0,3	1,3							
Results aided with 5 key phrases				2	0,3	0,3				
Results aided with 7 key phrases							2,3	0,3		
Results aided with 9 key phrases									2,7	1,7
Google Desktop results	1	1	1							

Table B.18: Evaluator scores for *graph data structure*.

Search phrase:	**Score for document in order:**									
graph data structure	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results										
Results aided with 3 key phrases	1,3	1,7	1,3	3	1,3	0,7	1	1	0,7	
Results aided with 5 key phrases										0,3
Results aided with 7 key phrases										
Results aided with 9 key phrases										
Google Desktop results	1	0	0							

Table B.19: Evaluator scores for *complexity analysis*.

Search phrase:	**Score for document in order:**									
complexity analysis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Direct search results										
Results aided with 3 key phrases	2,3	2,3	2,3	2,7	2,7	1,3				
Results aided with 5 key phrases							0,7			
Results aided with 7 key phrases								2,7	2,7	
Results aided with 9 key phrases										2
Google Desktop results	1	1,7	1,7	2,7	2,7	1,7	3,3	1,7	3,3	1,3

B.4 Collected Data from Evaluation Part 2

B.4.1 Scores for KEA_SimLi Index

Table B.20: User evaluation scores for *KEA_SimLi* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
abstract data types	4	4	4	4	4	3	3	2	0	2	0	0	0	0	0
application development	4	3	3	4	3	2	2	3	4	2	2	3	4	3	0
application development	4	4	4	4	4	2	3	4	3	2	0	4	3	3	1
automata theory	4	4	4	1	4	4	1	4	4	1	4	0	0	4	1
categorization methods	4	3	4	3	3	3	2	2	3	3	1	4	2	0	0
categorization methods	2	4	4	2	2	2	4	2	4	4	2	4	2	2	2
clustering algorithm	4	4	4	4	3	1	4	0	0	4	4	3	4	3	3
clustering algorithm	4	4	4	4	3	3	3	3	0	0	3	3	3	4	3
complexity analysis	4	3	2	1	1	2	3	3	3	2	0	2	3	1	1
complexity analysis	3	3	3	3	2	3	4	4	3	3	0	3	4	0	0
data caches	4	4	4	2	2	2	3	2	1	2	3	2	2	2	2
data caches	4	4	4	4	4	4	4	0	4	4	4	3	3	3	4
disk management	4	4	2	2	4	4	4	0	3	2	3	2	0	0	0
disk management	4	4	4	3	3	2	4	2	3	2	3	0	0	0	0
disk management	3	3	3	2	3	3	3	0	4	3	4	1	0	0	1
fault detection	4	4	4	4	2	3	1	4	3	3	4	3	1	1	2
formal languages	0	1	1	4	0	3	4	4	4	4	4	2	0	4	4
graph data structure	3	3	3	1	1	1	1	2	2	1	2	2	1	1	1
graph data structure	2	2	2	0	0	0	1	4	3	2	1	2	0	0	1
information retrieval	4	4	2	1	4	4	0	4	4	4	0	3	3	2	4
information retrieval	4	4	3	3	3	3	2	3	3	3	2	2	2	2	3
polynomial time approximation	4	4	4	4	4	3	3	3	2	3	3	3	3	2	1
recognition	0	4	1	4	4	4	4	4	0	2	4	4	4	4	0
sensor networks	4	4	0	4	4	4	4	1	2	4	0	1	1	1	2
sensor networks	4	4	0	4	2	3	3	2	0	3	2	0	2	2	3
sensor networks	4	4	2	4	4	0	4	4	1	2	4	0	1	2	0
sensor networks	4	4	3	4	4	1	4	4	3	2	4	2	3	4	1
sound frequency	4	3	3	3	4	3	2	3	3	3	3	1	3	1	1
test cases	4	2	4	2	2	4	4	1	1	1	2	1	1	3	4
tree topology	2	2	3	3	2	2	2	2	2	2	1	1	1	1	1
tree topology	1	1	3	3	0	3	1	2	2	2	2	1	0	0	2
tree topology	3	1	3	3	1	2	2	2	2	3	0	1	0	0	2

Table B.21: Similarity scores for keyphrases suggested with *KEA_SimLi* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
abstract data types	0,929	0,929	0,913	0,871	0,871	0,845	0,818	0,814	0,800	0,799	0,795	0,793	0,792	0,790	0,782
application development	0,883	0,877	0,869	0,862	0,855	0,848	0,811	0,807	0,805	0,800	0,799	0,796	0,795	0,793	0,788
automata theory	0,902	0,874	0,867	0,809	0,807	0,799	0,792	0,744	0,733	0,727	0,716	0,697	0,696	0,694	0,681
categorization methods	0,960	0,955	0,888	0,875	0,875	0,868	0,860	0,858	0,852	0,850	0,845	0,840	0,836	0,824	0,794
clustering algorithm	1	1	1	1	0,945	0,881	0,847	0,840	0,832	0,827	0,825	0,822	0,821	0,819	0,796
complexity analysis	0,867	0,831	0,821	0,821	0,776	0,759	0,755	0,750	0,742	0,741	0,740	0,721	0,718	0,712	0,704
data caches	1,000	0,904	0,889	0,880	0,832	0,825	0,823	0,818	0,817	0,809	0,797	0,789	0,778	0,772	0,764
disk management	0,832	0,812	0,771	0,768	0,762	0,748	0,744	0,713	0,710	0,709	0,707	0,694	0,689	0,689	0,685
fault detection	1,000	0,999	0,941	0,920	0,884	0,884	0,884	0,835	0,834	0,823	0,818	0,807	0,796	0,785	0,785
formal languages	0,771	0,765	0,765	0,729	0,706	0,697	0,695	0,691	0,688	0,687	0,685	0,679	0,666	0,665	0,662
graph data structure	0,842	0,842	0,842	0,823	0,800	0,798	0,784	0,782	0,781	0,778	0,777	0,767	0,754	0,754	0,753
information retrieval	0,898	0,859	0,853	0,845	0,818	0,818	0,814	0,806	0,789	0,788	0,771	0,769	0,769	0,761	0,760
polynomial time approximation	1,000	0,940	0,939	0,932	0,932	0,877	0,876	0,810	0,796	0,776	0,748	0,743	0,743	0,738	0,737
recognition	0,968	0,923	0,874	0,867	0,849	0,845	0,838	0,837	0,819	0,818	0,814	0,790	0,785	0,769	0,767
sensor networks	1,000	0,911	0,902	0,877	0,855	0,851	0,847	0,840	0,815	0,811	0,806	0,803	0,767	0,764	0,759
sound frequency	0,861	0,825	0,806	0,774	0,757	0,752	0,750	0,737	0,732	0,732	0,728	0,704	0,701	0,698	0,694
test cases	1,000	0,978	0,936	0,934	0,929	0,928	0,897	0,874	0,869	0,866	0,862	0,849	0,842	0,839	0,838
tree topology	0,770	0,770	0,731	0,731	0,718	0,702	0,687	0,658	0,652	0,651	0,642	0,636	0,636	0,635	0,633

B.4.2 Scores for KEA_SimWP Index

Table B.22: User evaluation scores for *KEA_SimWP* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
application development	3	4	3	3	4	3	2	2	2	2	3	2	2	3	2
application development	4	4	4	4	3	0	4	4	4	4	4	4	3	4	0
automata theory	4	1	4	4	1	0	0	2	0	0	0	0	0	0	0
clustering algorithm	4	4	4	3	1	3	0	2	3	0	0	4	4	1	0
clustering algorithm	4	4	4	3	3	3	2	1	3	0	0	3	3	0	0
complexity analysis	3	3	3	2	1	2	2	2	1	2	2	1	1	2	0
complexity analysis	4	3	4	3	3	0	4	4	0	3	3	0	0	3	0
data caches	2	2	3	2	3	3	1	1	1	1	1	1	1	1	0
data caches	0	0	4	1	4	1	0	0	0	0	0	0	0	0	0
disk management	4	4	4	4	4	4	3	2	0	0	3	0	0	4	2
disk management	4	4	4	3	1	3	3	3	0	0	0	3	0	0	2
disk management	3	3	3	3	3	3	3	2	1	0	0	4	1	2	3
fault detection	4	4	4	4	2	3	1	2	4	4	3	0	0	0	0
formal languages	4	4	4	3	4	4	3	1	0	0	3	4	2	4	2
graph data structure	3	2	3	2	2	2	1	0	0	0	1	0	0	0	1
graph data structure	2	2	2	1	1	0	0	0	0	1	2	0	0	0	3
recognition	0	0	2	4	0	0	1	4	4	4	0	0	0	1	4
sensor networks	4	4	0	1	3	2	4	1	0	0	0	0	4	4	2
sensor networks	4	2	2	2	2	2	4	2	1	1	1	3	3	0	2
sensor networks	4	4	0	2	2	3	4	2	0	0	0	0	4	4	1
sensor networks	4	4	1	4	4	4	4	3	1	1	1	1	4	4	4
sound frequency	3	3	3	3	3	4	3	3	2	1	2	2	2	1	3
test cases	4	2	3	1	4	1	1	3	3	1	4	0	4	0	4
tree topology	3	2	1	1	3	3	1	1	1	1	1	1	1	2	1
tree topology	1	1	1	1	3	3	0	2	0	0	0	0	0	0	2
tree topology	3	3	1	2	3	3	0	1	0	1	1	0	0	2	0

Table B.23: Similarity scores for keyphrases suggested with *KEA_SimWP* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
application development	0,944	0,944	0,933	0,933	0,930	0,929	0,929	0,919	0,919	0,914	0,911	0,906	0,906	0,905	0,905
automata theory	0,938	0,915	0,912	0,889	0,888	0,887	0,882	0,882	0,873	0,867	0,867	0,861	0,860	0,860	0,860
clustering algorithm	1,000	1,000	1,000	0,971	0,912	0,875	0,860	0,853	0,853	0,853	0,853	0,853	0,853	0,835	0,834
complexity analysis	0,944	0,938	0,889	0,875	0,875	0,864	0,858	0,858	0,856	0,853	0,853	0,845	0,845	0,845	0,844
data caches	0,937	0,929	0,900	0,900	0,900	0,900	0,889	0,889	0,872	0,868	0,868	0,865	0,858	0,858	0,856
disk management	0,944	0,944	0,921	0,906	0,878	0,865	0,856	0,850	0,844	0,840	0,833	0,833	0,828	0,825	0,823
fault detection	1,000	1,000	0,945	0,929	0,917	0,905	0,905	0,897	0,893	0,885	0,885	0,883	0,882	0,880	0,876
formal languages	0,875	0,856	0,848	0,846	0,833	0,833	0,833	0,817	0,808	0,808	0,807	0,806	0,800	0,797	0,794
graph data structure	0,880	0,880	0,880	0,876	0,876	0,876	0,876	0,872	0,872	0,865	0,860	0,860	0,853	0,850	0,850
recognition	0,982	0,943	0,941	0,905	0,879	0,875	0,874	0,872	0,872	0,872	0,860	0,860	0,860	0,857	0,854
sensor networks	1,000	1,000	0,941	0,938	0,912	0,912	0,912	0,912	0,906	0,906	0,900	0,898	0,893	0,893	0,890
sound frequency	0,874	0,871	0,871	0,871	0,871	0,867	0,858	0,856	0,839	0,839	0,839	0,839	0,839	0,839	0,833
test cases	1,000	0,985	0,962	0,962	0,954	0,938	0,929	0,929	0,928	0,923	0,920	0,914	0,914	0,914	0,910
tree topology	0,874	0,874	0,871	0,858	0,857	0,857	0,829	0,818	0,804	0,800	0,800	0,790	0,788	0,782	0,781

B.4.3 Scores for Author_SimLi Index

Table B.24: User evaluation scores for *Author_SimLi* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
abstract data types	4	4	4	4	4	2	0	0	2	0	0	0	3	3	2
application development	3	4	3	4	2	0	3	0	2	2	3	4	0	3	3
application development	4	4	4	4	4	0	4	0	3	4	3	3	1	3	3
automata theory	4	4	1	4	4	4	1	4	1	4	4	1	1	0	0
categorization methods	4	4	3	3	3	3	3	2	2	3	1	3	2	0	1
categorization methods	4	4	2	4	4	4	3	2	2	4	2	4	4	2	2
clustering algorithm	4	4	4	4	0	4	4	4	4	3	4	3	4	3	3
clustering algorithm	4	4	4	3	0	3	3	3	3	3	2	2	3	2	1
complexity analysis	3	3	2	2	2	2	3	2	2	2	1	1	2	0	1
complexity analysis	3	4	3	3	1	4	4	3	3	3	4	0	4	0	0
data caches	4	3	3	3	3	2	2	3	3	2	2	2	3	2	2
data caches	4	4	4	4	4	4	4	4	4	1	4	4	3	4	4
disk management	4	4	4	1	0	4	1	4	2	1	1	2	4	2	3
disk management	4	4	4	1	0	4	0	3	0	0	0	0	3	0	3
disk management	3	4	3	3	0	3	2	3	3	1	1	2	4	2	4
fault detection	4	4	4	4	4	4	0	4	4	4	4	3	3	3	3
formal languages	4	4	4	2	4	4	0	2	2	4	4	0	0	0	4
graph data structure	3	3	3	2	1	1	1	1	2	2	1	1	0	1	1
graph data structure	4	2	2	0	0	4	4	3	4	3	1	2	0	1	1
information retrieval	4	4	2	4	4	4	0	4	4	4	2	4	4	4	4
information retrieval	4	4	1	4	4	4	0	4	4	4	2	2	2	3	1
polynomial time approximation	4	4	4	3	2	4	0	3	4	0	0	2	2	0	0
recognition	3	4	0	4	4	4	4	4	4	4	1	3	3	0	0
sensor networks	4	4	4	2	4	3	3	3	2	3	2	2	2	0	0
sensor networks	4	4	3	3	3	3	3	2	2	0	2	0	0	1	1
sensor networks	3	4	4	4	3	4	4	2	2	2	2	1	1	0	0
sensor networks	4	4	4	3	4	4	4	3	3	3	4	2	3	1	1
sound frequency	4	4	3	3	2	3	2	1	1	3	1	2	2	2	1
test cases	4	0	0	1	0	0	0	4	0	4	0	2	0	1	4
tree topology	2	3	2	1	1	1	1	1	1	2	1	1	2	1	1
tree topology	4	4	2	0	3	3	3	3	3	0	0	2	3	0	0
tree topology	3	3	2	2	3	3	3	3	3	2	1	2	3	0	0

Table B.25: Similarity scores for keyphrases suggested with *Author_SimLi* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
abstract data types	1,000	0,871	0,842	0,800	0,786	0,767	0,761	0,761	0,761	0,757	0,743	0,740	0,739	0,738	0,736
application development	0,927	0,895	0,895	0,862	0,860	0,824	0,822	0,819	0,818	0,818	0,805	0,794	0,789	0,779	0,761
automata theory	0,867	0,818	0,813	0,810	0,807	0,799	0,755	0,744	0,738	0,716	0,710	0,706	0,680	0,674	0,670
categorization methods	0,945	0,945	0,945	0,862	0,858	0,850	0,841	0,836	0,824	0,804	0,794	0,791	0,777	0,773	0,768
clustering algorithm	0,882	0,882	0,881	0,830	0,827	0,825	0,778	0,747	0,737	0,736	0,722	0,720	0,718	0,706	0,702
complexity analysis	0,869	0,847	0,821	0,785	0,778	0,767	0,758	0,752	0,750	0,742	0,741	0,718	0,704	0,704	0,702
data caches	1,000	0,891	0,891	0,891	0,845	0,820	0,808	0,805	0,790	0,779	0,763	0,761	0,746	0,746	0,745
disk management	0,832	0,821	0,812	0,774	0,770	0,744	0,712	0,711	0,709	0,694	0,678	0,677	0,675	0,668	0,667
fault detection	1,000	0,941	0,884	0,884	0,884	0,884	0,842	0,823	0,818	0,815	0,801	0,795	0,790	0,787	0,776
formal languages	0,931	0,795	0,752	0,726	0,718	0,707	0,702	0,695	0,695	0,687	0,669	0,668	0,666	0,666	0,662
graph data structure	0,922	0,842	0,842	0,798	0,790	0,784	0,784	0,784	0,782	0,767	0,764	0,747	0,732	0,728	0,726
information retrieval	0,922	0,915	0,890	0,877	0,875	0,874	0,839	0,836	0,833	0,821	0,819	0,817	0,814	0,806	0,804
polynomial time approximation	0,940	0,932	0,877	0,870	0,765	0,754	0,743	0,743	0,743	0,731	0,723	0,720	0,715	0,713	0,713
recognition	0,923	0,914	0,888	0,873	0,869	0,845	0,838	0,837	0,834	0,833	0,819	0,819	0,819	0,818	0,812
sensor networks	1,000	0,911	0,902	0,902	0,829	0,822	0,822	0,822	0,815	0,796	0,771	0,771	0,767	0,759	0,733
sound frequency	0,809	0,795	0,785	0,769	0,750	0,732	0,721	0,681	0,678	0,669	0,662	0,650	0,645	0,642	0,640
test cases	0,936	0,904	0,904	0,904	0,892	0,871	0,856	0,842	0,837	0,821	0,817	0,817	0,817	0,811	0,804
tree topology	0,813	0,798	0,729	0,718	0,702	0,702	0,702	0,702	0,702	0,702	0,691	0,665	0,658	0,653	0,653

B.4.4 Scores for Author_SimWP Index

Table B.26: User evaluation scores for *Author_SimWP* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
abstract data types	4	4	4	3	2	3	2	0	2	0	0	0	2	0	1
application development	3	3	4	4	3	2	4	3	0	2	2	3	0	1	1
application development	4	4	4	4	4	3	3	4	0	3	3	4	0	4	4
automata theory	1	4	1	2	2	0	0	0	0	0	2	4	0	0	0
categorization methods	4	4	4	3	2	4	2	2	1	1	1	0	0	0	2
categorization methods	4	2	4	2	2	4	2	2	2	2	2	1	1	2	4
clustering algorithm	0	0	0	0	2	4	0	0	2	0	0	0	2	0	0
clustering algorithm	4	0	0	0	2	3	0	0	2	0	0	0	1	0	0
data caches	4	3	3	2	2	2	2	1	1	1	2	2	2	2	2
data caches	4	1	1	0	4	0	0	0	1	1	1	1	1	1	4
disk management	4	4	4	4	1	2	3	1	2	3	2	4	1	1	0
disk management	4	4	4	4	0	0	2	0	0	2	1	2	0	0	0
disk management	4	3	3	3	3	3	2	2	2	2	2	3	0	0	0
fault detection	4	4	0	4	4	4	4	4	4	3	3	1	4	3	0
formal languages	4	2	1	0	0	0	4	4	4	0	4	1	0	4	0
graph data structure	4	3	3	2	2	2	1	2	1	1	1	1	1	2	2
graph data structure	4	2	2	0	2	2	2	2	0	1	0	0	0	2	0
information retrieval	4	4	3	0	2	4	2	2	2	2	3	0	2	3	3
information retrieval	4	4	3	2	2	3	3	3	2	2	2	0	0	0	2
polynomial time approximation	4	4	4	0	3	3	0	4	4	0	3	0	0	0	0
recognition	2	2	0	0	4	4	0	0	1	0	1	4	4	4	4
sensor networks	0	4	4	2	4	2	2	0	2	0	4	0	1	0	0
sensor networks	0	4	4	2	3	0	2	2	2	2	3	2	0	1	1
sensor networks	0	3	4	2	4	4	4	2	4	2	2	3	1	1	3
sensor networks	2	4	4	3	4	4	4	3	4	3	3	3	3	4	2
sound frequency	3	2	4	2	4	3	1	1	0	2	1	1	3	1	2
test cases	4	0	0	2	0	0	0	0	0	3	0	2	2	3	4
tree topology	2	2	1	1	2	1	1	1	1	1	1	2	1	1	2
tree topology	4	0	0	0	4	2	0	0	0	0	0	0	0	0	0
tree topology	3	1	1	2	2	2	1	0	0	0	0	2	0	0	2

Table B.27: Similarity scores for keyphrases suggested with *Author_SimWP* index.

	Your scores for the first 15 keyphrase suggestions:														
Search phrase:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
abstract data types	1,000	1,000	0,956	0,925	0,898	0,898	0,882	0,880	0,880	0,877	0,870	0,866	0,859	0,852	0,848
application development	0,974	0,958	0,958	0,944	0,935	0,933	0,933	0,924	0,917	0,916	0,906	0,906	0,906	0,902	0,902
automata theory	0,902	0,889	0,888	0,882	0,882	0,868	0,861	0,860	0,860	0,859	0,857	0,857	0,856	0,856	0,855
categorization methods	0,917	0,917	0,917	0,904	0,900	0,900	0,900	0,900	0,900	0,900	0,900	0,895	0,895	0,895	0,889
clustering algorithm	0,912	0,864	0,864	0,853	0,853	0,833	0,833	0,833	0,813	0,813	0,812	0,812	0,812	0,802	0,800
data caches	1,000	0,900	0,900	0,853	0,834	0,833	0,821	0,821	0,817	0,812	0,812	0,812	0,812	0,812	0,812
disk management	0,944	0,944	0,944	0,921	0,912	0,878	0,850	0,849	0,844	0,841	0,829	0,823	0,821	0,821	0,817
fault detection	1,000	0,945	0,935	0,921	0,905	0,905	0,905	0,905	0,900	0,897	0,897	0,890	0,885	0,885	0,880
formal languages	0,875	0,858	0,853	0,853	0,845	0,839	0,833	0,833	0,833	0,829	0,826	0,817	0,813	0,812	0,807
graph data structure	0,920	0,880	0,880	0,876	0,865	0,850	0,844	0,843	0,841	0,841	0,840	0,840	0,840	0,829	0,821
information retrieval	0,954	0,920	0,917	0,906	0,900	0,885	0,885	0,885	0,885	0,885	0,875	0,871	0,864	0,864	0,861
polynomial time approximation	0,946	0,909	0,909	0,883	0,873	0,861	0,857	0,845	0,845	0,843	0,841	0,838	0,834	0,833	0,833
recognition	0,947	0,947	0,943	0,933	0,912	0,905	0,889	0,875	0,875	0,874	0,872	0,872	0,872	0,872	0,872
sensor networks	1,000	1,000	0,965	0,938	0,917	0,917	0,917	0,912	0,912	0,912	0,893	0,889	0,889	0,880	0,872
sound frequency	0,871	0,864	0,853	0,840	0,833	0,822	0,817	0,817	0,817	0,814	0,812	0,812	0,808	0,805	0,805
test cases	0,962	0,952	0,952	0,952	0,934	0,929	0,929	0,929	0,923	0,923	0,923	0,914	0,907	0,907	0,907
tree topology	0,950	0,858	0,855	0,845	0,818	0,808	0,807	0,804	0,804	0,798	0,798	0,788	0,788	0,785	0,782