USING CONCEPT MAPS AND FUZZY SET INFORMATION RETRIEVAL MODEL TO DYNAMICALLY PERSONALIZE RSS FEEDS

A THESIS IN ENGINEERING SYSTEMS MANAGEMENT
Master of Science in Engineering Systems Management, IT Management Theme

Presented to the faculty of the American University of Sharjah
College of Engineering
in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
HEBA MAHMOUD ISMAIL
B.S. 2007

Sharjah, UAE
Jan 2011

We approve the thesis of Heba Mahmoud Ismail — Date of signature

_____ _____

Dr. Imran Ahmed Zaulkernan
Associate Professor
Computer Science and Engineering
Thesis Advisor

_____ _____

Dr. Fouad Ben Abdelaziz
Professor
Engineering Systems Management Graduate Program
Graduate Committee

_____ _____

Dr. Hazim El-Baz
Associate professor
Engineering Systems Management Graduate Program
Graduate Committee

_____ _____

Dr. Abdelkader Daghfous
Associate Professor
School of Business & Management, Dept. of MIS
Graduate Committee

_____ _____

Dr. Moncer Abdelhamid Hariga
Director
Engineering Systems Management Graduate Program

_____ _____

Dr. Hany El-Kadi
Associate Dean, College of Engineering

_____ _____

Dr. Yousef Al-Assaf
Dean, College of Engineering

_____ _____

Dr. Gutam Sen
Vice Provost, Research and Graduate Studies

USING CONCEPT MAPS AND FUZZY SET INFORMATION RETRIEVAL MODEL TO DYNAMICALLY PERSONALIZE RSS FEEDS

Heba M. Ismail, Candidate for the Master of Science Degree

American University of Sharjah, 2010

ABSTRACT

Knowledge has always been at the heart of work and action of people. The amount of knowledge is growing and types of knowledge are becoming more diverse. Effective knowledge management may require going beyond basic knowledge capture. Recent work has focused on enhancing knowledge capturing and codification on the web as it is becoming the dominant source of knowledge. Semantic web and Web 2.0 have opened additional opportunities for knowledge management with technologies and tools such as topic maps and concept maps. However, using such technologies imposed many challenges because of the knowledge codification and generation tools such as blogs, wikis, forums, really simple syndicate (RSS), social tagging and others. This research provides a novel method for instantly updating web users with pieces of knowledge tailored to their personal knowledge interests from RSS feeds by combining fuzzy set information retrieval model with the semantics and structure of concept maps. Experiments show that the proposed method can achieve precision levels ranging between 80% and 100%. In addition, the method is robust and has a higher level of agreement with expert judges than the agreement within judges.

CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

ACKNOWLEDGEMENTS

Praise be to Allah, the most Gracious and Merciful, for providing me the strength and patience to complete this thesis successfully. All praise, thanks, and gratitude ultimately belong to Allah.

Here I would like to express my sincere gratitude and appreciation to several people who have been influential, either directly or indirectly, to make this work possible. Through these humble words, I would like to thank my advisor Dr. Imran Zualkernan who introduced me to the area of knowledge management on the web. Throughout these two years, he provided encouragement, sound advice, valuable teaching and lots of great ideas.

Of course, nothing would have been possible if I did not have such great professors in the department of Engineering Systems Management who put a lot of effort and time to make the course of my study valuable and memorable. I thank all of them for that.

On a personal note, I am much indebted to my husband, Ismail Saeed, for all the help, love, support and patience during my course of study. My Deepest and honest gratitude is due to my parents who raised me up to have a passion for learning and a determination to succeed, I thank them for their constant and unconditional love and support. Special thanks goes to my brothers, Khaldoon, Ahmed, and Mohammed, and to my sisters, Maysoon and Enas, for their constant loving encouragement. Without the sincere support of all these people it would have been impossible for me to finish this work. This thesis is dedicated to all of you!

Besides my advisor , professors and family, I would like to thank all the mobile application experts who took part in my study and were very instrumental in validating my research results.

CHAPTER 1

INTRODUCTION

The World Wide Web has revolutionized the way people access information. Researchers and institutions are increasingly making articles and papers available on the web. News websites, digital forums, wikis, blogs and other forms of online information sources have opened the door for the public to add to the wealth of knowledge on the web. However, information on the Web is largely disorganized, scattered and untraceable. Substantial time is spent filtering out relevant information that exactly meets user's interest.

Many efforts have been made to relate information, rank them and keep track of the updates in a certain field of science by means of information retrieval algorithms and utilities [1]. Search engines such as Google and Yahoo made it much easier to capture specific knowledge on the web. Yet, search engines works with limited number of keywords and generates results based on millions of users. Many search interfaces were created on top of the search engines that restructure users' search queries in order to help users get the most relevant knowledge [2]. Other attempts utilized some of Web 2.0 technologies in order to embed semantic in the user's search queries by using mind maps, concept maps and others [3], [4], [5]. These attempts improved the quality of knowledge returned to the user through search engines. However, the root cause of the problem is still not addressed. Search engines like Google are based on algorithms that generate results ranked according to millions of users' behavior. Therefore, search results may not closely match the specific interests of a single user.

In addition, users need to be continuously updated on newly created knowledge on the web. Just-In-Time knowledge is becoming more critical to web users. Really Simple Syndicate (RSS) technology was employed to keep users updated on new web contents as they become available [6]. However, RSS feeds are unstructured and are far from being specific to a user's actual information interests. For example considering the RSS feed of a news website (e.g. *CNN RSS Feed* [7]); users are presented with links to all new contents published on the website in the form of links to WebPages without specific

categorization as shown in figure 1.1. Using a third party RSS aggregator (e.g. *RSSOwl* [8]), users can categorize RSS according to subject or theme, see figure 1.2. On one hand, the time required to filter relevant information amongst the uncategorized RSS links or amongst the wide range of categories, prevents many people from leveraging the value that RSS feeds bring to them. In addition, user's specific interest is not taken care of as all users are presented with the same links and same categories.

With the problems inherent in the search engines and RSS feeds such as keywords constraints and lack of personalization which result in less relevant knowledge retrieved and lots of time spent on filtering results , a method that takes care of a single user's knowledge interests and provides Just-In-time knowledge relevant to these interests is required.

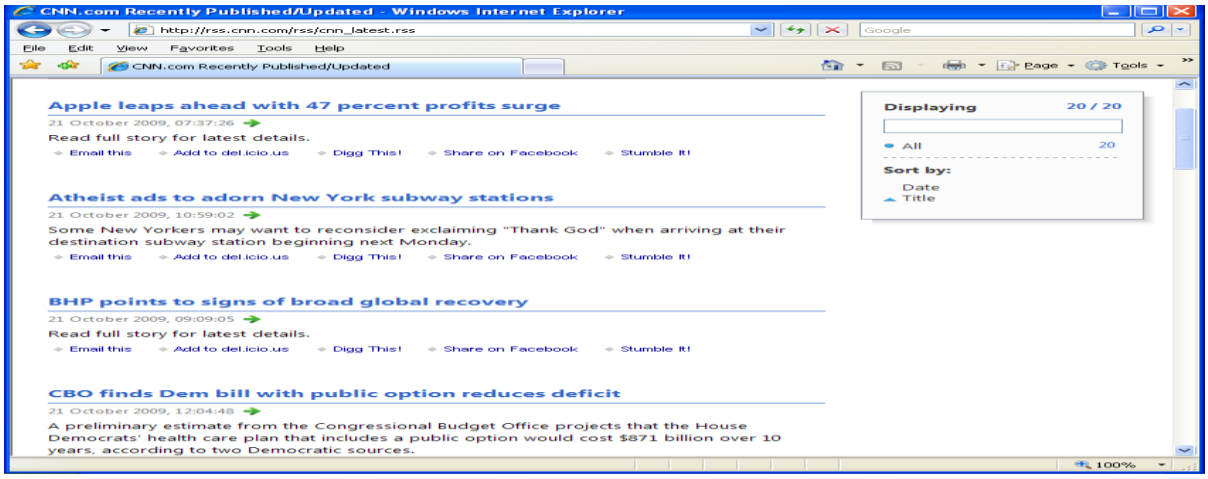


Figure 1.1 CNN RSS Feed

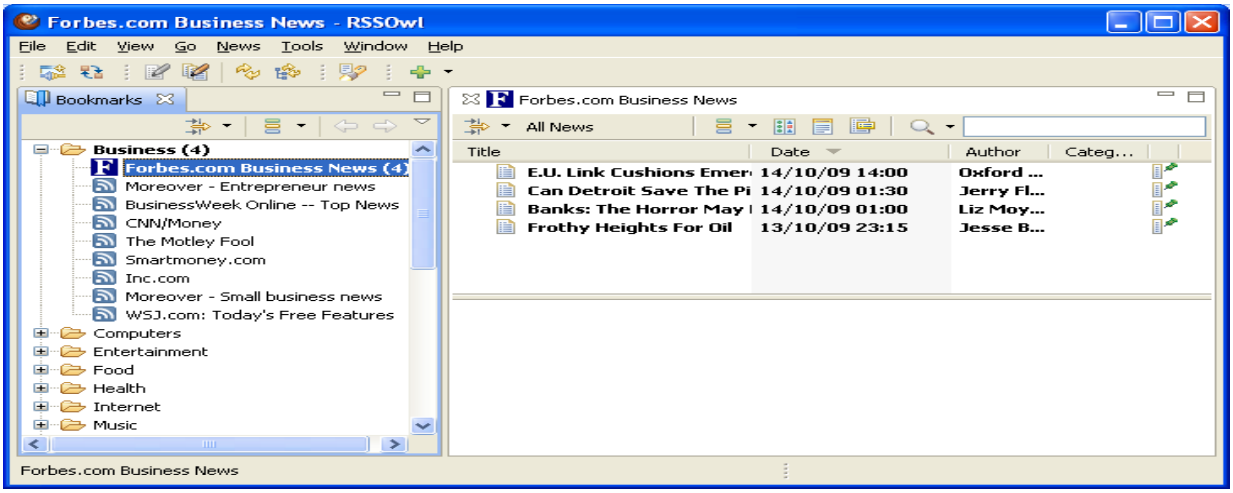


Figure1. 2 RSSOwl Aggregator

1.1 Thesis Objectives and Proposed Methodology

This research addresses the following question: how to instantly update users on newly created knowledge that is specific to their knowledge interests as it becomes available on the web? Current search engines and smart RSS readers suffer from at least one of the following problems:

1) Results are retrieved based on keywords rather than concepts.
2) Search engines allow limited number of keywords. (e.g. Google's input string is limited to 2048 bytes and 10 individual words) [9].
3) Results are ranked based on millions of user's rather than single user.
4) Users fail sometimes in defining their knowledge interests using limited number of keywords.
5) Retrieval of knowledge is based on exact match rather than partial match of meaning.

This research builds upon outcomes of prior research on using concept mapping to define the user's interest in Just-In-Time knowledge searching. RSS feeds are used to provide instant updates of newly created knowledge on the web, concept maps are proposed to represent user's knowledge interest where both the semantic and the structure of the concept map define the interest of the user, and Fuzzy Set Information Retrieval model (Fuzzy Set IR) [1] is used as an information retrieval logic that measures partial similarity and exact match between concepts in the concept map and terms in the RSS feed. The proposed Fuzzy Set IR model is based on custom fuzzy dictionary specific to the user's concept map. This in turn is different from any other information retrieval method which all work based on general thesaurus or indexes based on millions of users. A new method that combines the structure and semantic of the concept map with the sophisticated logic of fuzzy set IR model is proposed to solve the research question. The rest of the thesis is organized as follows:

Chapter 2, discusses related work. It is divided into two parts. Section 2.1 summarizes some of the researches about concept mapping and its applications in information retrieval, while section 2.2 presents some of the work done in the area of fuzzy-set information retrieval model and highlights some researches aimed at enhancing knowledge retrieved from RSS feeds. Chapter 3, introduces information retrieval (IR) as a science and highlights major types of IR along with examples. Fuzzy Set IR model and how it can be implemented is

explained in the last section of this chapter. Chapter 4, explains the overall proposed method and presents step by step how fuzzy thesauri and fuzzy association were combined with the structure of the concept map to retrieve relevant knowledge. Implementation of the method is explained at the end of this chapter. Chapter 5, presents evaluation results of the proposed method at three stages. First assessing consistency within one topological model. second, assessing the consistency amongst the three topological models. Finally assess the effectiveness and precision of the method in retrieving relevant knowledge. Chapter 6, sum up the findings and contributions of this research and highlight possible improvements.

CHAPTER 2

LITERATURE REVIEW

In this chapter, related work is discussed. This chapter is divided into two parts. Section 2.1 introduces concept mapping and summarizes some research about concept mapping and its applications in information retrieval, while section 2.2 presents some work done in the area of information retrieval in general and particularly in fuzzy-set information retrieval model, and highlights some research addressing RSS applications.

2.1 Concept Mapping and its Applications in IR

In this section the use of concept maps in defining user's knowledge interests is explained. Section 2.1.1 introduces concept mapping. Section 2.1.2 highlights the advantage of using concept maps over basic query keywords. Section 2.1.3 explains the affect of concept maps topology on concept's importance. Finally section 2.1.4 presents related work in utilizing concept mapping in knowledge capturing.

2.1.1 About Concept Maps

Concept mapping [10] was developed in educational settings by Joseph Novak, in an effort to design better teaching and learning method. Concept mapping was designed to support the learners by externalizing concepts known to the student, making them visually apparent to assist in connecting them with newly acquired concepts. Concept maps have been used by teachers to assess students' understanding, by students to compare their knowledge and collaboratively refine their understanding, and by experts as a vehicle for modeling and sharing their knowledge.

Concept maps are graphical tools for organizing and representing knowledge. They include concepts, usually enclosed in circles or boxes of some type, and relationships between concepts indicated by a connecting line linking two concepts. Words on the line, referred to as linking words or linking phrases, specify the relationship between the two concepts.

In concept maps, the concepts are represented in a hierarchical fashion with the most general concepts at the top of the map and the less general concepts arranged hierarchically below. The hierarchical structure for a particular domain of knowledge also depends on the context in which that knowledge is being applied. Therefore, it is best to construct concept maps with reference to some particular question we seek to answer, which is called a focus question. For example figure 2.1 shows a concept map that addresses the question "What is a Plant?"

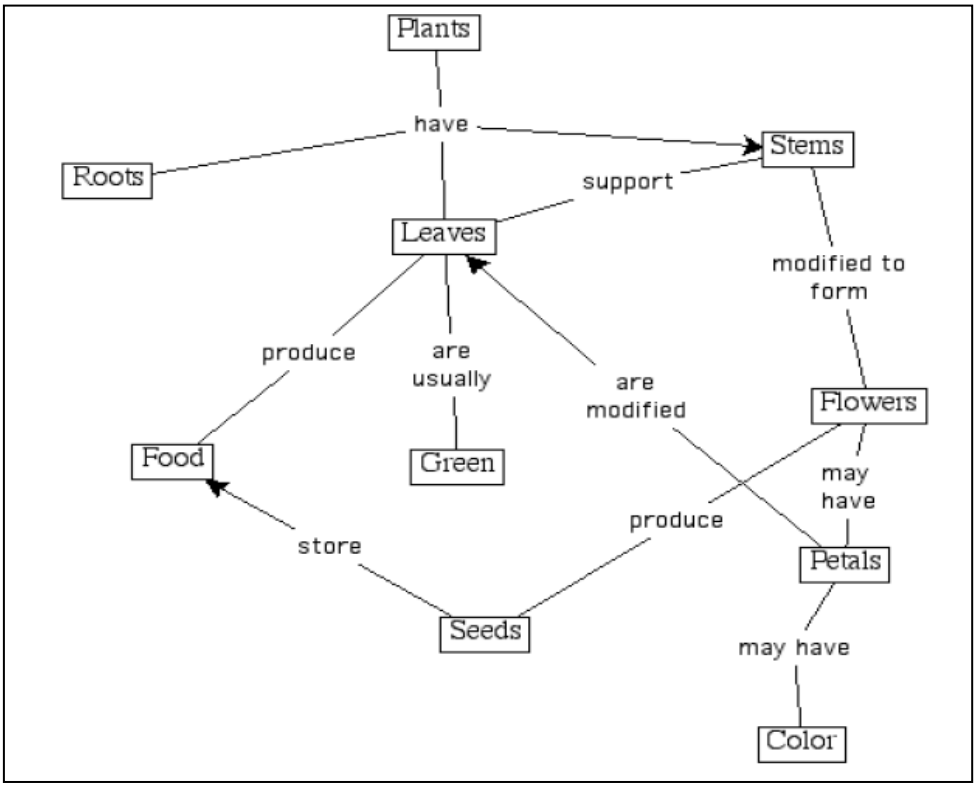


Figure 2.1 What Is a Plant Concept Map [10]

2.1.2 Why Use Concept Maps to Define User's Interests?

Knowledge extraction tools like search engines, RSS aggregators and others rely on keywords to define knowledge interests of users. Limited number of allowed keywords challenge the user to express his/her interest in the shortest possible sentence. Users many times fail to select good keywords that describe their interest or may even overlook some terms that can distinguish their interests from irrelevant domains of knowledge. Furthermore, search engines like Google and RSS aggregators like RSSOwl don't allow the user to personally assign different levels of importance to different keywords, rather, keywords weights are allocated based on algorithms that consider the behavior of millions of users on

the web. Given this problem with keywords, previous research attempted to provide tools that help users better express their knowledge interests for more effective retrieval of relevant knowledge. first attempts were in the field of restructuring the user's query in order to eliminate the problem that many people face in selecting good query terms. In 1999 Yukio [2] proposed a "Supplying Keywords System" that divided the user's query into sub-queries (i.e. terms making up the query) in order to enhance results returned by a search engine. Sub-queries were used to retrieve relevant documents then documents were compared. Sub-queries that generated most common documents with the highest relevant word frequencies became the user's final restructured query that represent the user's actual interest. The proposed method removed bad keywords from the query. Other attempts were made in the field of query restructuring. However, restructuring the user's query in all its flavors works on a bad query in the first place. It can be seen as a corrective rather than a protective solution. Other research attempted to start with the user by guiding the user in defining good queries from the beginning. Since concept maps and mind maps proved to be good tools for representing knowledge in educational context, [3], [4] and [5] proposed the use of a graph to express knowledge interest of specific users. Using a graph, users can completely and thoroughly define their knowledge interest without having to squeeze it into ten-words query or less. In addition, graphs have a hieratical representation which indicates the importance of different concepts as conceived by specific users.

2.1.3 Concept Maps Topology

A fundamental assumption shared by almost all knowledge organization theories dictates that knowledge can be modeled in terms of set of concepts and their relationships. Concept mapping is a method for making such concepts and relationships within individual's brain explicit. Starting from the idea that top-level concepts in the map are the most inclusive and bottom-level concepts are the most exclusive, research started to consider the layout and topology of the concept map in defining the user interest instead of relying only on the concepts and their relationships. [4] proposed that the closer two concepts in the graph the closer their relationship will be and hence the more similar their search results should be. [11] and [12] studied further concept map's structure influences and considered incoming and outgoing connections, and proposed 3 models that help assigning structural or topological weights to every concept in the map. In this research these topological models will be

considered. Connectivity Root-Distance (CRD), Path Counter (PC) and Hub Authority and Root-Distance Model (HARD) defined in [11] and [12] are explained later.

2.1.3.1 Connectivity Root-Distance Model (CRD)

The connectivity root-distance model is based on two observations. First, concepts that connect to more concepts are considered more important than concepts with lower connectivity. Second, Novak [10] argues that concept maps are best constructed if a focus question guides the selection of concepts and their hierarchical organization in the map. Therefore, the root concept, typically located at the top of a map, tends to be the most inclusive concept to the topic of the map. This suggests that concepts closer to the root are more important. The CRD model determines closeness to the root by counting the number of direct links between the map's root concept and a given concept. If concept ***k*** has ***o*** outgoing and ***i*** incoming connections to other concepts and is ***d*** steps distant from the root concept of the map, then the weight assigned to ***k*** by the CRD model is

$$W(k) = \left(\alpha \,.\, o(k) + \beta \,.\, i\,(k)\right).\left(1/(d(k) + 1)\right)^{1/\gamma 2}$$

The model fixed parameters α, β, γ determine influence of the incoming connections, outgoing connections, and distance to the root concept. The formula implies that the higher a concept's connectivity and the shorter its distance to the root concept, the larger its weight and therefore relevance to the topic of the map.

2.1.3.2 Path Counter Model (PC)

The Path Counter Model, like the CRD model, reflects the expectation that concepts participating in more propositions will tend to be more important to the topic of a map. However, instead of considering only a concept node's immediate connectivity the PC model considers indirect relationships as well. It counts all possible paths, starting from the root concept, that contains the concept in question and end directly or indirectly on the concept under examination. Formally, to determine the weigh ***W (k)*** of a concept ***k*** in a map, assume that ***n*** is the number of paths that lead to ***k*** from the root concept, then the weight is computed as ***W (k) = n***.

2.1.3.3 Hub Authority and Root-Distance Model (HARD)

While CRD performs a local analysis, only taking immediate neighbors into account, HARD performs a global analysis on the influences of the concepts on each other. Its analysis centers on three different types of concepts that may be found in a concept map:

- *Authorities* are concepts that have multiple incoming connections from hub nodes.
- *Hubs* are concepts that have multiple outgoing connections to authority nodes.
- *Upper nodes* include the root concept and concepts closest to the root concept.

To determine a node's role as a hub or authority**,** [11] adapted Kleinberg's algorithm [13] for analyzing hyperlinked graphs to concept maps. The adapted algorithm associates each concept with three weights between 0 and 1, each reflecting the concept's role as a hub, authority, or upper node. A given concept may simultaneously have properties of all three. In the HARD model, the three weights of a selected concept ***k*** are combined into a single weight as follows:

$$W(k) = \alpha \,.\, h(k) + \beta \,.\, a(k) + \gamma \,.\, u(k)$$

In the above formula ***h, a, u*** are the corresponding authority, hub, and upper node weights of a concept in a map and α, β, γ are the model fixed parameters.

2.1.4 Related Work In Utilizing Concept Mapping In Knowledge Capturing

In 2001 [11] proposed an algorithm to calculate the position weights of different concepts in the concept map. The algorithm was proposed based on the assumption that topological analysis of concept maps can be used to describe the relative arrangement of their concepts. According to their topological analysis of concept maps they described four different types of nodes:

1) ***Authorities*** are nodes having the largest number of incoming links coming from "hub nodes"

2) ***Hubs*** are nodes having the largest number of outgoing links ending at authority nodes.
3) ***Upper Nodes*** are nodes close to the root in the top of the concept map.
4) ***Lower Nodes*** are nodes appearing in the bottom of the concept map.

Each concept in the concept map is associated with four position weights, a-weight, h-weight, u-weight and l-weight. The weights represent to which kind of nodes each concept belong and hence describes the position of the concept in the map relative to other concepts. Once these weights are calculated they remain static unless the topology of the concept map is changed.

[12] proposed four models for analyzing concept map topology using the position weights proposed earlier. The models assigned overall weight for each concept in the concept map based on the concept's position, number of incoming links and number of outgoing links. The concept's overall weight reflects the concept's importance to the main topic of the concept map. The first model, the ***baseline model***, assumed that concept map topology has no influence on the importance of concepts in the map. The other three models:

1) *Connectivity Root-Distance Model (CRD)*
   This model relies on two observations. First, concepts with more incoming and outgoing links are assumed to be more important. Second, concepts close to the root are more important than lower concepts.

2) *Hub Authority and Root-Distance Model (HARD)*
   HARD performs global analysis of nodes by considering hub nodes which have multiple outgoing links ending at authority nodes, authority nodes which have multiple links coming from hub nodes, and upper nodes including root concept and concepts close to the root.

3) *Path Counter Model (PC)*
   This model counts all possible paths starting from root concept and ending at a concept either with no outgoing connections or has already been visited in the path.

A human-subject experiment was conducted to fit the four models predictions to human judgment. All models except the baseline model agreed with more than 67% of the decision reached by the participants which supports assumption that concept's position and number of connections influence its importance to the topic of the concept map.

[3] studied how to automatically generate good search queries from a concept map. They proposed the following steps:

1) Analyze different topological influences on the concept map using CRD , HARD and PC Models in order to qualify concepts with high topological weights.
2) Analyze combination of concepts and linking phrases to arrive at the most effective terms to use in the initial query. As a result, concept labels were found to be more effective than linking phrases in retrieving sources with good coverage for the topic.
3) Use the initial query generated from the concept map with the terms selected in the previous 2 steps to search for the topic in Google. The first 30 search results were then examined in order to extract discriminators for the concept map.
4) Form new query with discriminator and descriptor terms generated in the previous steps and submit it to Google for more precise results.

The returned results of the suggested query-refinement method were compared to search results of Inverse Map Frequency (IMF) method. The comparison showed that the suggested query-refinement method using concept map outperformed the IMF method (64% vs. 36%).

[5] studied how to formulate search queries that reflect the context of the user's knowledge model and how to automatically suggest topic descriptors and discriminators for the same knowledge model. In order to automatically suggest good descriptors and discriminators for a given knowledge model, the algorithm was applied to EXTENDER (i.e. a concept mapping tool) to provide suggested terms that returned documents related to a user's predefined concept maps. By analyzing the suggested terms and evaluating them with the proposed algorithm; good topic descriptors and discriminators could be automatically generated to the user. Test cases showed that the proposed method helped refining the search query of the user and generate knowledge that is related to the knowledge context in question.

[4] studied how to align search results returned from search engine to user's mind map using the optimization algorithm simulated annealing. This research attempted to address the problem that, Google and most search engines lack semantics in their current ranking and indexing algorithm. Search engines like Google rank pages and links based on millions of users. Hence individual thinking model is not taken care of when conducting a search on the

web. The study utilized a mind map to define user's concept. Subsets of user's concepts were generated from the main graph. Queries based on the subsets of the mind map were used to generate set of responses. After that, response sets were aligned with concepts in the map according to the following assumption; "if the distance between a concept **A** and concept **B** is less than the distance between concept **A** and another concept **C** in the mind map, then an alignment dictates that the distance between the response sets of **A** and **B** should also be less than the distance between the response sets of **A** and **C**." [4:154] Experimental results showed great enhancement in the precision of results retrieved by Google after the alignment. However, The suggested alignment method consider distances in the concept map and doesn't take into account the connectivity factor of different nodes in the graph.

[3],[4] and [5] proved the effectiveness and high precision of using concept mapping in defining knowledge interests and that concept mapping could be a better replacement of simple query keywords. Based on these findings this research is employing concept mapping for defining knowledge interests of specific users. [3] and [4] used concept mapping for refining search results after conducting a regular search using a search engine like Google whereas [5] used concept mapping to suggest keywords that can be used in regular search as well. However, in this research concept mapping in used in Just-in-Time knowledge search. Hence, in this research knowledge is not searched for using a search engine, rather, knowledge related to the concept map is instantly retrieved from RSS feeds.

2.2 Related Work in Fuzzy Set Information Retrieval Model and RSS Feeds

[14] proposed a method for retrieving documents that match a user query based on fuzzy set information retrieval model. They came up with the idea of *keyword connection matrix.* The keyword connection matrix was composed of keywords and their relationships , where relationship values represent the conceptual similarity between every two words in the matrix. Keyword connection matrix was represented by a **K x K** matrix where **K** is equal to the number of the keywords. The relationship values were restricted to the interval [0,1]. The relationship values were assigned based on the assumption that the more two keywords appeared together in one document the more related to each they were. Relationship between every two keywords in the matrix was calculated according to the following formula:

$$W_{ij} = \frac{N_{ij}}{N_1 + N_j - N_{ij}}, \; i \neq j \qquad (2.1)$$

Where N_{ij} is the number of documents containing both the i-th and the j-th keywords, and Ni and Nj are the number of documents including the i-th and the j-th keywords respectively. This research provided an implementation of fuzzy logic in retrieving relevant documents, especially that fuzzy logic was very appropriate for dealing with ill-defined information. However, relying on the concept of co-occurrence alone to determine similarity or relationship between words is sometimes inaccurate. For example if we consider a document that talks about computer architecture the keyword connection factor between "computer" and "architecture" would be high. In another document that talks about how architects use computers in their design again the keyword connection factor will be high as both words "computer" and "architecture" would appear in the same document quite often. And both factors calculated for each document might be the same which is not accurate and doesn't reflect the actual relationship between the two keywords. Another measure of similarity that considers not only the co-occurrence but also the distance between every two keywords in a given document can provide more accurate measure of similarity.

[15] proposed a method to detect similar HTML documents based on fuzzy set IR model. The proposed method started by processing the HTML documents in order to remove HTML tags and extract text from the documents. Then stop words were removed from the text and stemming algorithm [16] was applied. A word-word correlation matrix was built based on a representative corpora using word-connection correlation factors. Finally, stemmed documents were compared at two steps. First, word-sentence comparison was conducted using the correlation factors in the word-word matrix and a word-document similarity measure was computed. Second, document-document comparison was conducted to find out the overall similarity between every two documents. Using a similarity threshold and a variation threshold similar documents could be detected. This research provided a useful tool for organizing web contents and reducing redundancy. One problem with the method is the use of keyword connection matrix which rely on co-occurrence only to determine words similarity. Further improvements on top of this work can be done to consider different types of web documents such as XML which is becoming more widely used on the web.

[17] studied filtering RSS feeds and suggested a method that helps filtering RSS feeds without having to go through all incoming feeds. To illustrate the proposed method; [17] developed a Smart RSS Aggregator (SRA). Filtering method consists of the following steps:

1) *Pre-Filtering*: user profile is created specifying the search interests of the user. Accordingly relevant RSS feeds are defined for search using Feedster.com. As new RSS feeds are launched to the web that relates to the user's profile, they are added to the SRA.
2) *Filtering Process and input semantic*: users are prompted for filtering keywords that represents their search interests. [17] proposes a semantic structure for defining filtering keywords made up of one verb/adjective phrase, one noun phrase and one-to-many auxiliary phrases. The noun and verb/adjective phrases are considered to be the main focus of the user search. By accessing Synonym.com, synonyms for the verb/adjective and noun phrases are generated. Combination of the user's entered keywords and the generated synonyms are created to form multiple search queries.
3) *Post-Filtering*: contents of retrieved RSS feeds are searched against the generated semantics.

According to [17], in a test case the proposed method improves retrieved RSS relevance up to 44%. Nevertheless, Using linguistic synonyms is sometimes misleading and might result in irrelevant key words especially in the verb/adjective field. The examples illustrated by the researcher resulted in excellent results as the adjectives/verbs which he used assumed quite little synonyms. However, considering the word "worn" the past participle of wear and an adjective as well. Using Synonym.com (i.e. the same synonyms database used in SRA), we can get up to 23 adjective synonyms and more than 25 verb synonyms that would actually entirely change the meaning of the full search statement. Furthermore, some verbs consist of two words (e.g. take off, hand over, take over ... etc), where the verb/adjective field in SRA accepts one word only.

[18] And [19] studied how to use word-correlation factors in a fuzzy set information retrieval model to filter out unnecessary news articles from RSS feeds, and discard less-informative articles from the non-redundant ones. Initially a word-word correlation matrix was built based on a representative corpora using distance correlation factors. The clustering process starts by concatenating the title and the description of a given RSS feed and

comparing word to sentence using the correlation factors calculated earlier in the word-word matrix. Then computing the similarity measure between every two feeds in the RSS feed in order to reach at a set of redundant feeds (i.e. feeds with lower correlation factors) which are then discarded. According the researchers' experiments, the method outperform other clustering methods and achieves high accuracy up to 80%. The proposed method is efficient and requires little computations as the keyword-correlation factors are pre-computed and the clustering algorithms are computationally inexpensive. It can be further enhanced by considering semantics and incorporating users' interests.

[20] studied how to restructure knowledge using Fuzzy Formal Concept Analysis Theory. They proposed a framework for RSS Feed structuring in order to improve the RSS feed retrieval. The framework consists of the following parts:

1) *RSS-Feed Parser:* collected RSS Feeds are analyzed in order to construct an object-attribute matrix (i.e. context matrix) according to the Formal Concept Theory. This matrix shows the relationship between a keyword and a domain of knowledge in terms of degree values.
2) *Concept Hierarchy Producer:* the previously generated context matrix is now converted into a corresponding concept lattice on the basis of the Formal Concept Theory.
3) *Ontology Builder*: the produced concept lattice is now converted into ontology network of concepts in order to reveal knowledge structure and RSS Feed categorization. Human intervention may help at this point to validate the generated concept network.

This approach has been validated on a collection of RSS Feeds coming from OpenLearn Project. A sample of 443 RSS Feeds has been defined. The concept hierarchy builds a lattice composed of 193 concepts. After analyzing the lattice, the researchers find that most of the RSS feeds are categorized in the right categories. By comparing the grouping with OpenLearn's groups, only 13% of the lattice is found to be misclassified. This research enhances the retrieval of knowledge on the web through ontology-based categorization. However, at some point of the process human intervention is required for validation. On the other hand, the research takes into account all the possible ontologies without focusing on specific user's interest. It can be improved further by incorporating individual user's thinking concepts.

[14] and [15] introduced the use of fuzzy set information retrieval model and used word connection factors to build up the fuzzy sets. As explained earlier the word connection factor suffer from low accuracy compared to the distance correlation factors developed by [18] and [19]. All of these researches build the fuzzy thesauri on top of a common text corpora like Wikipedia. This research builds upon the outcomes of [14],[15],[18] and [19] in using fuzzy set information retrieval model using the distance correlation matrix for the creation of the fuzzy set. Furthermore, this research introduces the idea of custom text corpora which is built in linkage with the user's concept map. Custom text corpora is proposed to provide more precise results.

CHAPTER 3

FUZZY SET INFORMATION RETRIEVAL MODEL

Before proceeding with the explanation of the Fuzzy Set IR Model, it's important to go through an overview of information retrieval methodologies. Section 3.1 of this chapter introduces Information Retrieval - IR as a science and presents most common types of IR methodologies. Fuzzy Set IR Model is presented in section 3.2 of this chapter.

3.1 Information Retrieval – IR

The process of searching for relevant information from documents' corpora has always been a big concern for people in all fields. As an academic field of study, information retrieval (IR) can be defined as:

'*finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).*'[21:1]

In other words, it is the science of searching for information within documents of different types. Computerized documents and digital search tools gave large hopes and put big challenges in the field of IR. The idea of using computers to search for relevant pieces of information was popularized in the article "***As We May Think"*** by Vannevar Bush in 1945. Several key developments in the field happened in the 1960s. Most notable were the development of the SMART method by Gerard Salton and his students which allowed researchers to experiment with ideas to improve search quality [22]. The main components of any IR method are the user query, the matching algorithm, i.e. processor, document collection and user interface as shown in figure 3.1 [23] . Further components can be added to enhance the IR method and provide more features such as ranking.

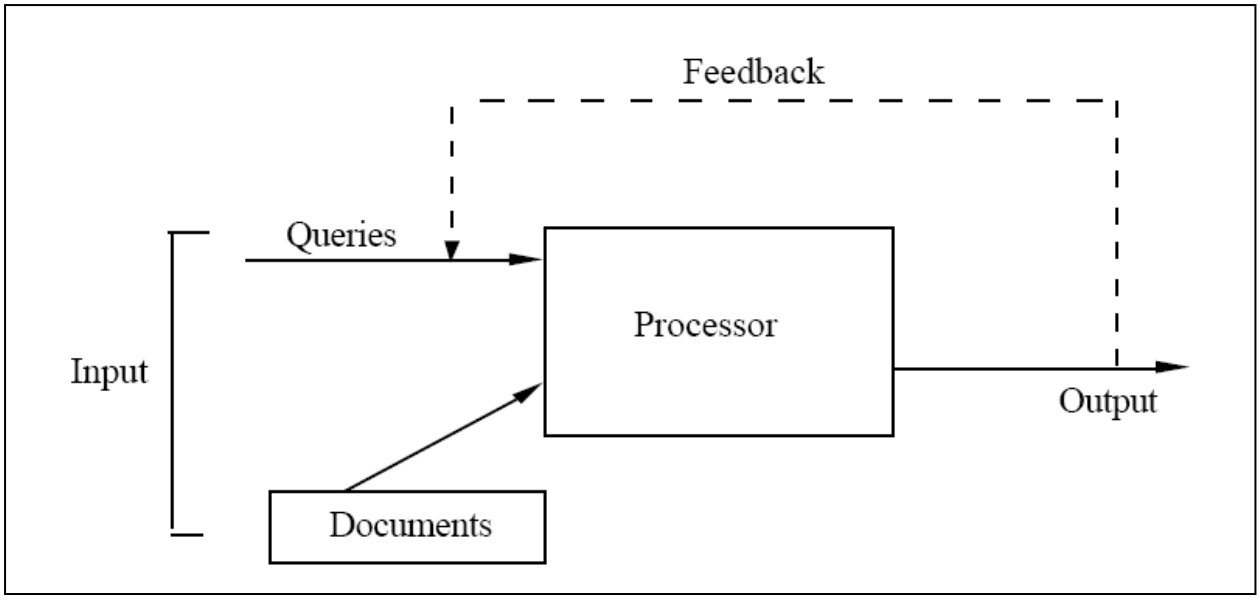


Figure 3.1 Information Retrieval Process [23]

Therefore, the most standard objective of any IR method is to provide a set of documents for the user from a document collection relevant to arbitrary user's information need. Documents are considered relevant if they are perceived to be containing information of value with respect to the user's information need. To assess the effectiveness of an IR method, user normally need to know two statistics about the method's returned results [21]:

1) **Precision:** What fraction of the returned results are relevant to the information need? It's calculated as in 3.1.

Precision = # (relevant Items Retrieved) / #(Retrieved Items) (3.1)

2) **Recall:** What fraction of the relevant documents in the collection were returned by the method? It's calculated as in 3.2.

Recall = # (relevant Items Retrieved) / #(Relevant Items) (3.2)

Different IR methodologies were developed and proved different levels of effectiveness in different applications. Most common algorithms and the root of other methodologies are Vector Space, and Boolean Retrieval. A brief overview of the two algorithms is presented in Section 3.1.1 and Section 3.1.2.

3.1.1 Boolean Retrieval

In this type of retrieval, only documents that are true for the query are returned. Therefore, query is viewed as a set of keywords with connecting logical operators such as AND, OR and Not. Documents are viewed as set of keywords as well [1]. The query is processed using inverted index file which is built for the documents' collection in advance. For each keyword in the query, the index is searched and the corresponding posting for the term is retrieved. Posting is a set of documents in which the respective term occurs [21]. Once all the postings for the terms in the query are retrieved, they are merged based on the operator given in the query. In Boolean retrieval, we deal with the exact match. Some methods, which operate by means of Boolean search, give the user access to a structured dictionary which, for any given term, stores related keywords which can narrow or broaden the search query.

3.1.1.1 Example On Boolean Retrieval

Consider a small document collection of three documents**:**

Doc1	Nokia has mobile applications
Doc2	Symbian is Nokia platform
Doc3	Symbian and Anroid are Mobile platform

The inverted index for the collection is shown in the table 3.1 below, sort-based indexing is used for building the index, a common technique in which the terms are sorted and grouped to build the index. The document frequency of each term is also stored on the index. This information is used to minimize the amount of temporary memory space during query processing.

Table 3.1 Inverted Index File For The Document Collection

Term	Document frequency	Posting list
And	1	3
Anroid	1	3
Applications	1	1
Are	1	3
Has	1	1

Term	Document frequency	Posting list
is	1	2
Mobile	2	1,3
Nokia	2	1,2
Platform	2	2,3
Symbian	2	2,3

In response to the query*: Anroid AND Symbian* Using the inverted index file created for the three documents earlier, posting for the first term (Anroid) is retrieved and intersected with posting of the second term (Symbian), such that {3} AND {2,3} resulting in {3}, therefore using Boolean retrieval the query returns document number 3 that contains both terms.

3.1.1.2 Problems in Boolean Retrieval

Boolean retrieval has some problems. The user has to have good knowledge on the search topic for the search to be efficient, e.g., a wrong word in a query could rank a relevant document non relevant. The retrieved documents are all equally ranked with respect to relevance and the number of retrieved documents can only be changed by reformulating the query.

3.1.2 Vector Space Model (VSM)

Vector space model (VSM) computes a measure of similarity by defining vectors for each document in the collection and for the query. Traditionally to measure the closeness of two vectors, the size of the inner angle is calculated. Relevance between the document and the query is measured by using the dot product of the query's vector and the document's vector. Often the expression "similarity measure" is used instead of inner angle. Other measures of similarity were also defined to evaluate the relevance of documents to a query such as the Cosine coefficient, the Dice coefficient and the Jaccard coefficient [1], [21]. Term weights used to indicate different importance level of different term are explained next.

3.1.2.1 Term Weights

Instead of simply specifying a list of terms in query, a user is given the opportunity to indicate that one term is more important than the others. Initially weights were assigned manually by the user. Later on, it became possible to assign weights automatically according to the term frequency across the entire collection. Based on collection frequency, weight is computed using the ***Inverse Document Frequency*** corresponding to a given term. To construct a vector corresponding to each document, consider the following definitions:

t = number of distinct terms in a collection of document

$\mathbf{tf_{ij}}$ = frequency of term j in document i

dfj = frequency of documents containing term j

$\boldsymbol{idf_j}$ = log (***d/dfj***) , ***d*** is total number of documents , $\boldsymbol{idf_j}$ is the inverse document frequency.

Document weight is then computed using a combination of term frequency and inverse document frequency, such that: Term Weight: $\boldsymbol{dij} = \mathbf{tf_{ij}}$ X $\boldsymbol{idf_j}$

As document weights play an important role in retrieving relevant documents and enhancing the effectiveness of the algorithm, many variations of term weights were studied to improve on the basic combination of **tf**-***idf***.

3.1.2.2 Example on VSM retrieval.

Consider a small document collection of three documents**:**

Doc1	Nokia has mobile applications
Doc2	Symbian is Nokia platform
Doc3	Symbian and Anroid are Mobile platform

In response to the query: *Symbian Mobile Applications*, using $\boldsymbol{idf_j}$ = log (***d/dfj***) , we calculate the inverse document frequency for all the terms in the document collection as in table 3.2:

Table 3.2 Inverse Document Frequency

Term	***idf***
And	0.477
Anroid	0.477
Applications	0.477
Are	0.477
Has	0.477

Term	***idf***
is	0.477
Mobile	0.176
Nokia	0.176
Platform	0.176
Symbian	0.176

Now document vectors of terms and their associated weights can be constructed. Weights are calculated as $\mathbf{tf_{ij}}$ X $\boldsymbol{idf_j}$ in table 3.3.

Table 3.3 Document Vectors with Weights

docID	**And**	**Anroid**	**Applications**	**Are**	**Has**	**Is**	**Mobile**	**Nokia**	**Platform**	**Symbian**
D1	0	0	0.477	0	0.477	0	0.176	0.176	0	0
D2	0	0	0	0	0	0.477	0	0.176	0.176	0.176
D3	0.477	0.477	0	0.477	0	0	0.176	0	0.176	0.176
Q	0	0	0.477	0	0	0	0.176	0	0	0.176

Now find the similarity between each document and the query by computing the dot product of their corresponding vectors with the associated weights.

SC(Q,D1) = (0.176x0.176) + (0.477 X 0.477) ≈ 0.258

SC(Q,D2) = (0.176x0.176) ≈ 0.031

SC(Q,D3) = (0.176)2 + (0.176)2 ≈ 0.062

Hence, the relevance ranking of the 3 documents would be D1,D3,D2

3.1.2.3 Problems in VSM

Vector space model overcomes some problems inherent in the basic Boolean retrieval algorithm by assigning weights to the terms in the query and in the documents. This in turn allows for ranking different keywords differently. However, weights assigned in the VSM are not so formal and to some extent can be considered intuitive. Furthermore, terms are considered independent and no clear associations between terms are defined.

3.2 Fuzzy Set Information Retrieval Model

Fuzzy Set IR model is adopted to determine the degree of membership between every keyword in a sentence and a fuzzy set that contains different words each belongs to the set at some degree of membership. The degrees of similarity or membership, also referred as the correlation factors among words, are given by a function which assigns a value in the range [0, 1] to any two words . There are several methods to define the correlation factors among different words. To better understand the Fuzzy Set IR model let's first go through an overview of Fuzzy Set Theory.

3.2.1 Fuzzy Set Theory

Conventional set or crisp set [24] is any collection of objects which can be treated as a whole. Cantor described a set by its members, such that an item from a given universe is either a member or not.

Example:

let A be the set of non-negative numbers less than 4.

Hence, $\boldsymbol{A}$ = { 0,1,2,3}

$\boldsymbol{A}$ is a finite crisp set. That's numbers either belong or don't belong to it. Infinite crisp set can be represented with a property that characterizes the elements in the set.

Fuzzy Set [25] contains items that belong to the set at different grades of membership. Following Zadeh, many sets have more than an Either-Or criterion for membership. Take for example the set of Young people. A one-year old baby will clearly be a member of the set, and a 100 years old person will not be a member of this set, but what about people at the age of 20, 30, or 40 years? Another example is a weather report regarding high temperatures, strong winds, or nice days. Zadeh proposed a *Grade of Membership*, such that the transition from membership to non-membership is gradual rather than abrupt. An item's grade of membership is normally a real number between 0 and 1, often denoted by the Greek letter **μ**. The higher the number, the higher the membership . Zadeh regards Cantor's set as a special case where elements have full membership, **μ = 1**. He nevertheless called Cantor's sets non-Fuzzy ; today the term crisp set is used, which avoids that little dilemma. Zadeh does not give a formal basis for how to determine the grade of membership. The membership for a 50 year old in the set Young-People depends on one's own view. The grade of membership is a precise, but subjective measure that depends on the context. A membership function $\boldsymbol{fA}: \boldsymbol{C} \rightarrow [\mathbf{0}, \mathbf{1}]$ is used to indicate the degree of membership of each element in the set. Therefore, a fuzzy set A on C can be represented as:

$A = \{ (C1, fA(c1)), (C2, fA(c2)), \ldots\ldots , (Cn, fA(cn)),\}$

3.2.2 Operations on Fuzzy Set

Basic operations of intersection, union, and complement on fuzzy set are described below. Essentially the intersection uses minimum of two membership functions, for the same element. The union uses the maximum of two membership functions for the same element and the complement subtracts each value in the set from 1 [1].

$$fA \cap fB(Ci) = Min\left(fA(ci), fB(ci)\right), \forall\, ci \in C \qquad (3.3)$$

$$fA \cup fB(Ci) = Max\left(fA(ci), fB(ci)\right), \forall\, ci \in C \qquad (3.4)$$

$$fA'(Ci) = 1 - \left(fA(ci)\right), \forall\, ci \in C \qquad (3.5)$$

3.2.3 Defining The Fuzzy Association and Membership Function

As explained in section 3.2.1 the fuzzy set theory relies on two main principles: (1) sets are not well defined, and (2) elements belong to the fuzzy set at different levels of membership. English sentences and documents are a typical examples of fuzzy sets. Every term in a sentence is associated with a fuzzy set that contains words similar in meaning and there is a degree of similarity (i.e. less than one) between the term in the sentence and the set. Hence, if two sentences contain many terms that belong to the same fuzzy sets at high degree of membership then the two sentences are similar.

[14] adopted a fuzzy set IR model to determine whether a keyword in a sentence belongs to a fuzzy set that contains words with different levels of similarities amongst themselves. [14] called the fuzzy set a ***keyword-connection-matrix*** and defined it as a kind of thesaurus (i.e. dictionary) which described relations between keywords by assigning similarity grades restricted to the interval [0,1]. Later [15] used the same keyword-connection-matrix proposed by [14] to detect similar HTML documents. Using the keyword-connection-matrix [15] compared every keyword ***k*** in sentence ***i*** with every keyword ***w*** in document ***d*** and calculated a word-sentence similarity using the following fuzzy association:

$$\mu_{k,d} = 1 \_ \prod_{w \in d}(1 - C_{kw}) \qquad (3.6)$$

where C_{kw} is the fuzzy relationship between ***k*** and ***w***. The average of all μ-value is calculated to find the overall similarity between sentence ***i*** and document ***d*** as follows:

$$\text{Sim}\,(i, d) = \frac{\mu_{k1,d} + \mu_{k2,d} + \ldots + \mu_{n1,d}}{n} \qquad (3.7)$$

Where **n** is the number of keywords ***k*** in sentence ***i***.

In 2008 [18] and[19] adopted the same fuzzy set IR model adopted by [14] and [15] to cluster similar RSS feeds. However, [18] and[19] didn't rely on the keyword-connection-matrix only to define the fuzzy membership or the correlation factor, instead they used three different types of correlation factors: (1) word Connection, (2) word co-occurrence, and (3) distance. Their experiments proved that the correlation factors in the distance matrix provided the most accurate results amongst all the three matrices with accuracy rate of 94% compared

to 47% for the keyword-connection and 52% for the Co-occurrence [26] . That is because distance correlation factors accounts for frequency and co-occurrence at the same time.

3.2.4 Advantages of Using Fuzzy Set IR Model

Fuzzy set information retrieval model provides solutions for problems inherent in other IR methods such as VSM and Boolean models as explained earlier. It proved effectiveness in dealing with ill-defined information. Furthermore, fuzzy set IR model proved to work well on partially related semantics and similar sentences along with exact match. Our Implementation of fuzzy set IR model with concept maps to retrieve knowledge from RSS feeds provides a working example on fuzzy set IR. It will be explained in chapter 4 of this thesis.

CHAPTER 4

COMBINING CONCEPT MAPS WITH FUZZY SET IR MODEL

In chapter 2 concept maps and how they can be used to describe user's interests was introduced. Fuzzy-Set IR Model was introduced in chapter 3. This chapter explains how the structure of concept map with the logic of Fuzzy-Set IR Model can be combined to retrieve relevant knowledge from RSS feeds as shown in figure 4.1. In section 4.1 the process of generating the fuzzy set of words using distance correlation factors is explained. Section 4.2 explains the process of analyzing the structure of the concept map and calculating concept's weights. In section 4.3 the process of retrieving knowledge from RSS feeds using the fuzzy set and the concepts' weights is explained. The proposed processes were verified in a JAVA application that applies all the steps of the proposed method starting from creating the fuzzy dictionary and moving into defining specific RSS feeds and updating the user on relevant knowledge. Implementation is discussed in section 4.4.

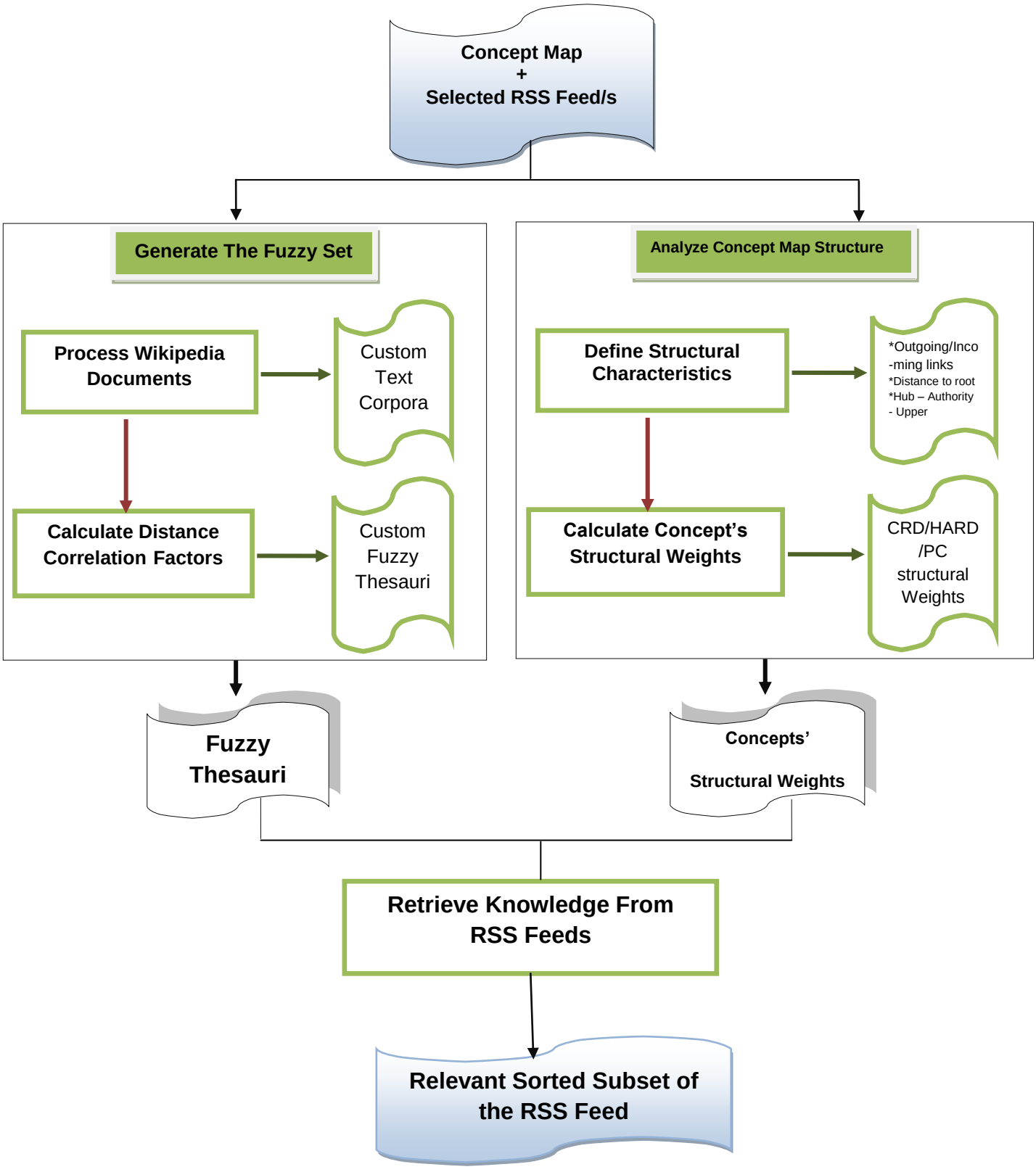


Figure 4. 1 High Level Definition of The Proposed Method

4.1 Generating The Fuzzy Set – Distance Matrix

As proposed by [14] and later implemented and applied by [15], [18] & [19], Fuzzy Set IR Model starts from fuzzy sets that contain keywords and their relationships. Relationship values ranging from 0 to 1 represent the conceptual similarity between two keywords. The conceptual similarity or the fuzzy relationship is also referred as ***correlation factors*** among words. There are many methods to calculate correlation factors such as keyword connection, co-occurrence and distance matrices. In this research the distance matrix is used to calculate the correlation factors in the fuzzy set since it's proved in [26] to be the most efficient method over others.

Starting from the same concept as proposed by [14], [15], [18] and [19], a set of representative English documents was required to create the distance correlation matrix and hence generate the fuzzy set of keywords. Looking for an unbiased set of documents in terms of writing styles that's diverse in contents and regularly updated, Wikipedia was selected. Wikipedia is a free online encyclopedia, which contains around over 3 millions articles in English as of 2010 . Wikipedia contains almost all possible topics in different areas of study, is constantly updated, and is unbiased in terms of writing styles and authorship. With the use of the Wikipedia documents to generate the distance correlation matrix, we can obtain reliable similarity measures between different words according to their occurrences and distance in various documents.

Since in [15] the objective was to generally compare HTML documents, and in [18], [19] the objective was to cluster RSS feeds from different general topics; a comprehensive fuzzy dictionary that contained keywords related to all subjects was required. This research aims at providing specific knowledge to specific users where users define their knowledge interests using concept maps. Concept maps are built around a focus question as explained earlier. Therefore, in order to enhance the results of the proposed knowledge retrieval method, a ***custom distance matrix*** is developed for every single concept map. By doing this there will be less noise (i.e. *irrelevant words that specifically and uniquely belong to other fields*) in the distance matrix and hence it's more likely that irrelevant knowledge will be ignored and discarded. In section 4.1.1 the process of creating custom text corpora from Wikipedia is explained and in section 4.1.2 the process of calculating distance correlation factors and creating custom fuzzy thesauri is presented. In order to explain the proposed

process through a working example a concept map about "*What you need to know when you develop a mobile application?"* was developed with an actual mobile application developer who works on Symbian and uses Netbeens and Carbide C++. The concept map is shown in figure 4.2.

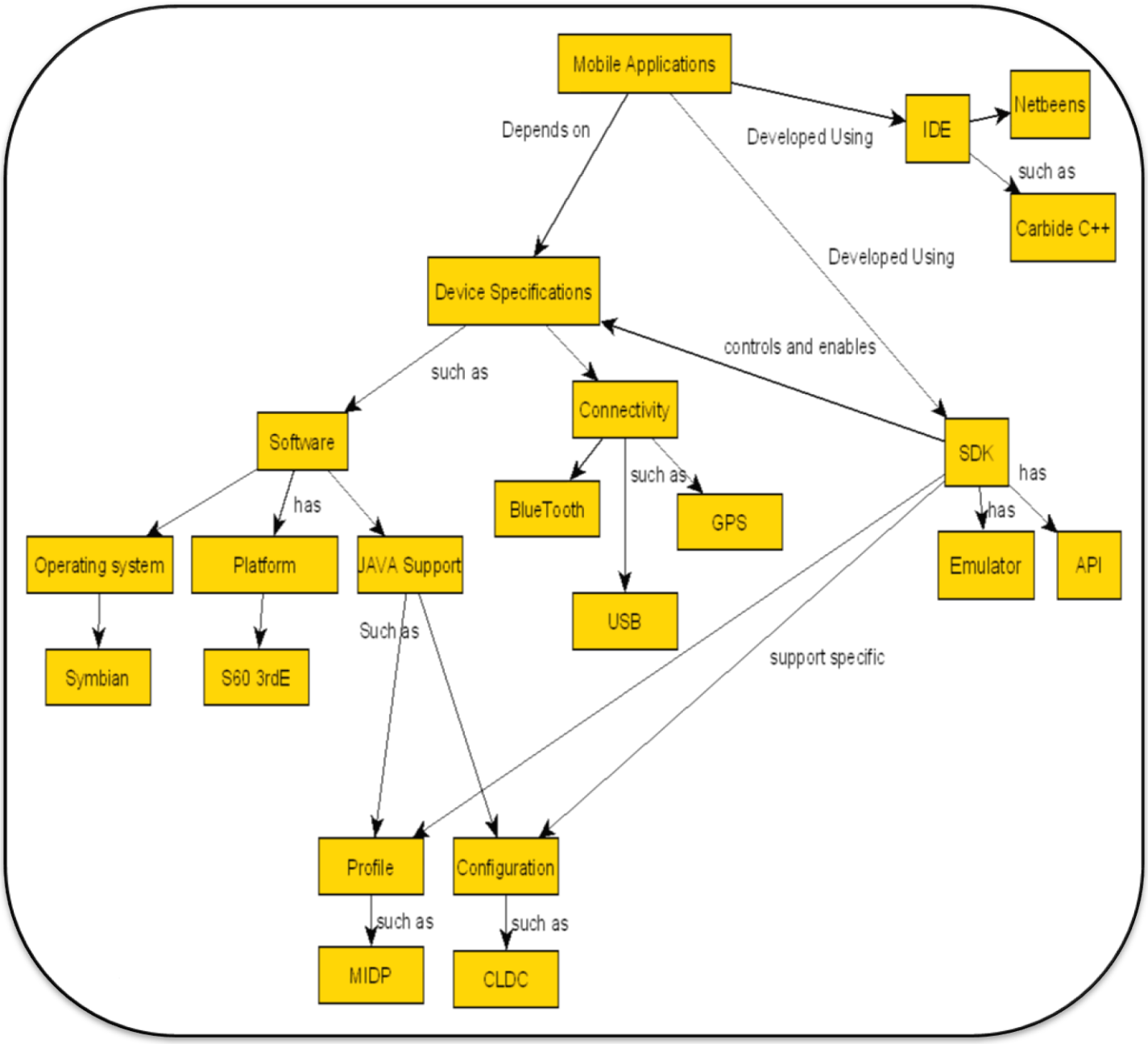


Figure 4. 2 Mobile Applications concept Map

4.1.1 Processing Wikipedia Documents – Create Custom Text Corpora

In order to create a custom fuzzy thesauri, a custom text corpora is required. The process of exporting Wikipedia articles specific to the user's concept map in order to create the custom text corpora is composed of the following steps as illustrated in figure 4.3 :

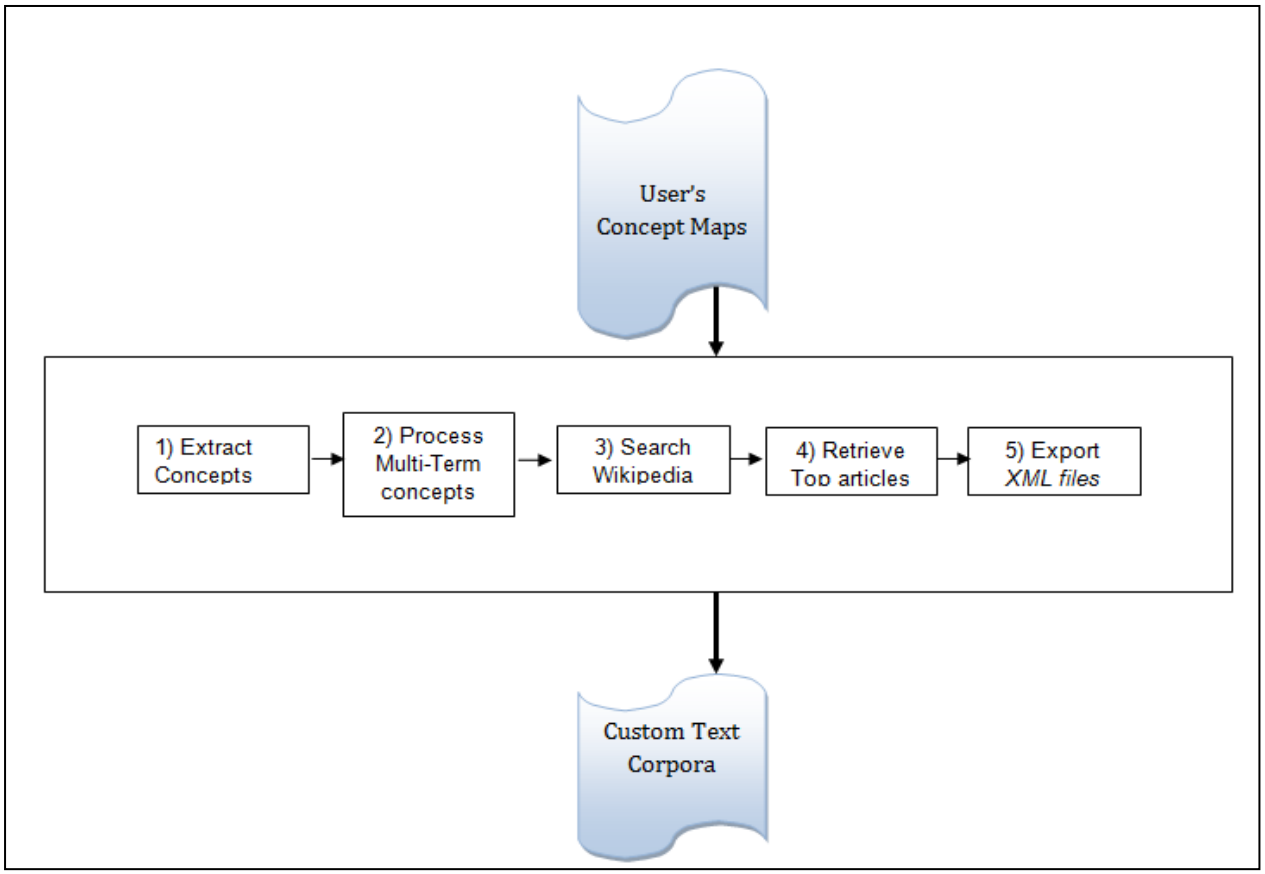


Figure 4.3 Creating Custom Text Corpora

1) Read every concept in the concept map
2) Perform text checking to deal with multi-term concepts. For example "Mobile Applications" is a two-terms concept. Later in section 4.4 implementation aspects are explained.
3) Use Wikipedia ***Search***[1] tool to search the encyclopedia for relevant articles.
4) Retrieve the top 1,2,3,4 or 5 articles' titles from Wikipedia search results.
5) Pass the retrieved titles to Wikipedia ***Export***[2] utility to retrieve corresponding full articles in XML format.

[1] ***http://en.wikipedia.org/w/index.php?title=Special:Search***

4.1.2 Calculating Distance Correlation Factors – Create Custom Fuzzy Thesauri

After a custom text corpora is obtained, it's processed in order to generate a custom fuzzy thesauri. The process of creating custom fuzzy thesauri is shown in figure 4.4.

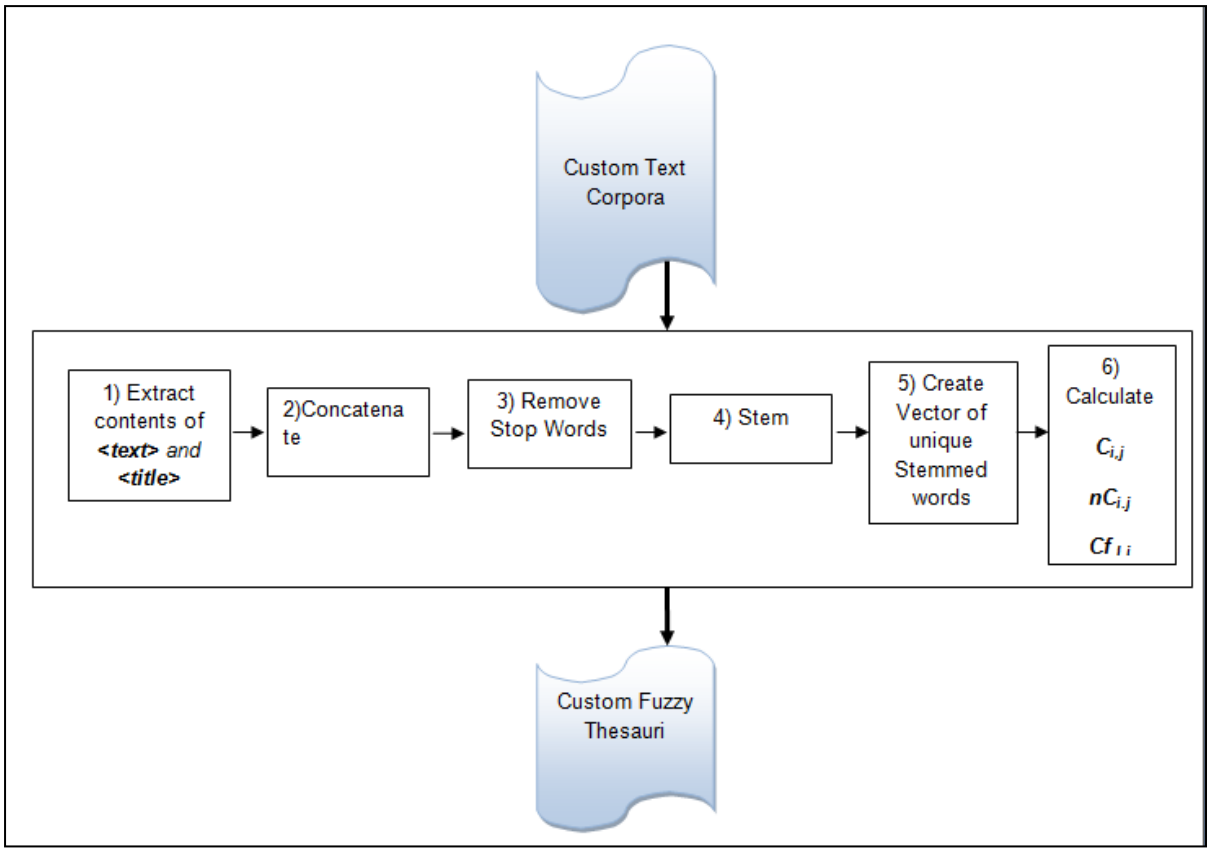


Figure 4.4 Creating Custom Fuzzy Thesauri

The Wikipedia documents resulted from the export process are then processed in order to extract the contents of the <*title*> and <*text*> tags from the corresponding XML document and concatenate them. Then, all common English StopWords that don't add to the meaning are removed such as propositions and common words (e.g. "The", "about", "above", "across", "after"… etc). Since words that are derived from a single root usually share a common meaning, similarly stemming is used here to group words that share a same stem into semantically related sets. The stem doesn't have to be the same as the root of the word; it is usually enough that related words map to the same stem, even if this stem is not by itself a valid root. Porter stemming algorithm was very commonly used and became the de-facto standard stemming algorithm used for automatic English language stemming. All articles are stemmed using Porter Stemming Algorithm [27]. The Porter stemming algorithm (or 'Porter stemmer') is a process for removing term suffixes such as -ED, -ING, -ION, TION.

[2] ***http://en.wikipedia.org/wiki/Special:Export***

Furthermore, the suffix removal process will reduce the total number of terms in the custom text corpora, and hence reduce the size and complexity of the data in the method, which is always useful.

Stemmed articles are then used to generate vector of all distinct stemmed words along with the documents they appear in and their positions in every document. For example "mobile" stem is "mobil", vector's entry for the word "mobil" would be as illustrated in table4.1.

Table 4.1 Sample Vector Entry

Term	Document ID	Position
Mobil	Doc1	1, 8, 20, 45
	Doc2	12, 30
	Doc3	7, 10, 27
	Doc4	30

Using the vector of distinct words, (i) the frequency of co-occurrence and relative distance in a single document, i.e., $\boldsymbol{Ci,j}$, (ii) the normalized value, i.e., $\boldsymbol{nC\ i,j}$, and finally (iii) the distance correlation factor, i.e., $\boldsymbol{Cf\ i,j}$, can be defined for every pair of keywords across all documents as follows:

$$C_{i,j} = \sum_{x \in V(wi)} \sum_{y \in V(wj)} \frac{1}{d(x,y)} \qquad (4.1)$$

$$nC_{i,j} = \frac{C\ i,j}{|V(wi)| \times |V(wj)|} \qquad (4.2)$$

$$Cf_{i,j} = \frac{\sum_{m=1}^{k} nCij}{k} \qquad (4.3)$$

where ***d(x, y) = |Position(x) - Position(y)| + 1*** is the distance, i.e., the number of words, between words ***x*** and ***y*** in a single Wikipedia document, V (wi) and V (wj) is the set of all stemmed variations of words wi & wj in a single Wikipedia document, |V (wi)| & |V

(wj)| is the number of words in V (wi) & V (wj) respectively, and ***m*** is the mth out of the ***k*** Wikipedia documents in which both wi and wj occur where ($1 \leq m \leq k$). As a result a matrix of all distinct stemmed words and their relationships i.e. ***cf,*** is constructed. This matrix is the custom fuzzy thesauri which will be used to measure the partial similarity and exact match between concepts in the concept map and terms in the RSS feed.

4.2 Analyzing Concept Map's Structure

Once the custom fuzzy thesauri is created to handle the semantic similarities, the structure of the concept map is analyzed to measure the relative importance of different terms in the user's concept map. As discussed in chapter 2 of this thesis, many studies [11], [12] has shown that the structure of concept map implies the relative importance of different concepts in the mind of the user of the concept map. Whereas other studies attempted to deploy this structural influences of concept maps to modify search queries [3], [5] and filter the results of search queries [4]. Using the outcomes of those researches, this research tries to incorporate concept's weights with the semantic similarity in order to ensure that retrieved knowledge from RSS feeds is the most relevant to the user of the concept map and hence increase the precision of the retrieved knowledge. The three topological models (i.e. CRD, HARD and PC) proposed by [11], [12] are used to calculate concept's weight. The three models were implemented since in [11] and [12] the assessment of the model was conducted on human subjects of different backgrounds. Although assessment was made objective by choosing common concepts not specific to a certain field of knowledge, still human subjects' differences and background influences can't be ignored. And hence, a particular model can't be nominated for this study. That's the three models will be used and tested later in chapter 5.

The analysis of the structural weights goes through two steps. First, the structural characteristics of each concept need to be defined as per the selected model. For CRD model, each concept need to be characterized for its connectivity and direct steps from the root concept. For HARD model each concept need to be characterized as being a *hub* with mostly outgoing connections, *authority* with mostly incoming connections or *upper* node that is closer to the root node. For PC model the number of direct and indirect paths from root concept need to be noted. Second, using the structural characteristics the relative concept's weight is calculated.

4.2.1 Defining Structural Characteristics

At this stage the concept map is analyzed and structural characteristics of each node is defined as per every model, i.e. CRD, HARD, PC. For the CRD model the following characteristics need to be define for every concept:

1. Outgoing Connectivity (***o***) : number of outgoing connections or links.
2. Incoming Connectivity (***i***) : number of incoming connections or links.
3. Distance (***d***): direct steps distance from the root concept.

For example in the concept map in figure 3.1, the concept "SDK" is one step away from the root, hence, it has a distance of *d=1*, and connectivity of *o=4*, and *i=1*.

In the PC model a concept is characterized by the number of paths crossing the concept starting from the root concept. For example concept "SDK" has path count of one, whereas concept "Configuration" has a path count of three. Look to table 4.2 for position characteristics for each concept in "Mobile Applications" Concept Map as calculated by the program.

Table 4. 2 Position Characteristics for models CRD and PC

Concept	Incoming Connections	Outgoing Connections	Distance to Root	Path Count
Mobile Applications	0	3	0	1
Device Specifications	2	2	1	2
Software	1	3	2	2
Connectivity	1	3	2	2
operating method	1	1	3	2
platform	1	1	3	2
java support	1	2	3	2
profile	2	1	2	3
configuration	2	1	2	3
CLDC	1	0	3	3
MIDP	1	0	3	3
USB	1	0	3	2
bluetooth	1	0	3	2
GPS	1	0	3	2
SDK	1	5	1	1
IDE	1	2	1	1
Emulator	1	0	2	1
API	1	0	2	1
NetBeen	1	0	2	1
Carbide c++	1	0	2	1
S60 3rdE	1	0	4	2
symbian	1	0	4	2

Finally in the HARD model, concepts are characterized as hub, authority and upper nodes. In [11] HITS iterative algorithm was adapted to calculate the relative hub, authority and upper nodes' positional weights. [11] proved that the proposed algorithm produces positional weights which are ensured to reach a fixed point after a number of iterations equivalent to the number of concepts in the corresponding concept map. The algorithm follows these steps:

1. Set all concepts' weights ***W(C)*** to 1 such that:
   - → hub_weight = 1
   - → authority_weight = 1
   - → upper_weight = 1

In the following steps *links* will refer to the set of edges in the concept map. ***q*** and ***p*** will represent any two nodes, i.e. concepts , in the concept map. Hence, the weight of concept ***q*** will be expressed as ***W (q)*** and the link between concept ***q*** and concept ***p*** will be represented as ***(p,q).***

2. Normalize weights such that:

$$\sum_{\substack{(C)\in Concept\ Map \\ W\in\{\,authority\_weight\,,\\ hub\_weight\,,\\ upper\_weight\,\}}} W(C)^2 = 1$$

To ensure that this constraint is met, in every step of this algorithm the positional weight (e.g. hub_weight, authority_weight, upper_weight) value for every concept is divided by the sum of the squares of all corresponding positional weight values in the concept map. This will be further explained in every step later on.

3. Calculate hub_weight such that:
   hub_weight(p) = sum of authority_weight of all nodes, i.e. ***q_1, q_2,*** pointed to by the current node as shown in figure 4.5.

$$\text{hub\_weight}\ (p) = \sum_{(p,q)\in links} \text{authority\_weight}\ \ (q)$$

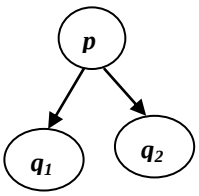


Figure 4.5 The Basic Hub Operation

4. Normalize hub_weight to match the constrain in step 2 as:

   $$\text{hub\_weight(p)} = \frac{\text{hub\_weight(p)}}{\sum_{C \in Concept\ Map} (\text{hub\_Weight(C)})^2}$$

5. Calculate authority weight such that:
   authority_weight (p) = sum of hub_weight of all nodes, i.e. q_1, q_2, pointing at the current authority as shown in figure 4.6.

   $$\text{authority\_weight}\ (p) = \sum_{(q,p)\in links} \text{hub\_weight}\ \ (q)$$

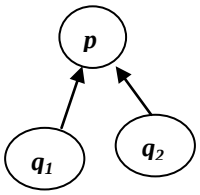


Figure 4.6 The Basic Authority Operation

6. Normalize authority_weight to match the constrain in step 2 as:

   $$\text{authority\_weight (p)} = \frac{\text{authority\_weight (p)}}{\sum_{C \in Concept\ Map} (\text{authority\_weight (C)})^2}$$

7. Repeat step 3 to 6 until weights converge. Normally it's repeated as many times as the number of concepts in the concept map.

8. Calculate Upper node weight as:

   $$\text{upper\_weight}\ (p) = \begin{cases} 1\ if\ \nexists\ (p,q) \in links \\ \sum_{(q,p)\in links} u - Weight\ (q)^2 \end{cases}$$

That is if the node is one level from the root node then assign a weight of one, otherwise sum up the square of upper_weight of nodes between the current node and the root until the root node is reached then sum up the value of one.

9. Normalize upper_weight according to the constraint in step 2 until they converge as follows:

$$\text{upper\_weight (p)} = \frac{\text{upper\_weight (p)}}{\sum_{C \in Concept\ Map} (\text{upper\_weight (C)})^2}$$

Look at table 4.3 for authority_weight, hub_weight, and upper_weight as calculated by the program for each concept in the "Mobile Applications" Concept Map. We can see that root node , i.e. *Mobile Applications*, with zero incoming links has a-weight = zero, whereas leaf nodes with zero outgoing connections , e.g. CLDC, MIDP, USB, has h-Weight = zero.

Table 4.3 Hub-Weight, Authority-Weight and Upper-Weight for the Mobile Applications Concept Map

Concept	Authority Weight	Hub Weight	Upper Weight
Mobile Applications	0.0	0.30127055432506217	0.22404013297790595
Device Specifications	0.21432922519005737	0.3012705574388439	0.22404006414038696
Software	0.21432919533731346	0.3012707293860387	0.22403974246796754
Connectivity	0.21432926347692438	0.30127097549556475	0.22403942096168228
operating method	0.21432940216400945	0.30127039860282046	0.2240381359264399
Platform	0.21432968036836816	0.30126998690136514	0.2240368591492703
java support	0.21433023706586787	0.30127214688299847	0.22403431045141253
Profile	0.21433590062934701	0.3012675624581721	0.224030997892358
Configuration	0.21434308910046893	0.30126418156836504	0.22402449614433687
CLDC	0.21433917820744042	0.0	0.22396792569282445
MIDP	0.2143478705319647	0.0	0.2239400975546893
USB	0.21436562147110838	0.0	0.22387548481800387
Bluetooth	0.21440124888965453	0.0	0.22371228832135642
GPS	0.2144725562867696	0.0	0.22338558262239433
SDK	0.21461523780784258	0.30276979535499987	0.22273438890281544
IDE	0.21490086478256168	0.302423025614977	0.22143443381309022
Emulator	0.21548994090100104	0.0	0.21631335963115128
API	0.21665718572368847	0.0	0.2113263935378981
NetBeen	0.21899484639198538	0.0	0.19935996314420204
Carbide c++	0.2237610239221156	0.0	0.18162474680477467
S60 3rdE	0.2333839048458918	0.0	0.15427281184720937
Symbian	0.2541160148953251	0.0	0.10613788906382356

4.2.2 Calculating Concepts' Structural Weights

After defining the structural characteristics of every concept in the concept map using the three different model, i.e. CRD, HARD and PC. Now the concept weight that reflects its importance in the mind of the user can be calculated as follows:

For CRD Model:

$$W(k) = \left(\alpha \,.\, o(k) + \beta \,.\, i\,(k)\right).\,(1/(d(k) + 1))^{\ 1/\delta} \qquad (4.4)$$

For HARD model:

$$W(k) = \alpha \,.\, h(k) + \beta \,.\, a(k) + \gamma \,.\, u(k) \qquad (4.5)$$

For PC Model the path count reflect the concept's weight.

After calculating the weights they need to be normalized in order to fall in the interval [0,1]. To do so every concept weight is divided by the maximum weight for the corresponding topological model for the current concept map. For example if in a concept map using CRD model the highest weight was five for a given concept, then all concept weights are divided by five. Weights along with similarity factors are combined later in section 4.3 to retrieve relevant knowledge for users from RSS feeds. The following 2 tables show the topological weights for each concept as per the three models CRD, HARD and PC before and after normalization, see tables 4.4 and 4.5 respectively.

Table 4.4 Topological Weights Before Normalization

Concept	CRD Weight	HARD Weight	PC Weight
Mobile Applications	27.0	2.9354751219034654	1.0
Device Specifications	10.0	3.1498043062800396	2.0
Software	9.333333333333332	3.1498055022796296	2.0
Connectivity	9.333333333333332	3.1498074638986897	2.0
operating method	2.5	3.149801125515834	2.0
platform	2.5	3.1497964216299246	2.0
java support	4.75	3.149813869464267	2.0
profile	3.6666666666666665	3.149774960645254	3.0
configuration	3.6666666666666665	3.1497452193600908	3.0
CLDC	0.25	0.43830710390026484	3.0
MIDP	0.25	0.438287968086654	3.0
USB	0.25	0.4382411062891123	2.0
bluetooth	0.25	0.438113537211011	2.0
GPS	0.25	0.4378581389091639	2.0
SDK	23.0	3.1622777849056565	1.0
IDE	9.5	3.158142529130445	1.0
Emulator	0.3333333333333333	0.4318033005321523	1.0
API	0.3333333333333333	0.4279835792615866	1.0
NetBeen	0.3333333333333333	0.4183548095361874	1.0
Carbide c++	0.3333333333333333	0.4053857707268903	1.0
S60 3rdE	0.2	0.3876567166931012	2.0
symbian	0.2	0.36025390395914864	2.0

Table 4.5 Topological Weights After Normalization

Concept	CRD Weight	HARD Weight	PC Weight
Mobile Applications	1.0	0.9282787033812219	0.3333333333333333
Device Specifications	0.37037037037037035	0.9960555398753531	0.6666666666666666
Software	0.345679012345679	0.9960559180836167	0.6666666666666666
Connectivity	0.345679012345679	0.9960565384020055	0.6666666666666666
operating method	0.09259259259259259	0.9960545340294338	0.6666666666666666
platform	0.09259259259259259	0.9960530465301598	0.6666666666666666
java support	0.17592592592592593	0.996058564019618	0.6666666666666666
profile	0.13580246913580246	0.9960462599711886	1.0
configuration	0.13580246913580246	0.9960368549513939	1.0
CLDC	0.009259259259259259	0.13860487082836756	1.0
MIDP	0.009259259259259259	0.1385988195530172	1.0
USB	0.009259259259259259	0.13858400055205358	0.6666666666666666
bluetooth	0.009259259259259259	0.13854365966906404	0.6666666666666666
GPS	0.009259259259259259	0.13846289563781222	0.6666666666666666
SDK	0.8518518518518519	1.0	0.3333333333333333
IDE	0.35185185185185186	0.9986923173558787	0.3333333333333333
Emulator	0.012345679012345678	0.13654818769978322	0.3333333333333333
API	0.012345679012345678	0.1353402858232314	0.3333333333333333
NetBeen	0.012345679012345678	0.1322954016035845	0.3333333333333333
Carbide c++	0.012345679012345678	0.12819423159530705	0.3333333333333333
S60 3rdE	0.007407407407407408	0.1225878126657575	0.6666666666666666
symbian	0.007407407407407408	0.11392228275413713	0.6666666666666666

4.3 Retrieving Knowledge From RSS Feeds

After calculating the distance correlation factors between every two stemmed words in the documents collection, the fuzzy set is ready to be used. Also the structure of the concept map was analyzed in section 4.2 and concept's structural weights were calculated. Now, the user is asked to define one or multiple RSS feeds as the source of knowledge and choose the topological model to be used, i.e. HARD, CRD, PC.

Once RSS feeds and topological model are defined , the knowledge retrieval process works automatically following the steps shown in figure 4.7:

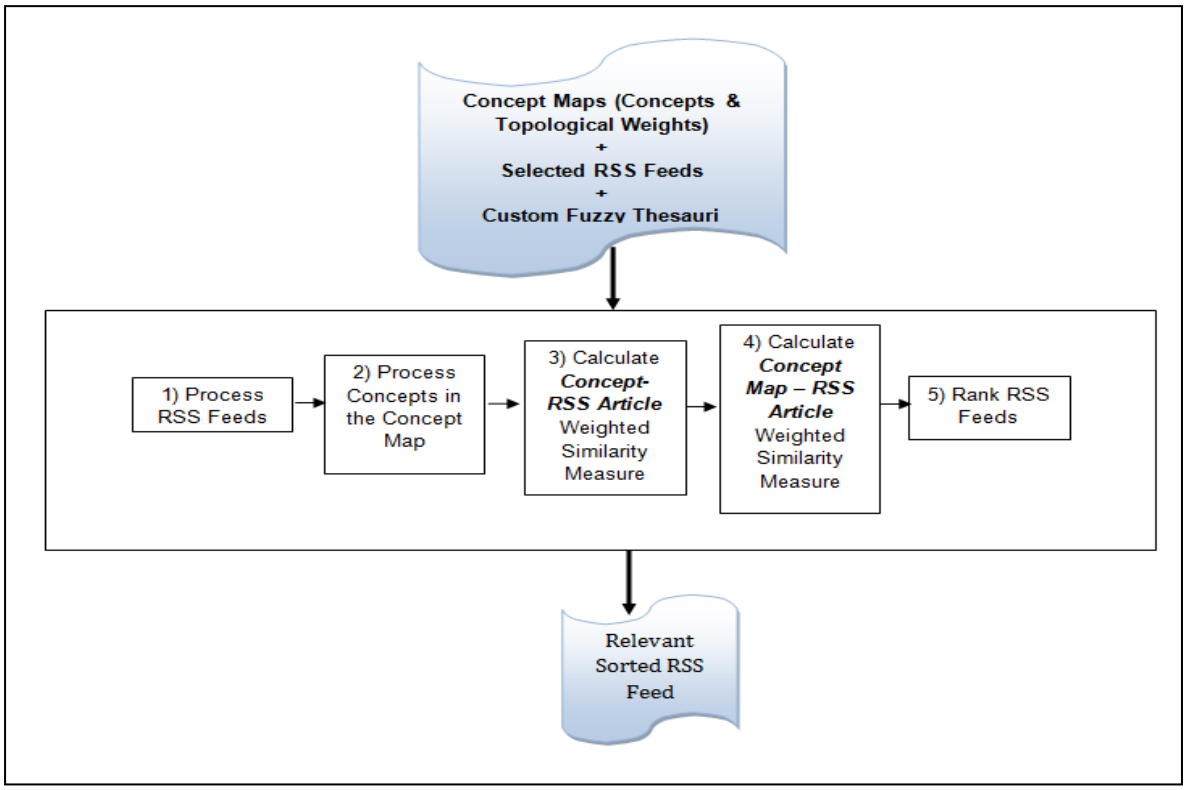


Figure4.7 Process of Retrieving Knowledge From RSS Feeds

1) Process the RSS feeds as follows:
    a. Extract plain text from the content of the <*title*> and <*content*> tags of the RSS feed
    b. Concatenate the content of the <*title*> and <*content*>
    c. Remove stop words and stem
    d. Insert all processed RSS articles (i.e. called *items* in *XML* syntax) in a vector.

2) Process Concepts in the concept map by stemming. Porter Stemming algorithm explained earlier in section 4.1.2 is used her to remove the suffixes from every concept in the concept map.
3) Compare every RSS article to the concept map and assign a ***concept-article*** similarity weight as follows:

Every stemmed concept, i.e. *k,* in the concept map is compared with every stemmed word, i.e. *w,* in a single RSS article, i.e. *d,* against the pre-computed distance correlation factors, i.e. *Cf,* as shown in figure 4.8 , and a similarity measure between the single concept, i.e. *k,* and the RSS article, i.e. *d,* is calculated according to 4.6.

$$\textit{Concept\_Article Similarity}\ (\mu_{k,d}) = 1 - \prod_{w \in d}(1 - Cf_{kw}) \tag{4.6}$$

For multiple-term concepts the distance correlation factors, i.e. *Cf,* for every term is compared separately and then the average distance correlation factors is computed to be used in equation 4.1. For example for the concept "Mobile applications", the proposed method would look up for $Cf_{\text{mobil,w}}$ and $Cf_{\text{applic,w}}$ and use the average value of both correlation factors.

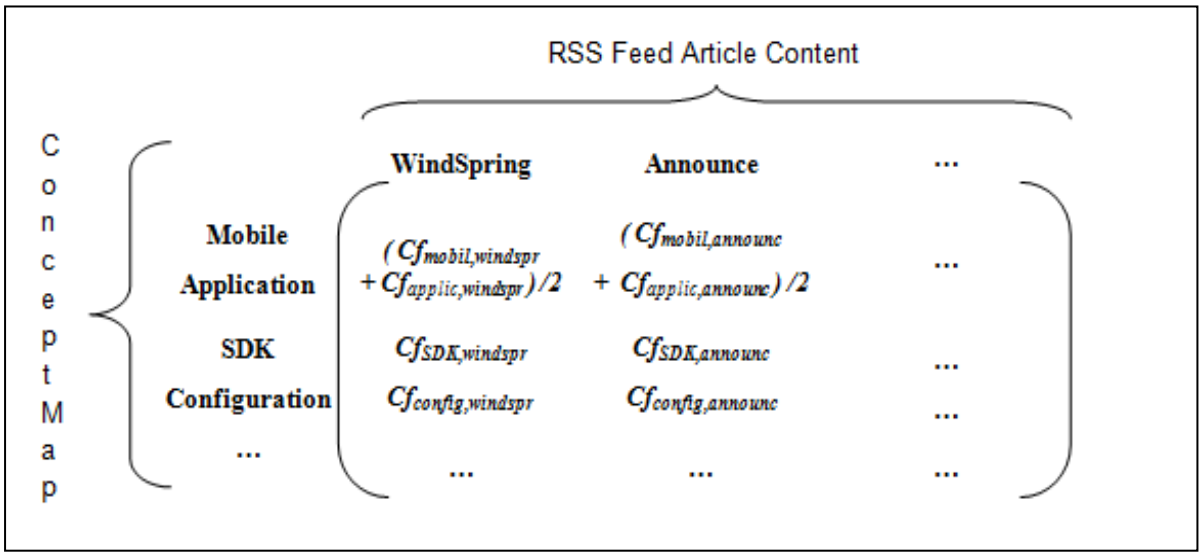


Figure 4.8 Concept-RSS Article Comparison

a. The similarity measure is then multiplied by the concept's structural weight i.e. ***W,*** calculated earlier in section 4.2. A weighted similarity measure is produced using 4.7:

$$\textit{Weighted Concept\_Article similarity} = W(k)*\mu_{k,d} \quad (4.7)$$

b. The overall weighted similarity between the concept map and a single RSS feed article i.e., ***d,*** is calculated according to 4.8.

$$\textit{Weighted ConceptMap\_Article similarity} = \frac{W(k1).\mu_{k1,d} + W(k2).\mu_{k2,d} + \ldots + W(kn).\mu_{kn,d}}{n} \quad (4.8)$$

Where **n** is the number of concepts ***k*** in the concept map.

c. Repeat steps (a) to (c) until all RSS articles are compared to the concept map

4) Display the top 5, 10, 15 or 20 RSS articles to the user in a new RSS feed, figure 5.14.

The proposed method was evaluated with experts and users to measure consistency and precision of the results. Experimental results are discussed in chapter 5.

4.4 Implementation

For the sack of validation of the proposed method, a JAVA program was developed. Eclipse was used as a development environment with JAVA Runtime Environment (JRE) version 6. yFiles 2.7.0.2 library [27] was used for graph, i.e. concept map, parsing and analysis. Lucene 3.0.1 library [28] was used for applying the porter stemming and stopWords removal algorithms. Wikixmlj library [29] was used to parse Wikipedia files in xml format. Rome 1.0 library [30] was used for parsing RSS feeds. The program was tested on a Linux machine so as to allow for the maximum size of the JAVA Virtual Machine Memory (JVM memory) of 3 GB. This huge amount of virtual memory was required for the creation and processing of the fuzzy thesauri. In addition, this hardware limitation put some constraints on

the testing as will be shown later. The program is composed of 3 modules as shown in figure 4.9. Source code of the program is available in appendix A.

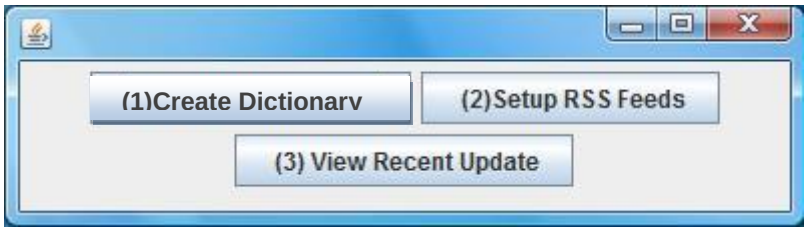


Figure 4.9 JAVA Program Three Modules

The first module, i.e. Create Dictionary, was designed to retrieve all Wikipedia articles related to the user's concept map in XML format and store them in a specific directory in the user's computer, then process them to create the fuzzy dictionary as explained earlier in section 4.1. The fuzzy dictionary is stored in a HashMap in a location relative to the program's location on the user's computer. HashMap was chosen as it allows for faster access and consumes smaller space on the hard disk. The second module, i.e. Setup RSS Feeds, was designed to store the RSS feeds of interest to the user of the concept map. The last module, i.e. View Recent Updates, was designed to retrieve the content of the pre-defined RSS feeds, process the content, performs structural analysis of the concept map and calculates the structural weights of the concepts as explained in section 4.2, and finally performs the similarity assessment and displays the top related feeds. In order to run the third module it's necessary to run the first two modules at least once. In the next section a presentation of how the program work will be presented.

4.4.1 Testing The Program

In order to test the program, a full cycle of the proposed method was run on the program using the concept map about "Mobile Applications" shown in figure 4.10. Starting with the first module, i.e. Create Dictionary, to create the fuzzy thesauri, the program prompts the user for a concept map location and for the number of Wikipedia articles to be retrieved. In figure 4.10 the user specifies the location of the "Mobile Applications" concept map which must be in graphml format. Then the user choose to retrieve the first article returned from Wikipedia search results for every concept in the concept map as you can see in figure 4.11. These articles will make up the custom text corpora on top of which the fuzzy

thesauri will be created. The hardware limitations didn't allow the testing of more than one article for each concept since the maximum possible size for the JAVA Virtual Machine Memory (JVM memory) on Linux machine was 3 GB.

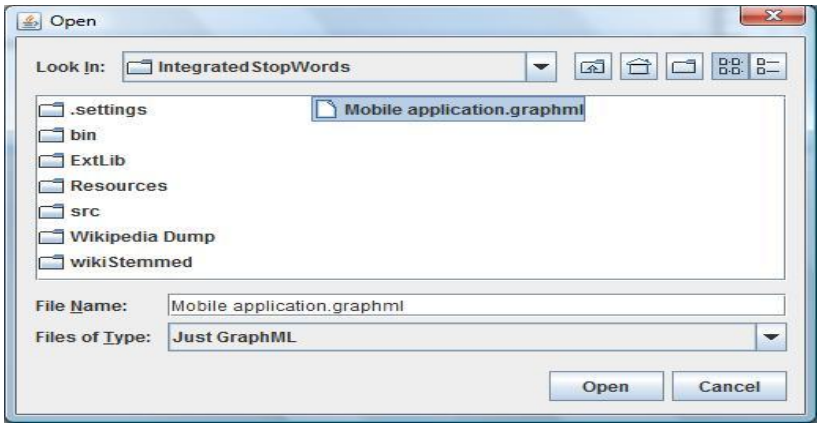


Figure 4.10 Retrieve Concept Map

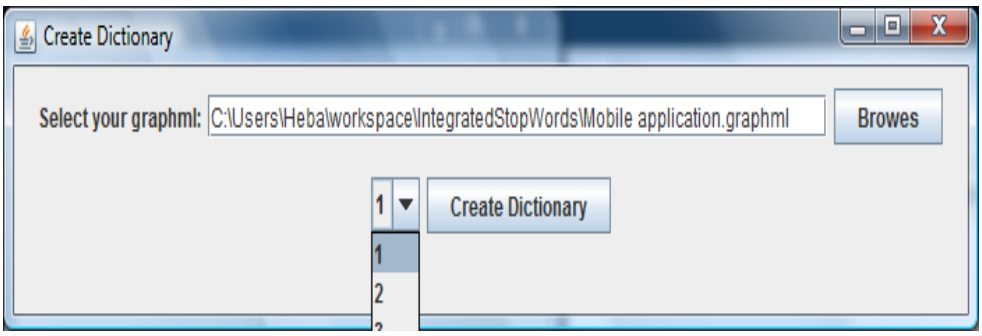


Figure 4.11 Specify The number of Wikipedia Documents to Be Retrieved

The "Mobile Applications" concept map contains 22 concepts. The user chose to retrieve the first article in the search results that should be the most relevant article. For example for the first concept "Mobile Applications", the program passes "Mobile and Applications" to ***Wikipedia Search***, and retrieve the title of the first article in the search results, i.e. "*Mobile Application Development*". The retrieved title is then passed by the program to the ***Special Export Utility*** of Wikipedia to retrieve the article in XML format as shown in figure 4.12.

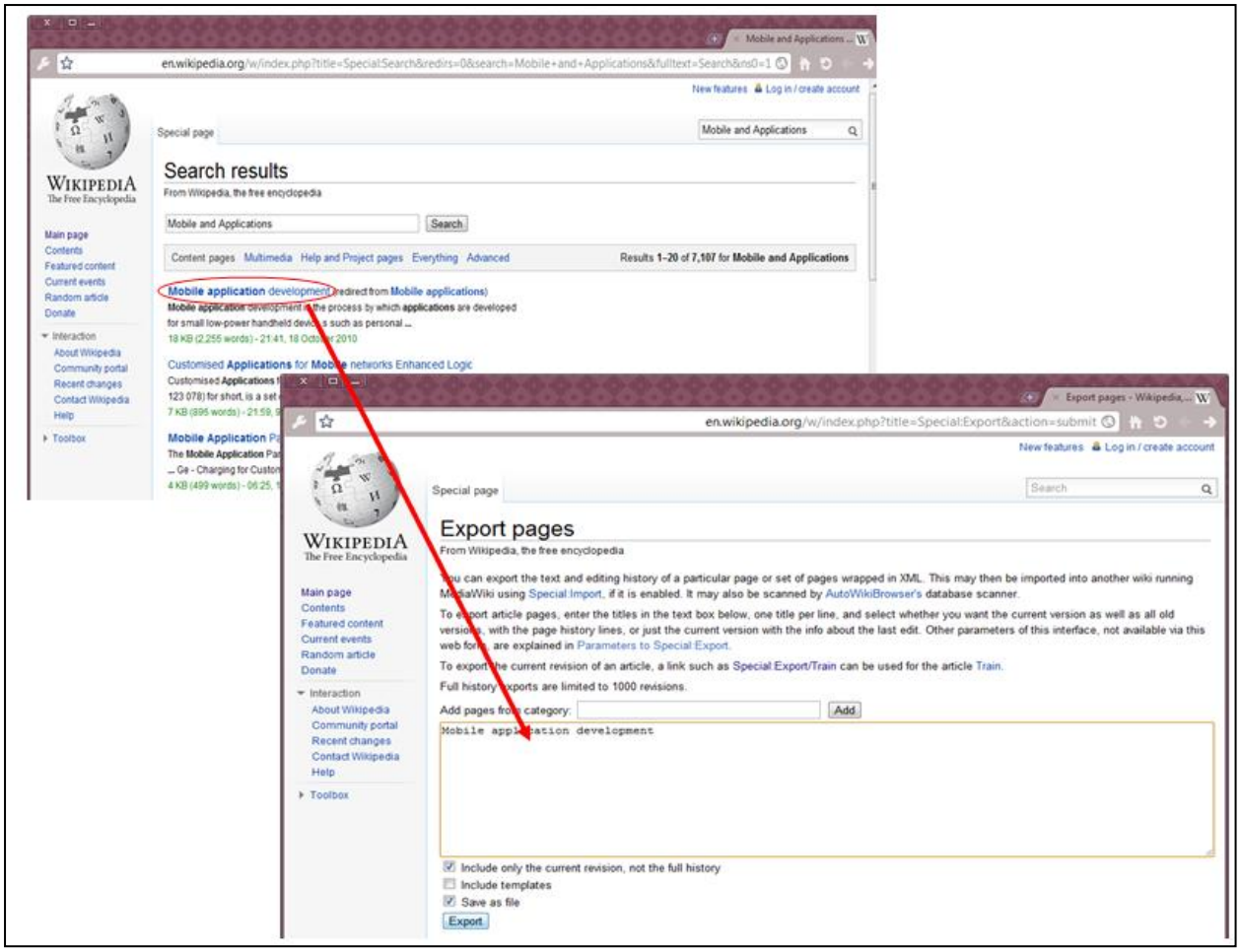


Figure 4.12 Wikipedia Search and Export Utilities

For the " Mobile applications" concept map 2 concepts didn't return any search results, the program retrieved the following 20 articles:

1) **Mobile application development**
2) **Symbian**
3) **Carbide.c++**
4) **Application Programming Interface API**
5) **Emulator**
6) **IDE**
7) **Software Development Kit SDK**
8) **Global Positioning System GPS**
9) **Bluetooth**
10) **Universal Serial Bus**
11) **Mobile Information Device Profile MIDP**
12) **Connected Limited Device Configuration CLDC**
13) **Configuration**
14) **Unix shell**
15) **Java Telephony API**
16) **Platform**
17) **Operating method**
18) **Connection**
19) **Computer software**
20) **IEC 62056 -DLMS**

Amongst the 20 articles retrieved by the documents' retrieval process 19 articles specifically were related to the concepts in the concept map. "***Unix Shell***" article is generally related topic in software development. All the articles are then processed to extract the content of the <***title***> and <***text***> tags from the corresponding XML files, concatenate the content, remove the stop words, stem the content and finally create the distance correlation matrix which will be used as the fuzzy thesauri. Figure 4.13 shows part of a Wikipedia document about "**Symbian**" and how it looks after extracting ***text*** content from its XML document, removing stopWords and finally stemming.

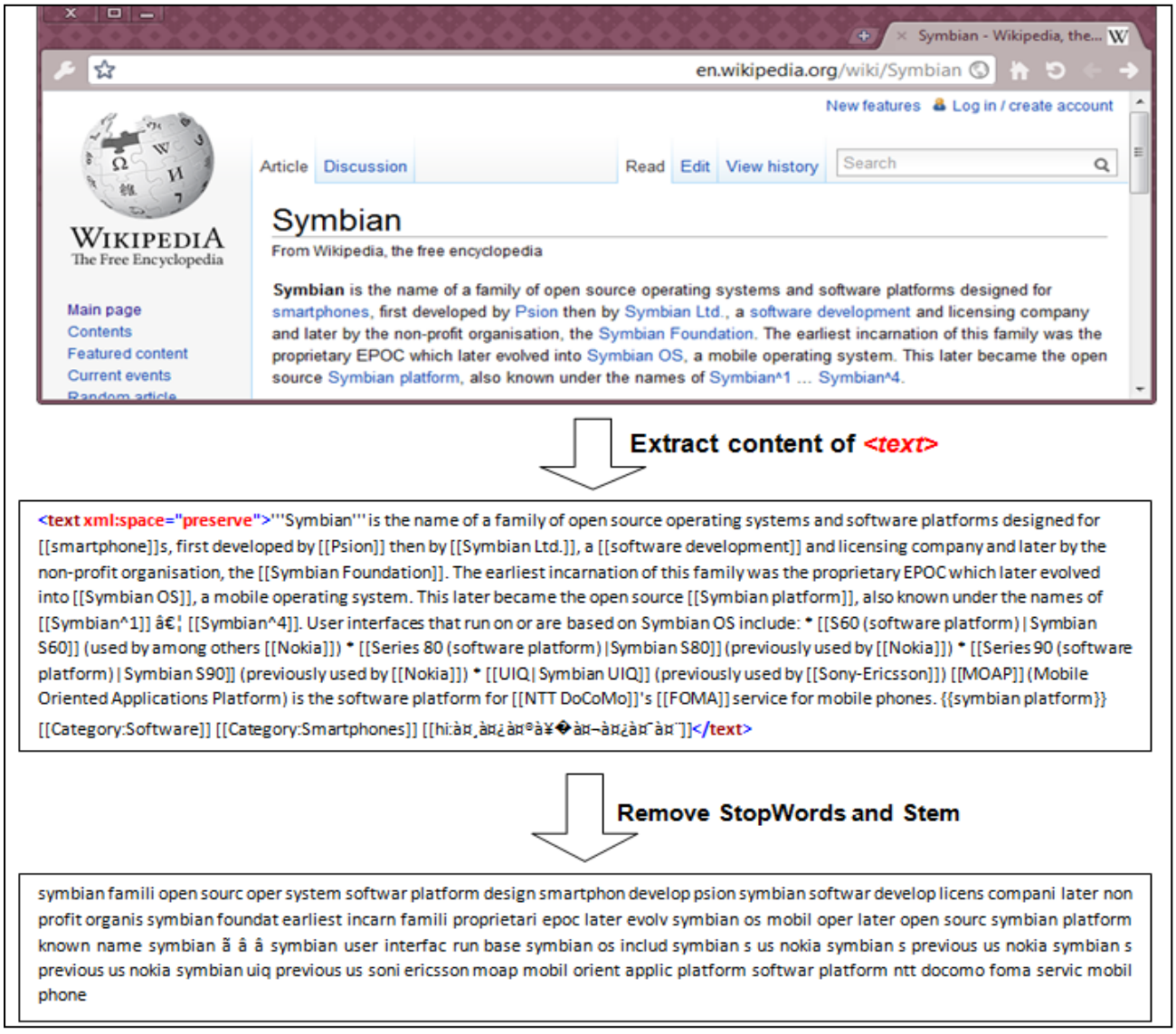


Figure 4.13 Processing Wikipedia Article

By processing all the 20 retrieved Wikipedia articles related to "Mobile Applications", the program has extracted ***4202*** distinct stemmed words. After that, distance correlation factors are calculated as explained in section 4.3 and stored in a HashMap on the

user's hard disk. For example figure 4.14 shows part of the HashMap, the term "Windows XP" after stemming and its relationship to few other keywords in the corpora.

windowsxp	multiprocess	0.004237288
windowsxp	fortran	0.0038315852
windowsxp	directmedia	0.004032258
windowsxp	sdl	0.0040
windowsxp	swig	0.0032280248
windowsxp	opensourc	0.0032174834
windowsxp	mangl	0.0033670033
windowsxp	corba	0.003267974
windowsxp	intercommun	0.003215434
windowsxp	xpcom	0.0032051282
windowsxp	dom	0.0030674846
windowsxp	chanc	0.0030487804
windowsxp	osid	0.0029239766
windowsxp	mikespedia	0.0
windowsxp	carbid	0.0

Figure 4.14 Part of the HashMap - Fuzzy Thesauri

After calculating the distance correlation factors between every two stemmed words in the documents collection, the fuzzy set is ready to be used. Now, the user is asked to define one or multiple RSS feeds as the source of knowledge as shown in figure 4.15.

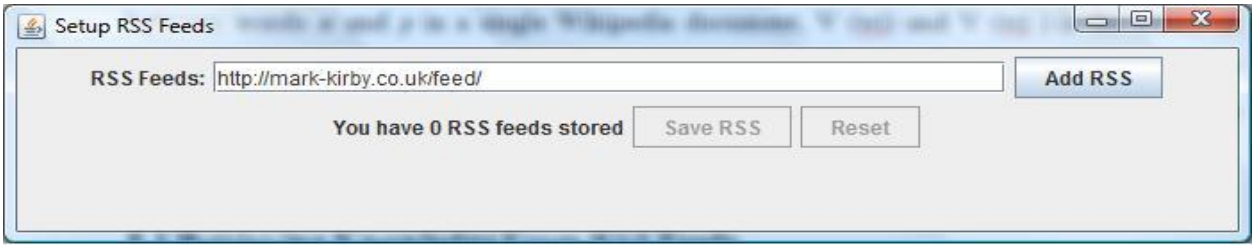


Figure 4.15 Define RSS Feed/s

Then the user is asked to choose the structural model to be applied and the number of retrieved feeds as shown in figure 4.16. At this point the structure of the concept map is analyzed according to the selected structural model by the user and concept's structural weights are calculated as explained in section 4.2. Finally the fuzzy thesauri and the structural weights are used to assess similarity and return top related feeds as shown in figure 4.17.

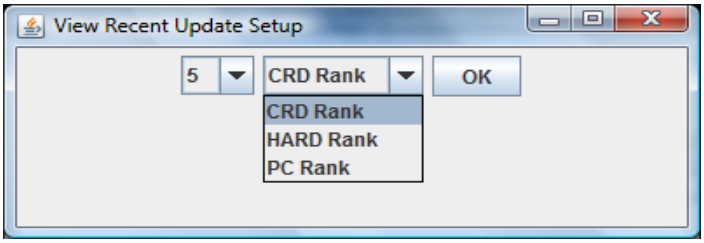


Figure 4.16 Choose Concept Map Topological Model

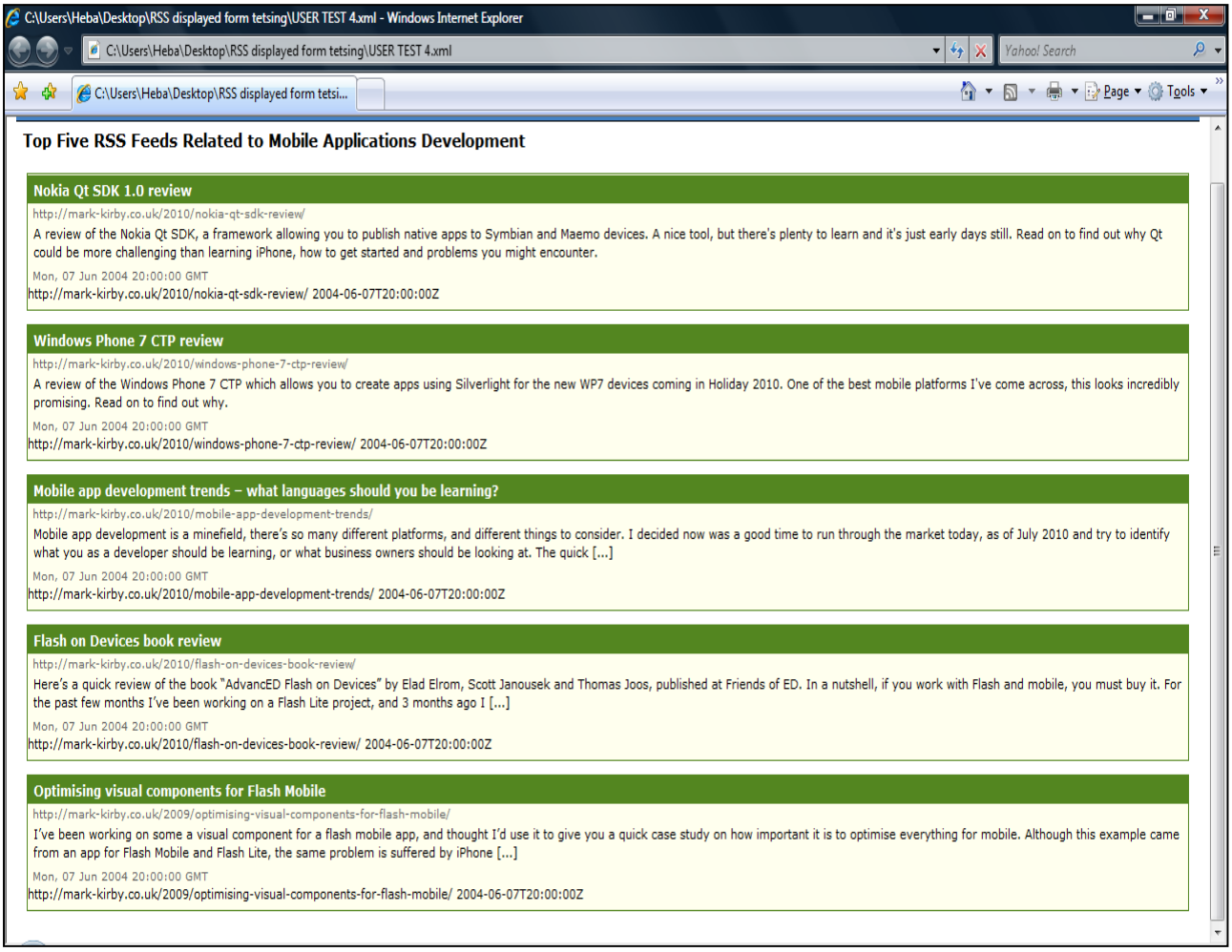


Figure 4.17 Sample Results Using HARD Model

CHAPTER 5

EVALUATION

In order to evaluate the proposed method multiple tests were conducted. First, the fixed parameters in models HARD and CRD were tested to find out how sensitive the method results were to those parameters and whether different values of the parameters would affect the usefulness of the results generated by the method. A factorial sensitivity analysis was conducted to assess the parameters. Second, the stability of the method was tested to see whether the method produces results consistent with the context of the concept map or not. Human experts were asked to generate ranking of a test RSS feed. The human's ranking were compared to the method ranking using Fleiss' Kappa and Cohen's Kappa Coefficients. Finally, relevance and precision of the results produced by the method were tested with the actual user of the test concept map. The user was asked to select top five related articles in multiple tests, then user's selection was compared to method's selection to see to what degree the method matches the user's knowledge interests.

Throughout the evaluation process, the test concept map about mobile application development, introduced in chapter 3, was used. User of the concept map and experts are all in the field of mobile application developments.

5.1 Models' Fixed Parameters

In [11] best-fit parameters for CRD and HARD models indicated that authority nodes in CRD model are of more importance than hub nodes. On the other hand, hub nodes were found to be not relevant when calculating node's weight in the HARD model. This information was obtained by conducting human subjects' evaluation on 10 different concepts map. Researchers used artificial labels in order to eliminate the influence of participants' domain knowledge. Given that it's difficult to get perfectly objective human subjects' evaluation we need to further examine the effects of the model's parameters on the method's results in a more controlled sensitivity experiment that doesn't involve human subjects' judgment to eliminate background influences or personal preferences of subjects.

To find out the critical model's parameters, a test feed was created using "Google Alerts". The feed retrieves the 20 most recent web contents (e.g. news, blogs, videos and discussions) related to the keywords: Mobile; Applications; IDE; SDK. The feed sorts the retrieved articles according to their publishing dates. The selected keywords help retrieving articles generally related to the development of mobile applications, but it's not necessary to have all the 20 articles exactly related to mobile applications development. On the other hand, sorting articles by date will ensure that the test feed will continuously provide objective and stable rank.

Best-fit parameters proposed by [11] falls between 0 and 10. In order to examine the whole range of the best-fit parameters, two fractional factorial experiments [31] were conducted, each tests the three parameters (i.e. α, β, δ) at 2 levels as shown in figure 5.1. The first experiment tests the levels 3 and 7. The second experiment tests the levels 1 and 9. Observations were taken on 2 different days, each day represents a replica.

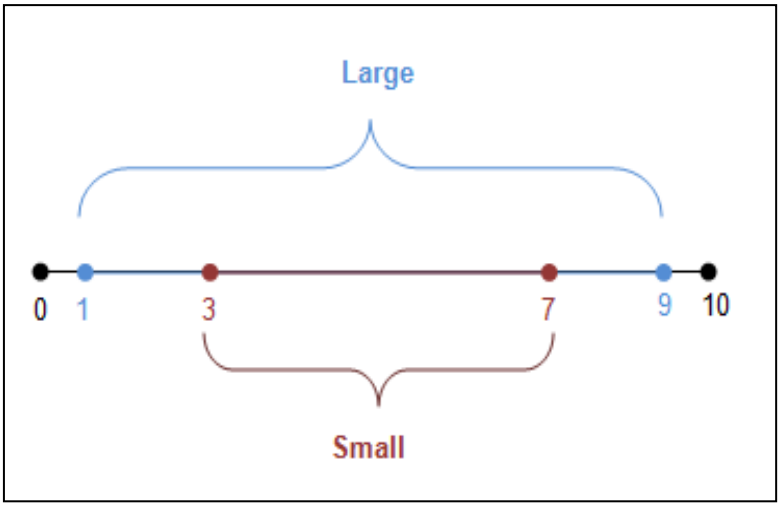


Figure 5.1 Parameters' Range in the Factorial Experiment

5.1.1 Fractional Factorial Experiments – 2^{3-1}

Purpose: To determine the significant factors that affect the ranks generated using the proposed method.
Factors: α, β, δ
Response Variable: The Mean Square Error (MSE) between Google feed's rank and the proposed method's rank is used as a response.
Number of replica: 2 replica
Experiments Design:

Four factorial experiments were conducted. “Small” is used to indicate the experiments with levels 3,7 , and “Large” is used to indicate the experiments with levels 1,9 as shown earlier in figure 5.1.

1. HARD small (a 2^{3-1} resolution III design with 0 center point) , levels 3,7
2. HARD large (a 2^{3-1} resolution III design with 0 center point), levels 1,9
3. CRD small (a 2^{3-1} resolution III design with 0 center point), levels 3,7
4. CRD large (a 2^{3-1} resolution III design with 0 center point), levels 1,9

5.1.2 Analysis of the Experiments’ Data

(The experimental data was analyzed using Minitab 15 Professional software.)

Scanning the column of *p*-values (labeled P in Minitab output) for small values (i.e. 3,7) and large values (i.e. 1, 9) of the HARD model shows no significant effects of the parameters α, β and δ at the 0.05 level, figure 5.2.

Results for: HARD Small.MTW
Factorial Fit: MSE versus Alpha, Beta, Gama

Estimated Effects and Coefficients for MSE (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		54.112	3.591	15.07	0.000
Alpha	-2.975	-1.487	3.591	-0.41	0.700
Beta	2.725	1.362	3.591	0.38	0.724
Gama	-0.575	-0.288	3.591	-0.08	0.940

Results for: HARD Large.MTW
Factorial Fit: MSE versus Alpha, Beta, Gama

Estimated Effects and Coefficients for MSE (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		54.612	4.109	13.29	0.000
Alpha	-7.525	-3.762	4.109	-0.92	0.412
Beta	4.825	2.412	4.109	0.59	0.589
Gama	0.875	0.437	4.109	0.11	0.920

Figure 5.2 Values for HARD Model Factorial Experiment

Similarly, for the CRD model, when Scanning the column of *p*-values for small values (i.e. 3,7) and large values (i.e. 1, 9) we can see no significant effects of the parameters α, β and δ at the 0.05 level, figure 5.3

Results for: CRD Small.MTW
Factorial Fit: MSE versus Alpha, Beta, Gama

Estimated Effects and Coefficients for MSE (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		56.188	4.067	13.81	0.000
Alpha	-1.525	-0.762	4.067	-0.19	0.860
Beta	1.525	0.762	4.067	0.19	0.860
Gama	-2.375	-1.188	4.067	-0.29	0.785

Results for: CRD Large.MTW
Factorial Fit: MSE versus Alpha, Beta, Gama

Estimated Effects and Coefficients for MSE (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		56.900	4.026	14.13	0.000
Alpha	-3.850	-1.925	4.026	-0.48	0.658
Beta	3.700	1.850	4.026	0.46	0.670
Gama	-3.650	-1.825	4.026	-0.45	0.674

Figure 5.3 Values for CRD Model Factorial Experiment

Discussion

As analyzed above, the p-value indicates that there is no significant effect of any of the parameters used in the models HARD and CRD. As a result, changing the values of the parameters α, β and δ doesn't significantly change the rank generated by the method. This in turn, indicates that the method performance is stable using different parameters values and consequently the major influence on the method's result come from the similarity factors and the topological weights of the concepts in the concept map rather than method's parameters. Therefore, the method's results are more likely to specifically reflect the concept map's context without being influenced by the method's parameters.

5.2 Method Consistency Compared to Human Experts Judgment

As analyzed earlier the different method's models are not significantly sensitive to different parameters values. Therefore, the proposed method's performance is likely to remain consistent for a single concept map using any model (i.e. PC, HARD , CRD) with different combination of parameters. On the other hand, In order to assess consistency amongst the three models implemented (i.e. HARD, CRD, PC) with the user's interest presented by the concept map, an inter-rater reliability tests were conducted using Cohen's Kappa and Fleiss' Kappa [32] coefficients at three steps:

Step 1: testing consistency amongst experts.
Step2: testing ranking consistency amongst the results of the 3 different models (i.e. HARD, CRD, PC)
Step3: testing consistency between method's results and experts.

5.2.1 Inter-Rater Reliability Tests

The "Mobile Applications" concept map created in chapter 4 is used here. The concept map specifically represents the user's knowledge interests. Using Google Alerts' feed created earlier for the factorial experiment, five experts working in the development of mobile applications were asked to rank the feed's 20 articles according to their relevance to the context of the user's concept map. Test Document is presented in appendix D. Information about experts are shown in table 5.1. On the other hand, the proposed method using the three models (i.e. HARD, CRD, PC) was used with different parameters to generate ranking of the Google feed's articles relevant to the context of the concept map of the user.

Table 5.1Experts' Backgrounds and Years of Experience

Experts	**E1**	**E2**	**E3**	**E4**	**E5**
Area	Nokia - Symbian	Nokia - Symbian	Symbian & Anroid	Symbian & Anroid	iPhone
Experience	More than 5	2-5	2-5	2-5	2-5

In the tests, experts and the method are considered raters, whereas every rank is considered a category. Therefore, there are 20 categories. The objective of this test is to assess whether raters agree on the same categories or not. The Kappa [32] is a measurement of agreement. In Cohen's Kappa, it measures agreements between two raters. In Fleiss's Kappa, it measures the overall agreements between all the raters. For most purposes, values greater than 0.75 or so may be taken to represent excellent agreement, values below 0.40 or so may be taken to represent poor agreement, and values between 0.40 and 0.75 may be taken to represent fair to very good agreement.

First of all, experts' agreement is assessed in two steps. First, agreement amongst all experts is analyzed using Fless's kappa coefficient for multiple users test and then pair-wise comparison is conducted between every two experts using Cohen's kappa coefficient. see appendix B for count matrixes.

5.2.2 Fleiss' Kappa – All Experts

Kappa value of 0.0158 indicates very low agreement amongst experts. A maximum of two experts agreed on some ranks. This is expected as with human raters it's very difficult to eliminate the influence of the raters' backgrounds. Raters come from different mobile

developments backgrounds. Some of them work on iPhone while others work on Symbian and Anroid. Experts have different years of experience which can also influence their judgments. To further support Fleiss' kappa results a pair-wise comparison was conducted between every two experts and Cohen's kappa was calculated.

5.2.3 Cohens' Kappa –Experts' Pair-Wise Comparison

Table 5.2 shows that highest agreement level was between E1 and E2 who both work on Symbian but have relatively different years of experience. E2 has more than 2 years of experience whereas E1 has more than 5 years of experience. E2 and E1agreed on giving low importance to the articles related to Blackberry and iPhone. They agreed on ranks 12, 15, 19 and 20. However, they disagreed on the top five ranks. E2 considered articles related to IDE setup and platform very important whereas E1rated more advanced articles related to interfacing web application with mobile devices as being more important. This reflects the influence of years of experience on the experts' judgment. Even though kappa between E2 and E1is higher than others, it still indicates poor agreement because of the difference in the years of experience as explained above. On the other hand, agreement between E5 and all other experts was the lowest. E5 is the only expert specialized in iPhone and never worked on Nokia based platform (i.e. Symbian).

Table 5.2 Experts' Pair-Wise Kappa

	E3	E5	E2	E1	E4
E3		0.1128	0.1429	0.1429	0.1429
E5	0.1128		-0.0827	-0.0526	0.0827
E2	0.1128	-0.0827		0.3383	0.0226
E1	0.1429	-0.0526	0.3383		0.0376
E4	0.1429	0.0827	0.0226	0.0376	

Looking at the MSE values between E2's , E1's ranking and method's ranking in table 5.3 and table 5.4, E2's ranking has relatively low MSE in comparison with E1's. The user of the concept map has around less than two years of experience and works on Symbian. Thus, the concept map has the concepts: IDE, API and Platform on the top and consequently the articles related to these concepts are ranked on the top in the method's results. given that E1has more experience in the field, E1 would be more interested in advanced topics. On the other hand, E2 and the user of the concept map who have relatively less experience in the field would be more interested in topics related to IDE setup, API installation and code tips. This explains why E2's ranking would be the closest to the method ranking which reflects the context of the user's concept map.

Table 5.3 MSE Between System's Ranking and Experts Ranking -I

Model(α, β, δ)	PC	HARD 1-1-9	HARD 1-9-1	HARD 9-1-1	HARD 9-9-9	HARD 3-3-7	HARD 3-7-3	HARD 7-3-3-	HARD 7-7-7
E2	34.3	22.7	28.2	19.3	22.7	22.7	25.7	23.1	22.7
E1	78.2	69.6	74.1	66.6	69.6	69.6	72.6	70	69.6

Table 5.4 MSE Between System's Ranking and Experts Ranking -II

Model(α, β, δ)	CRD 1-1-9	CRD 1-9-1	CRD 9-1-1	CRD 9-9-9	CRD 3-3-7	CRD 3-7-3	CRD 7-3-3-	CRD 7-7-7
E2	24.1	30	24.9	24.1	24.1	26.2	25.2	24.1
E1	66.8	75.6	65.4	66.8	66.8	72.4	66.5	66.8

Looking at the MSE values between the other experts and method's ranking in table 5.5 and table 5.6, E5 has higher MSE than other experts who have some experience on Symbian platform (i.e. E3 & E4). E5 is specialized in iPhone. The results clearly show how experts' backgrounds influence their judgments.

Table 5.5 MSE Between System's Ranking and Experts Ranking -III

Model(α, β, δ)	PC	HARD 1-1-9	HARD 1-9-1	HARD 9-1-1	HARD 9-9-9	HARD 3-3-7	HARD 3-7-3	HARD 7-3-3-	HARD 7-7-7
E3	47.9	57.3	55.2	57.2	57.3	57.3	57.2	57.6	57.3
E4	53.9	52.2	53.4	54.6	52.2	52.2	53.3	52.7	52.2
E5	65.4	67.2	69.6	66.6	67.2	67.2	67.3	65.6	67.2

Table 5.6 MSE Between System's Ranking and Experts Ranking -IV

Model(α, β, δ)	CRD 1-1-9	CRD 1-9-1	CRD 9-1-1	CRD 9-9-9	CRD 3-3-7	CRD 3-7-3	CRD 7-3-3-	CRD 7-7-7
E3	53.2	54.2	52.6	53.2	53.2	57.1	53.3	53.2
E4	48.5	52.9	50.1	48.5	48.5	51.7	50.3	48.5
E5	64.2	69.2	62.4	64.2	64.2	68.1	62.5	64.2

5.2.4 Fleiss' Kappa – Models (CRD , HARD , PC)

Kappa value of 0.5553 indicates very good agreement amongst different methods' models using different parameters' levels. this in turn supports the result of the factorial experiment. The method's results are relatively consistent and are not influenced by any human factors as background or years of experience. Hence method's results are more likely to specifically reflect the context of the user's concept map.

5.2.5 Fleiss' Kappa – Models vs. Experts

Table 5.7 shows that all Kappa values between model and experts' ranking are very low which indicates poor agreement between experts and method. This is expected as experts come from different areas of mobile development, therefore their judgment is to some extent influenced by their area of expertise. The concept map represents knowledge of a mobile

application developer working on Sybian platform and uses netBeens and Carbide C++ as IDEs. However, some experts work on iPhone or Anroid, and not all of them rely on netBeens and Carbide C++. Also years of experience to a large extent influence experts' judgment.

Table 5.7 Experts vs. Method's Kappa

Model (Small/Large)	PC	HARD (3,7)	HARD (1,9)	CRD (3,7)	CRD (1,9)
Experts	**0.0175**	**0.1520**	**0.1272**	**0.1170**	**0.1038**

Discussion

As analyzed above that different method's models produces repeatedly almost similar results for a single concept map. It was also clear that method's results were consistent with the context of the concept map when compared to experts' judgment. Experts in the field of mobile development provided different ranking which were not perfectly consistent with the context of the concept map because of background and work experience influences. E2's ranking was the closest to the method's ranking. E2's background and work experience fall in the same area as of the user of the concept map. This supports how consistent the method results were to the context of the concept map. Considering the MSE values between E2 - who had similar knowledge interest, background and experience as of the user- and method, The Least MSE value(i.e. MSE = 19.3) was observed using the HARD model with parameters values of $\alpha=9$, $\beta=1$ and $\delta=1$. Therefore, these parameters value will be used in the next section for the relevance and precision tests.

5.3 Relevance and Precision

In any information retrieval (IR) system, typically precision and recall are used to measure the effectiveness and efficiency of the system. Recall assesses how well an IR system retrieves **all** the relevant information. Whereas, precision assesses how well the IR system distinguishes between relevant and irrelevant information. Usual web users would like every result on the first page to be relevant, but don't have the slightest interest in knowing all relevant documents on the web. Precision is more critical for ordinary web users. In contrast, various professional searchers especially in academia would be very concerned with getting all the relevant documents, and would tolerate fairly low precision results. Given that the proposed method is designed for almost every web user, precision is more critical issue than recall in this context.

Using the concept map about "Mobile Applications" developed in chapter 4, four tests were conducted with the user of the concept map. In order to assess the precision of the proposed method four categories were selected and a single RSS feed was randomly chosen from each category using an online RSS feeds' directory (i.e. http://www.feedage.com/). The test categories are:

1) Mobile Application Development
   - ***Selected Feed*** *: http://mark-kirby.co.uk/feed/*
2) Software Programming
   - ***Selected Feed :****http://www.i-programmer.info/index.php?option=com_ninjarsssyndicator&feed_id=3&format=raw*
3) Technology
   - ***Selected Feed :*** *http://telecomyatra.afaqs.com/rss/index.xml?section=research&category=Technology*
4) News
   - ***Selected Feed :*** *http://rss.cnn.com/rss/edition_americas.rss*

The categories were selected in a way that gradually moves from very specific to very general. The first category, i.e. Mobile Application Development, would contain exactly what the user of the concept map is looking for and at the same time belongs to a broader category, i.e. Software Programming, that belongs to a broader category, i.e. Technology. The last category, i.e. News, is very general. All the four categories may have articles related to the context of the concept map. However, closely related articles are expected to be in the category "Mobile Application Development". The proposed method should distinguish between generally related articles and closely related ones.

After selecting a single feed from every category, they were inputted into the method to retrieve the top 5 relevant articles. Finally, the user was asked to do manual selection of top five relevant articles without looking at the results of the method. User-selected top five articles were then compared to method's to see how well the method matches user's expectations and to what level the method was precise. The four tests are presented next. Results generated by the proposed method are presented in appendix).

5.3.1 Test No. 1

In the first test a single RSS feed from the category "Mobile Applications Development" was selected. The feed is published in the website of ***Mark Kirby*** who's an expert in mobile application development. In the website Mark publishes advice and thoughts on building great mobile applications.

User Evaluation:

The user's and the method's top three articles exactly matched. The top three articles were related directly to mobile application development, Nokia platform and windows phone 7 CTP. The user commented that the information related to the top three articles exactly answers the question (***What do you need to know when you develop a mobile application?).*** The remaining 2 articles selected by the method were very interesting to the user but with different ranking. The method succeeded in excluding the irrelevant articles which were not interesting to the user (e.g. articles about mobile application users' reviews). Overall the user found that all the 5 retrieved articles were relevant and provided the same ranking for the top 3 articles.

Precision = # (relevant Items Retrieved) / #(Retrieved Items)
Precision = 5/5 = 1

5.3.2 Test No. 2

In the second test two RSS feeds were used. One feed was selected from the category "Mobile Applications Development" and the second feed was selected from the category "Technology". The first feed is the same one used in test# 1. The second feed is published in ***afaqs! Telecom Yatra*** website. The website has a range of RSS feeds that keeps web users posted on the latest happenings in the technological world in India.

User Evaluation:

Again The user's and the method's top three articles exactly matched. The same top three articles as in test#1 were selected. One of the remaining 2 articles selected by the method was very interesting to the user. The last article selected by the method was not

interesting to the user as it relates to hardware configuration rather than software development.

Precision = # (relevant Items Retrieved) / #(Retrieved Items)
Precision = 4/5 = 0.8

5.3.3 Test No. 3

In the third test three RSS feeds were used. One feed was selected from the category "Mobile Applications Development" , the second feed was selected from the category "Technology", and the third feed was selected from the category "Software Programming". The first and second feeds are the same feeds used in test#2. The third feed is published in ***I PROGRAMMER*** website that has a group of experts who post articles, tutorials, and books' reviews about different areas of programming including mobile applications development.

User Evaluation:

The user's and the method's top three articles exactly matched. The same top three articles as in test#1 and test#2 were selected. One of the remaining 2 articles selected by the method was very interesting to the user but not amongst the user's top 5. The last article selected by the method was not interesting to the user as it was a news articles about mobile applications' users.

Precision = # (relevant Items Retrieved) / #(Retrieved Items)
Precision = 4/5 = 0.8

5.3.4 Test No. 4

In the last test a feed from the category "Mobile Application Development" was selected and a feed from the category "News". The first feed is the same one used test#1. The second feed is the CNN America edition RSS feed.

User Evaluation:

Results are exactly as of test#1. This is expected as CNN news RSS feed is not relevant to technology in general neither to mobile applications in particular so mainly articles were retrieved from the first RSS feed.

Precision = # (relevant Items Retrieved) / #(Retrieved Items)
Precision = 5/5 = 1

Discussion

The proposed method exhibited high ability in selecting relevant articles from single and multiple RSS feeds. It also successfully excluded noise or irrelevant knowledge that was not interesting to the user. When applying the method to RSS feeds with number of articles between 10 and 20, the method successfully and repeatedly from test#1 through test#4 selected top three articles exactly as selected by the user of the concept map. The remaining 2 articles selected by the method were found interesting most of the time by the user. In 2 of the tests precision value of the generated results was 100%, where it was 80% in the other two test as shown in figure 5.4. The method repeatedly never selected any of the articles that were highlighted by the user as very uninteresting.

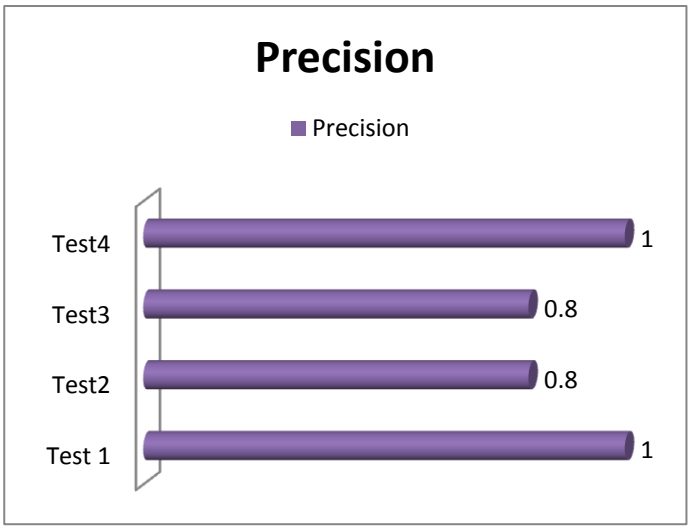


Figure 5.4 Precision Values for Users Test

CHAPTER 6

CONCLUSION AND FUTURE WORK

The problem of instantly updating users on new knowledge specific to their interests is not completely solved; there is still room for improvement. In this chapter, some suggestions for future research to improve the performance of the proposed method are discussed in section 6.1. Furthermore, section 6.2 summarizes the contributions of this thesis to the area of knowledge management on the web.

6.1 Future Work

In this thesis the objective of instantly updating web users on newly created knowledge that is tailored to their specific interests was accomplished as shown by the experimental results in chapter 6. However, the proposed method by no means exhausted the subject. Further efforts can enhance the precision and the performance of the proposed method:

- User Interest Definition : using concept map as a tool to define knowledge interest of single users proved to be effective. However, if user didn't create good concept map then the proposed method won't be as precise in retrieving relevant knowledge. Some efforts can be made to help users create good concept maps. Concept suggestion was proposed to deal with user's selection of good concepts further efforts can be made on *topology analyzers correctors* to help in creating concept maps that are more clearly and distinctly representative of user's knowledge interests.

- Sources of Knowledge : the proposed method can be generalized to account for other sources of knowledge such as forums, wikis and blogs.

- Custom Fuzzy Thesauri : the proposed custom Fuzzy thesauri proved to be effective in measuring partial similarity (i.e. dealing with ill defined information) and excluding irrelevant knowledge. However, in our

experiment it was built using one article for every concept. A larger number of articles for every concept would have increased accuracy and precision of results.

6.2 Conclusion

Internet has become a major source of knowledge for almost everybody. High precision and timeliness of results are critical to web surfers. A typical web surfer like in every web search to have every result on the first page to be relevant. Search engines provides more precise results every day but are ranked based on millions of users. A more personalized ranking would provide more relevant knowledge for every single user.

The proposed method attempted to provide a solution for web users who continuously look for new knowledge in specific area of their interests. RSS feed is number one tool used on the web to get instant updates of newly created contents. Considering RSS feed as a source of newly created knowledge on the web the proposed method attempts to combine the semantic and structure of concept maps along with the logic of Fuzzy Set IR Model to update users of the web on relevant knowledge as it becomes available.

Concept map was chosen as it can define not only concepts in the mind of the user but also it can define somehow the level of importance a user have for every concept. Fuzzy Set IR Model proved to work well with partially related semantics and hence was selected to deal with ill-defined sentences and information.

Experimental results show that the proposed method is consistent in retrieving relevant knowledge for a single concept map. The partial factorial experiment conducted on two replicas show that the method's results are not significantly affected by different parameters value hence results are more influenced by the semantic and structure of the concept map. Comparing method's results to experts' show that the results generated using all topological models (i.e. HARD, CRD and PC) are consistent. Finally in order to test the precision of the results a user test was conducted at 4 stages with an RSS feed relevant to mobile application development, an RSS feed generally related to technology, an RSS feed

generally related to software development and an irrelevant RSS feed. Precision in the four stages ranged between 100% and 80% which is relatively very high.

There is always a room for improvements. Some efforts can be put on top of this work to enhance the quality of concept map created by the user, consider other sources of knowledge and deepen the proposed fuzzy thesauri to allow for higher precision.

REFERENCES

[1] David Grossman, Ophir Frieder. "Retrieval Startegies" in *Information Retrieval Algorithms and Heuristics,* 2nd ed. Netherlands: Springer, 2004, pp. 9-91.

[2] Watraru Sunayama, Yukio Osawa, Masahiko Yachida. "Serach Interface for Query Restructuring with Discovering User Interest" , in *IEEE Third International conference on Knowledge-Based Intelligent Information Engineering Systems,* Adelaida, Australia: IEEE, 1999 , pp.538-541.

[3] David Leak, Ana Maguitman, Thomas Reichherzer. "GOOGLING" From a Concept Map: Towards Automatic Concept-Map-Based Query Formation", in *Concept Maps: Theory, Methodology, Technology. Proceedings of the First International Conference on Concept Mapping,* Pamplona, Spain: Universidad Pública de Navarra, 2004, pp. 409-416.

[4] Imran A. Zualkernan, Mohammed A. AbuJayyab, Yaser A. Ghanam, "An Alignment Equation for Using Mind Maps to Filter Learning Queries from Google" , *in Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06),* The Netherlands, 2006, pp. 153-155.

[5] Ana Gabriela Maguitman, David B. Leake, Thomas Reichherzer, Filippo Menczer. "Dynamic Extraction Topic Descriptors and Discriminators: towards automatic Context-Based Topic Search" *ACM-CIKM International Conference on Information and Knowledge Management,* Washington DC, USA, 2004, pp. 463-472

[6] Brian, Kelly. "RSS - More Than Just News Feeds" *New Review of Information Networking,* vol. 11, 2005

[7] CNN.com, *http://rss.cnn.com/rss/cnn_latest.rss*, accessed on 21/Oct.2010

[8] RSSOwl, *http://www.rssowl.org*, accessed on 21/Oct.2010

[9] Google SOAP Search API , *http://code.google.com/intl/ar/apis/soapsearch/reference.html,* accessed on 1/10/2010

[10] Joseph Novak, Alberto Canas, "The Theory Underlying Concept Maps and How to Construct Them", *Technical Report IHMC CmapTools 2006-01,* Florida, USA: Florida Institute for Human and Machine Cognition (IHMC), 2006.

[11] Alberto J. Cañas, David B. Leake, Ana Gabriela Maguitman. "Combining Concept Mapping with CBR: Towards Experience-Based Support for Knowledge Modeling", *in Fourteenth International Florida Artificial Intelligence Research Society Conference.* Florida , USA: AAAI Press, 2001, pp.286 - 290.

[12] David Leak, Ana Maguitman Thomas Reichherzer. "Understanding Knowledge Models : Modelling Assesment of Concept Importance in Concept Maps", *in 26th*

Annual Conference of the cognitive Science Society, New Jersey: Lawrence Erlbaum, 2004, pp.785-790.

[13] Amy Langville, Carl Meyer. "the HITS Method For Ranking Web Pages" in *Googlee's Page Rank and Beyond The Science of Search Engine Rankings.* 1st ed. Princeton University Press, 2006, pp. 115-126.

[14] Yasushi Ogawa, Tetsuya Morito, Kiyojiko Kobayashi. "A Fuzzy Document Retrieval System Using The Keyword Connection Matrix and a Learning Method", *in Fuzzy Sets and Systems,* 39(2), 1991, pp. 163-179.

[15] Rajiv Yerra, Yiu-Kai Ng. "Detecting Similar HTML Documents Using a Fuzzy Set Information Retrieval Approach", *in Granular Computing, 2005 IEEE International Conferenc,* 2005, pp. 693 - 699.

[16] Porter., Martin. "The Porter Stemming Algorithm", Jan 2006, in *Martin Porter's Home Page,* accessed in April 2010, url: <http://tartarus.org/~martin/PorterStemmer/>.

[17] Ng, Chris Tseng and Patric. "Precisiated Information Retrieval for RSS Feeds", in *Information Management & Computer Security,* 15(3), 2007, pp. 184-200.

[18] Nathenal Gustafson, Maria Pera and Yiu-Kai Ng. "Generating Fuzzy Equavelace Classes on RSS News Articles for Retrieving Correlated Information" , in *International Conference on Computational Science and Applications,* Berlin, Heidelberg: Springer-Verlag, 2008, pp.232-247.

[19] Maria Pera and Yiu-Kai Ng. " Utilizing Phrase-Similarity Measures For Detecting And Clustering Informative RSS News Articles" , *in Integrated Computer-Aided Engineering,* vol. 15, 2008, pp. 331–350

[20] C. Demaio, G. Fenza, V. Loia and S. Senatore. "Ontology-Based Knolwedge Structuring: An Application on RSS Feeds", i*n IEEE - 2nd International Conference on Human System Interaction,* Catania, Italy, 2009, pp.467-473.

[21] Christopher Manning, Prabhakar Raghavan. "Ch1, Ch6, Ch8, Ch21" *in Introduction to Information Retrieval,* USA: Campridge University Press, 2008.

[22] Amit, Singhal. "Modern Information Retrieval: A Brief Overview", *IEEE Data Engineering Bulletin,* vol. 24, pp. 35-43, 2001

[23] C. J. van Rijsbergen, "Chapter 1" , in *Information Retrieval ,* 2nd ed , Department of Computing Science University of Glasgow, http://akira.ruc.dk/~jv/KIIS2004/ir1_2.pdf, accessed on April 2010

[24] Thomas Jech, "Set Theory", *published Thu Jul 11, 2002,* http://plato.stanford.edu/entries/set-theory/, accessed on April, 2010.

[25] Zadeh, "Fuzzy Sets.", *in Information and Control* , vol. 8, 1965, pp. 338-353.

[26] Ian Garcia, "Eliminating Redundant And Less-Informative RSS News Articles Based On Word Similarity And A Fuzzy Equivalence Relation", M.Sc. Dissertation, Brigham Young University, 2007

[27] yFiles for Java by *yWorks GmbH*, http://www.yworks.com/yfiles

[28] Apache Lucene by the *Apache Software Foundation* , http://www. lucene.apache.org/

[29] Wikixmlj by *Delip*, http://code.google.com/p/wikixmlj/

[30] ROME by *JAVA.Net*, http://wiki.java.net/bin/view/Javawsxml/Rome

[31] Ronald Walpole, Raymond Myers, "Factorial Experiments" in *Probabilities and Statistics for Engineers and Scientists,* 2nd ed. USA: Macmillan Publishing, 1978.

[32] Fleiss, Joseph. "CH 18", *in Statistical methods for rates and proportion*,. 3rd ed. Hoboken, New Jersey: John Wiley & Sons Inc, 2003, pp. 598-621.

APPENDIX A

JAVA Program Source Code

```
1.  // =============== [ Main Form ] ================
2.  package org.apache.lucene.analysis;
3.  import java.awt.event.ActionEvent;
4.  import java.awt.event.ActionListener;
5.  import java.io.BufferedWriter;
6.  import java.io.FileNotFoundException;
7.  import java.io.FileOutputStream;
8.  import java.io.IOException;
9.  import java.io.OutputStreamWriter;
10. import javax.swing.AbstractButton;
11. import javax.swing.JButton;
12. import javax.swing.JComboBox;
13. import javax.swing.JFileChooser;
14. import javax.swing.JFrame;
15. import javax.swing.JLabel;
16. import javax.swing.JPanel;
17. import javax.swing.JTextField;
18. public class MainForm implements ActionListener {

19. MainForm(){

20. JFrame frame;
21. JTextField textField;
22. JComboBox combo;
23. String graphmlpath = "";

24. frame = new JFrame("");
25. JPanel mainPanel = new JPanel();

26. JButton CreateRSS= new JButton(" (1) Create Disctionary", null);
27. JButton SetupRSS= new JButton(" (2)Setup RSS Feeds", null);
28. JButton ViewUpdate= new JButton(" (3) View Recent Update", null);


29. CreateRSS.addActionListener(this);
30. SetupRSS.addActionListener(this);
31. ViewUpdate.addActionListener(this);


32. mainPanel.add(CreateRSS);
```

```
33.     mainPanel.add(SetupRSS);
34.     mainPanel.add(ViewUpdate);

35.     frame.add( mainPanel);
36.     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37.     frame.setSize(400,400);
38.     frame.setVisible(true);
39.     }//

40.     @Override
41.     public void actionPerformed(ActionEvent actionEvent)
42.     {
43.     AbstractButton aButton = (AbstractButton) actionEvent.getSource();
44.     String label = aButton.getText();

45.     if(label.equals(" (1) Create Disctionary"))
46.     {
47.     CreateDictionaryForm uirss= new CreateDictionaryForm();
48.     }
49.     else if(label.equals(" (2)Setup RSS Feeds"))
50.     {

51.     SetupRssForm i= new SetupRssForm();
52.     }
53.     else if(label.equals(" (3) View Recent Update"))
54.     {
55.     viewRecentUpdateForm vruf= new viewRecentUpdateForm();
56.     }

57.     }//actionPerformed

58.     public static void main(String[] args) throws Exception
59.     {



60.     MainForm mf= new MainForm();

61.     }//main
62.     }//MainForm
63.     // ================== [ Export Custom Text Corpora] ==================
```

```
64.  package org.apache.lucene.analysis;


65.  import java.net.*;
66.  import java.io.*;
67.  import edu.jhu.nlp.util.FileUtil;
68.  import edu.jhu.nlp.wikipedia.WikiTextParser;


69.  public class ExportCorpora{
70.  String userCategory;
71.  String []graphmlContent;
72.  int docNum;
73.  int noOfPages;

74.  ExportCorpora(String path, int nopage)throws Exception
75.  {
76.  //CreateRadioButton();
77.  noOfPages= nopage;
78.  ReadGraph rg=new ReadGraph(path);
79.  graphmlContent=rg.getGraphmlContent();

80.  try {
81.  URL();
82.  } catch (Exception e) {
83.  // TODO Auto-generated catch block
84.  e.printStackTrace();
85.  }

86.  StemCorpora tst = new StemCorpora();
87.  try {
88.  tst.testDefaults();
89.  tst.WordCounter();
90.  } catch (IOException e) {
91.  // TODO Auto-generated catch block
92.  e.printStackTrace();
93.  }
94.  }

95.  void URL()throws Exception
```

```
96.     {


97.     docNum=1;
98.     for(int i=0;i<graphmlContent.length;i++){
99.     graphmlContent[i]=graphmlContent[i].replace(" ", "%2B");
100.    System.out.print(graphmlContent[i]+"@");
101.    String
url="http://en.wikipedia.org/w/index.php?title=Special:Search&redirs=1&search="+graphmlContent[i]+
"&fulltext=Search&ns0=1";


102.    String inputLine=getxmlPageContent(url);
103.    //System.out.print(inputLine);
104.    GetPageTitles(inputLine);
105.    }//for i
106.    }//url
107.    void GetPageTitles(String s)
108.    {
109.    System.out.println("######################");
110.    String str="mw-search-results";
111.    int Start= s.indexOf(str);
112.    if(Start!=-1){
113.    String str2="mw-search-pager-bottom";
114.    int End=s.indexOf(str2);
115.    //System.out.print(Start+"#"+End+"\n");
116.    String Titles=s.substring(Start, End);
117.    String extitle=ExtractTitles(Titles);
118.    //System.out.print("\n"+extitle);
119.    ////////////////////////////
120.    extitle=extitle.replace(" ", "_");
121.    String []Title=extitle.split("\n");
122.    int loopControl = 0;


123.    if(Title.length>0){
124.    /*if (Title.length < 2)
125.    loopControl =Title.length;
126.    else
127.    loopControl =1;*/
```

```
128.  for(int i=0;i<noOfPages;i++)
129.  {//System.out.print(Title[i]);

130.  try {
131.  OutputStreamWriter  fstream= new OutputStreamWriter(new FileOutputStream("./Wikipedia
Dump/Doc"+docNum+".xml"),"UTF-8");


132.  BufferedWriter out = new BufferedWriter(fstream);

133.  String url="http://en.wikipedia.org/wiki/Special:Export/"+Title[i];

134.  URL wikipedia = new URL(url);


135.  URLConnection yc = wikipedia.openConnection();
136.  BufferedReader in = new BufferedReader(new InputStreamReader(yc.getInputStream()));
137.  String inputLine="";



138.  int c;
139.  while ( (c=in.read()) != -1)
140.  {
141.  out.append((char)c);

142.  }// end while

143.  in.close();
144.  out.close();


145.  docNum++;
146.  } catch (IOException e) {
147.  // TODO Auto-generated catch block
148.  e.printStackTrace();
149.  }//catch
```

```
150.    }//for
151.    }//if


152.    }//if start !=-1
153.    }//GetPageTitles
154.    String ExtractTitles(String s)
155.    {
156.    String Result="";
157.    while (s.contains("<a")==true)
158.    {
159.    int start=s.indexOf("title=")+7;
160.    int end=s.indexOf("\"",start);
161.    Result+=(s.substring(start, end)+"\n");
162.    s=s.substring(end);
163.    }//while
164.    return Result;
165.    }


166.    String demoGetText(String wikiFile) {
167.    String wikiText = FileUtil.readFile(wikiFile);
168.    WikiTextParser wtp = new WikiTextParser(wikiText);
169.    return wtp.getPlainText();
170.    //return wtp.getText();
171.    }//end


172.    String getxmlPageContent(String url) throws IOException
173.    {
174.    URL wikipedia = new URL(url);


175.    URLConnection yc = wikipedia.openConnection();
176.    BufferedReader in = new BufferedReader(new InputStreamReader(yc.getInputStream()));
177.    String inputLine="";
178.    String getstr;
179.    while ((getstr = in.readLine()) != null)
180.    {
181.    inputLine+=(getstr+"\n");
```

```
182.    }
183.    in.close();
184.    return inputLine;


185.    }//getxmlPageContent
186.    /*
187.    public static void main(String[] args) throws Exception {

188.    ExportCorpora EC= new ExportCorpora();

189.    }
190.    */
191.    }//end class
192.    //============= [ Stem Corpora ] ============
193.    package org.apache.lucene.analysis;

194.    // Lucene packages
195.    import org.apache.lucene.analysis.tokenattributes.TermAttribute;
196.    import org.apache.lucene.util.Version;
197.    //wikixmlJ package
198.    import edu.jhu.nlp.util.FileUtil;
199.    import edu.jhu.nlp.wikipedia.WikiTextParser;

200.    //File Processing and HashTable
201.    import java.io.BufferedReader;
202.    import java.io.BufferedWriter;
203.    import java.io.File;
204.    import java.io.FileInputStream;
205.    import java.io.FileNotFoundException;
206.    import java.io.FileOutputStream;
207.    import java.io.FileReader;
208.    import java.io.FileWriter;
209.    import java.io.ObjectInputStream;
210.    import java.io.ObjectOutputStream;
211.    import java.io.StringReader;
212.    import java.io.IOException;
213.    import java.util.Enumeration;
214.    import java.util.HashMap;
215.    import java.util.Hashtable;
```

```
216.  import java.util.Set;
217.  import java.util.HashSet;
218.  import java.util.Vector;


219.  // Define the Class
220.  public class StemCorpora extends CorporaTestCase {

221.  //create 2 vectors , first vector for each word in every document along with the freq and
positions ,
222.  //the second vector stores distinct words in whole corpora
223.  private Vector <VecFCtable> WordsVector= new Vector <VecFCtable> ();
224.  private Vector <String> uniquWords= new Vector <String> ();
225.  //create a vector of class worddoc to store unique words and their associated docs and pos
226.  private Vector <VecWorddoc> WordDoc  = new Vector <VecWorddoc>();

227.  //constructor
228.  public StemCorpora() {


229.  }


230.  //============== [ find the positions of the stemmed words in the stemmed text file ]
======================
231.  public int [] wordindex(String word,String text,int foc)
232.  {

233.  int indexes[]=new int[foc];


234.  //-----------------calculate position as a rank----------
235.  // Count the distinct words in the fileContents String
236.  // First dump all the words into an array of strings
237.  // Note that the \\s will split on whitespace (space, tab, newline)
238.  String[] wordList = text.split("\\s");
239.  // The words array has been created for us by the split
240.  // and we need to know how many slots have been made in it
241.  int size = wordList.length;
```

```
242.    int j = 0;

243.    for(int i=0 ; i<size; i++)
244.    {
245.    if(word.equals(wordList[i])==true)
246.    {
247.    indexes[j]=i;
248.    j++;
249.    }//end if

250.    }//end for loop i
251.    return indexes;
252.    }//wordindex

253.    //==============[ parse wiki text ]==============================
254.    public static String demoGetText(String wikiFile) {
255.    String wikiText = FileUtil.readFile(wikiFile);
256.    WikiTextParser wtp = new WikiTextParser(wikiText);
257.    return wtp.getPlainText();
258.    //return wtp.getText();
259.    }//end

260.    // ============= [ Remove Stop Words and do Stemming then Create New files called
stem*.txt ] =====================
261.    public void testDefaults() throws IOException {


262.    System.out.println(java.lang.Runtime.getRuntime().maxMemory());
263.    //store the number of wikipages to process
264.    File[] wikiPagesList = new File("./Wikipedia Dump").listFiles();
265.    int NumberOfWikiPages = wikiPagesList.length;

266.    // == Builds an analyzer which removes words in ENGLISH_STOP_WORDS_SET.
267.    Set inValidTokens = new HashSet();


268.    String [] arr = {"a", "about", "above", "across", "after", "afterwards", "again", "against", "all",
"almost", "alone", "along", "already", "also","although","always","am","among", "amongst", "amoungst",
"amount",  "an", "and", "another", "any","anyhow","anyone","anything","anyway", "anywhere", "are",
"around", "as",  "at", "back","be","became", "because","become","becomes", "becoming", "been",
```

"before", "beforehand", "behind", "being", "below", "beside", "besides", "between", "beyond", "bill", "both", "bottom","but", "by", "call", "can", "cannot", "cant", "co", "con", "could", "couldnt", "cry", "de", "describe", "detail", "do", "done", "down", "due", "during", "each", "eg", "eight", "either", "eleven","else", "elsewhere", "empty", "enough", "etc", "even", "ever", "every", "everyone", "everything", "everywhere", "except", "few", "fifteen", "fify", "fill", "find", "fire", "first", "five", "for", "former", "formerly", "forty", "found", "four", "from", "front", "full", "further", "get", "give", "go", "had", "has", "hasnt", "have", "he", "hence", "her", "here", "hereafter", "hereby", "herein", "hereupon", "hers", "herself", "him", "himself", "his", "how", "however", "hundred", "ie", "if", "in", "inc", "indeed", "interest", "into", "is", "it", "its", "itself", "keep", "last", "latter", "latterly", "least", "less", "ltd", "made", "many", "may", "me", "meanwhile", "might", "mill", "mine", "more", "moreover", "most", "mostly", "move", "much", "must", "my", "myself", "name", "namely", "neither", "never", "nevertheless", "next", "nine", "no", "nobody", "none", "noone", "nor", "not", "nothing", "now", "nowhere", "of", "off", "often", "on", "once", "one", "only", "onto", "or", "other", "others", "otherwise", "our", "ours", "ourselves", "out", "over", "own","part", "per", "perhaps", "please", "put", "rather", "re", "same", "see", "seem", "seemed", "seeming", "seems", "serious", "several", "she", "should", "show", "side", "since", "sincere", "six", "sixty", "so", "some", "somehow", "someone", "something", "sometime", "sometimes", "somewhere", "still", "such", "system", "take", "ten", "than", "that", "the", "their", "them", "themselves", "then", "thence", "there", "thereafter", "thereby", "therefore", "therein", "thereupon", "these", "they", "thickv", "thin", "third", "this", "those", "though", "three", "through", "throughout", "thru", "thus", "to", "together", "too", "top", "toward", "towards", "twelve", "twenty", "two", "un", "under", "until", "up", "upon", "us", "very", "via", "was", "we", "well", "were", "what", "whatever", "when", "whence", "whenever", "where", "whereafter", "whereas", "whereby", "wherein", "whereupon", "wherever", "whether", "which", "while", "whither", "who", "whoever", "whole", "whom", "whose", "why", "will", "with", "within", "without", "would", "yet", "you", "your", "yours", "yourself", "yourselves", "the"};

```
269.    for (int i = 0 ; i< arr.length ; i++)
270.    inValidTokens.add(arr[i]);

271.    StopAnalyzer stop = new StopAnalyzer(Version.LUCENE_CURRENT, inValidTokens);

272.    //loop through all the wiki articles and process  them
273.    for(int i=1;i<=NumberOfWikiPages;i++){
274.    assertTrue(stop != null);
275.    //=====read each file in the wiki dump
276.    StringReader reader = new StringReader(demoGetText("./Wikipedia Dump/Doc"+i+".xml"));
277.    //===== remove  stop words
278.    TokenStream stream = stop.tokenStream("test", reader);
279.    //====== stemming
280.    stream = new PorterStemFilter(stream);
281.    assertTrue(stream != null);
282.    TermAttribute termAtt = stream.getAttribute(TermAttribute.class);
```

```
283.  // ================write to a file / create files ==========


284.  //create file for stemmed words
285.  FileWriter Pstream = new FileWriter("./wikiStemmed/stem"+i+".txt");
286.  BufferedWriter stem = new BufferedWriter(Pstream);

287.  //=====  start writing  =====
288.  int counter =0; // we want to omit the first 43 tokens because they are related to wikipedia
format and not to a specific articles
289.  while (stream.incrementToken()) {

290.  if (counter > 43){

291.  stem.append(termAtt.term()+" ");
292.  }// end if
293.  counter ++;
294.  }//end while


295.  //Close the output stream

296.  stem.close();

297.  }// end for loop

298.  }//end Function test Defaults

299.  // ================ [ Count distinct words and calculate frequency ]
==================
300.  public void WordCounter() throws IOException {


301.  //store the number of wikipages to process
302.  File[] wikiPagesList = new File("./Wikipedia Dump").listFiles();
303.  int NumberOfWikiPages = wikiPagesList.length;

304.  wikiPagesList = null;
```

```
305.    //loop through all documents in the corpora   $$$ h<28
306.    for(int h=1;h<=NumberOfWikiPages;h++){  // $$ 28 the loop stop number
307.    // The following code reads in the contents from a file
308.    // then counts all the words in the file taking into
309.    // account duplicates.
310.    String filename = "./wikiStemmed/stem"+h+".txt"; // Relative pathname -- place in project
directory
311.    String fileContents = ""; // String to hold the contents eventually

312.    try {
313.    FileReader fr = new FileReader(filename);
314.    BufferedReader br = new BufferedReader(fr);
315.    String lineRead = br.readLine();
316.    while(lineRead != null){
317.    fileContents = fileContents + " " + lineRead;
318.    lineRead = br.readLine();
319.    }//end try
320.    // Close the file after reading
321.    br.close();
322.    } catch (FileNotFoundException e) {
323.    e.printStackTrace(); // Display the error
324.    System.exit(1); // Stop the program from running
325.    } catch (IOException e) {
326.    e.printStackTrace();
327.    System.exit(1);
328.    }

329.    // Count the distinct words in the fileContents String
330.    // First dump all the words into an array of strings
331.    // Note that the \\s will split on whitespace (space, tab, newline)
332.    String[] wordList = fileContents.split("\\s");
333.    // The words array has been created for us by the split
334.    // and we need to know how many slots have been made in it
335.    int size = wordList.length;
336.    // Create two (parallel) arrays for counting the distinct words
337.    // Call these arrays words and counts.
338.    String [] words;
339.    int [] counts;
340.    words = new String [size];
341.    counts = new int [size];
```

```
342.  //reset counter to zero
343.  for (int k=0 ; k < size ; k++ )
344.  {
345.  counts[k]=0;

346.  }//end for

347.  // Go through each word in the word list and count it
348.  //
349.  // First see if the word is on the list of words you have already seen.
350.  // Note that when comparing Strings, you should not use (str1 == str2)
351.  // Instead you should use (str1.equals(str2))
352.  //
353.  // If it is not on the list then add it to the end and put in a count of 1
354.  // on the counts list at the end as well.
355.  // Otherwise, increase the word's associated count by 1

356.  for(int i = 0; i < size; i++){
357.  words[i] = wordList[i];
358.  for(int j = 0; j < size; j++){
359.  if(words[i].equals (wordList[j])){
360.  counts[i] = counts[i] +1;
361.  }//end if
362.  }//end inner for

363.  //initialize an array for  positions
364.  int arr[]= wordindex(words[i],fileContents,counts[i]);
365.  //add each word in the list to the vector along with the other details as in the argument list
366.  WordsVector.addElement(new VecFCtable(words[i],("Doc"+h),counts[i],arr));

367.  String positions = "";

368.  for ( int p=0 ; p< arr.length ; p++)
369.  positions = positions + " " + arr[p];


370.  }//end outer for

371.  } //end the very outer for
```

```
372.  // ---- create a vector ,string. of distinct words in the whole corpora ----
373.  for(int j=0;j<WordsVector.size();j++)
374.  if(newString(uniquWords,WordsVector.elementAt(j).word)==true) // call function newString to
verify the uniqueness of a String
375.  {
376.  if( WordsVector.elementAt(j).word.length()>1)
377.  uniquWords.addElement(WordsVector.elementAt(j).word) ;
378.  }//if


379.  //---------- copy unique words into the new vector WordDoc----------
380.  // assign all unique words to the vector worddoc
381.  for(int i=0;i<uniquWords.size();i++)
382.  {
383.  WordDoc.add(new VecWorddoc(uniquWords.elementAt(i))) ;
384.  }//end for

385.  //-------assign doc list along with pos in each doc to every unique word
386.  for(int i=0; i<uniquWords.size();i++) // loop through  unique words across the corpora
387.  for(int j=0;j<WordsVector.size();j++) // loop through all words in each document

388.  // when words match/ retrieve the associated Doc ID and Pos
389.  if(WordDoc.elementAt(i).word.equals(WordsVector.elementAt(j).word)==true)
390.  {
391.  //check if a single DocID was retrieved already, if not then retrieve ID and Pos
392.  if(WordDoc.elementAt(i).check(WordsVector.elementAt(j).docID)==true){
393.  WordDoc.elementAt(i).addDoc(WordsVector.elementAt(j).docID);
394.  WordDoc.elementAt(i).addPosition(WordsVector.elementAt(j).pos);
395.  }//if

396.  }//if
397.  // End Loops i , j

398.  WordsVector.clear();
399.  uniquWords.clear();
400.  System.out.println("Created vect");

401.  // Create Map table to store correlation values
```

```
402.  HashMap hashMap = new HashMap();


403.  int cftsize=WordDoc.size();

404.  float corr = 0;

405.  //--------fill the correlation factor table
406.  for(int i=0; i <cftsize;i++) //  i <cftsize
407.  {
408.  for(int j=i; j<cftsize;j++) // i <cftsize
409.  {
410.  if(i==j){ // check if it's the same word then correlation factor is one
411.  corr=1;

412.  }//end if
413.  else{
414.  corr=getCF(WordDoc.elementAt(i).word,WordDoc.elementAt(j).word,i,j);

415.  }//IF else


416.  hashMap.put(WordDoc.elementAt(i).word+ " " + WordDoc.elementAt(j).word, new
Float(corr));
417.  System.out.println(WordDoc.elementAt(i).word+ " " + WordDoc.elementAt(j).word+ " " + corr);
418.  }//end for j

419.  // System.out.println("filling up the hashtables" + "i  " + i);

420.  }//i
421.  WordDoc.clear();

422.  //---------------- STORE HASH TABLE ON THE HARD DISK----------------
423.  try {
424.  FileOutputStream fileOut = new FileOutputStream("CrFcHashMap.ser");
425.  ObjectOutputStream outHASH = new ObjectOutputStream(fileOut);

426.  outHASH.writeObject(hashMap);
427.  hashMap.clear();
428.  outHASH.close();
```

```
429.    fileOut.close();

430.    } catch(FileNotFoundException e) {
431.    e.printStackTrace();
432.    } catch (IOException e) {
433.    e.printStackTrace();
434.    }// try catch

435.    }// end function word counter

436.    //====================== [ calculate the distance correlation factor across all documents]
===============
437.    float getCF(String word1,String word2,int inx1, int inx2)
438.    {



439.    float cf=0;
440.    Vector <VecFCValue> cfDoc=new Vector <VecFCValue>(); // temp vector to store all double
values of nfc
441.    int index1=inx1,index2=inx2;



442.    for(int i=0;i<WordDoc.elementAt(index1).Doc.size();i++)
443.    {
444.    for(int j=0;j<WordDoc.elementAt(index2).Doc.size();j++)
445.    {
446.    //check if wi, wj are not identical
447.    if(WordDoc.elementAt(index1).word.equals(WordDoc.elementAt(index2).word)==false){
448.    //make sure that they exist in the same doc
449.    if(WordDoc.elementAt(index1).Doc.elementAt(i).equals(WordDoc.elementAt(index2).Doc.ele
mentAt(j))==true)
450.    {
451.    //calculate the sum of 1/distance
452.    cf=Calculate(
WordDoc.elementAt(index1).poistion.elementAt(i),WordDoc.elementAt(index2).poistion.elementAt(j));
453.    // normalize the previouse value
454.    cf=Normalize(cf,WordDoc.elementAt(index1).poistion.elementAt(i).length,WordDoc.elementAt
(index2).poistion.elementAt(j).length);
455.    //add the normalized values to vector cfDoc
456.    cfDoc.add( new VecFCValue(cf));
```

```
457.    //reset cf to zero
458.    cf=0;

459.    } //end inner if
460.    }// end outer if
461.    }//end for j
462.    } //end for i

463.    float cfij=0;
464.    // -- sum all the normalized cf between wi,wj in all documents
465.    for(int i=0;i<cfDoc.size();i++)
466.    cfij+=cfDoc.elementAt(i).f;

467.    if (cfDoc.size() != 0) // check if the size of cfDoc is zero then the words have never occured
together in the same doc --> cfij is zero
468.    //devide by the number of common document
469.    cfij=cfij/(float)cfDoc.size();


470.    return cfij;

471.    }//end getCF


472.    //============== [ calculate distance Correlation factor in a single Document]
===========
473.    public float Calculate(int []pos1,int[]pos2)
474.    {
475.    float distance=0;
476.    // count for all possible combinations of positions
477.    for(int i=0;i<pos1.length;i++)
478.    {
479.    for(int j=0;j<pos2.length;j++)
480.    {
481.    // distance is the absolute difference between positions --> retrieve the number of words
between w1,w2
482.    distance+=(Math.pow((((float)Math.abs(pos1[i]- pos2[j]))+1),-1.0));
483.    }//end for j
484.    }//end for i
```

```
485.    return distance;
486.    }//end Calculate

487.    //============== [Normalize correlation factor] ===========
488.    public float Normalize(float cf,int frq1,int frq2)
489.    {

490.    float ncf =0;
491.    //devide cf(i,j) by the the number of occurances of wi times the number of occurances of wj in
a single document
492.    ncf=(cf/(float)(frq1*frq2));

493.    return ncf;
494.    }

495.    //end Normalize

496.    //============ [ test whether 2 words are identical or not ] =========
497.    public boolean newString(Vector <String> words,String str)
498.    {
499.    for(int i=0;i<words.size();i++)
500.    if(words.elementAt(i).equalsIgnoreCase(str)==true)
501.    return false;

502.    return true;
503.    }//newString
504.    }//end class

505.    // ============= [ Create Custom Dictionary ] ====================
506.    package org.apache.lucene.analysis;
507.    import java.awt.Color;
508.    import java.awt.GridLayout;
509.    import java.awt.event.ActionEvent;
510.    import java.awt.event.ActionListener;
511.    import java.io.BufferedWriter;
512.    import java.io.File;
513.    import java.io.FileNotFoundException;
514.    import java.io.FileOutputStream;
515.    import java.io.IOException;
516.    import java.io.OutputStreamWriter;
```

```
517.  import javax.swing.AbstractButton;
518.  import javax.swing.ButtonGroup;
519.  import javax.swing.JButton;
520.  import javax.swing.JComboBox;
521.  import javax.swing.JFileChooser;
522.  import javax.swing.JFrame;
523.  import javax.swing.JLabel;
524.  import javax.swing.JPanel;
525.  import javax.swing.JRadioButton;
526.  import javax.swing.JTextField;
527.  public class CreateDictionaryForm implements ActionListener {
528.  JFrame frame;
529.  JTextField textField;
530.  JComboBox combo;
531.  String graphmlpath = "";
532.  CreateDictionaryForm(){
533.  frame = new JFrame("Create Dictionary");
534.  JPanel mainPanel = new JPanel();
535.  JPanel panel = new JPanel();
536.  JPanel paneltwo = new JPanel();
537.  JLabel label = new JLabel("Select your graphml:");
538.  JButton ok= new JButton("Create Dictionary", null);
539.  JButton browes= new JButton("Browes", null);
540.  textField = new JTextField(40);
541.  // String filename = File.separator+"tmp";

542.  String noOfpage[] = {"1","2","3","4","5"};

543.  combo = new JComboBox(noOfpage);
544.  //create folders
545.  createDirctory("./Wikipedia Dump");
546.  createDirctory("./wikiStemmed");
547.  createDirctory("./Resources");

548.  browes.addActionListener(this);
549.  ok.addActionListener(this);
550.  // combo.setBackground(Color.gray);
551.  // combo.setForeground(Color.red);
```

```
552.    panel.add(label);
553.    panel.add(textField);
554.    panel.add(browes);
555.    // panel.setLayout(new GridLayout(1,2));


556.    paneltwo.add(combo);
557.    paneltwo.add(ok);


558.    mainPanel.add(panel);
559.    mainPanel.add(paneltwo);
560.    //mainPanel.setLayout(new GridLayout(2,0));

561.    frame.add( mainPanel);
562.    frame.setSize(700,150);
563.    frame.setVisible(true);
564.    }//UIRS

565.    @Override
566.    public void actionPerformed(ActionEvent actionEvent)
567.    {
568.    AbstractButton aButton = (AbstractButton) actionEvent.getSource();
569.    String label = aButton.getText();

570.    if(label.equals("Browes"))
571.    {
572.    // Show open dialog; this method does not return until the dialog is closed
573.    JFileChooser fc = new JFileChooser();
574.    fc.addChoosableFileFilter(new graphmlFilter());
575.    fc.showOpenDialog(frame);
576.    String FileName = fc.getSelectedFile().getPath();
577.    textField.setText(FileName);
578.    graphmlpath=textField.getText();


579.    try {
```

```
580.  OutputStreamWriter fstream2;
581.  fstream2 = new OutputStreamWriter(new FileOutputStream("./Resources/Path.txt"));
582.  BufferedWriter out2 = new BufferedWriter(fstream2);
583.  out2.write(graphmlpath);
584.  out2.close();

585.  } catch (FileNotFoundException e) {
586.  // TODO Auto-generated catch block
587.  e.printStackTrace();
588.  } catch (IOException e) {
589.  // TODO Auto-generated catch block
590.  e.printStackTrace();
591.  }

592.  }//if
593.  else if(label.equals("Create Dictionary"))
594.  {
595.  int numofpages=combo.getSelectedIndex()+1;
596.  try {
597.  ExportCorpora EC= new ExportCorpora(graphmlpath,numofpages);
598.  } catch (Exception e) {
599.  // TODO Auto-generated catch block
600.  e.printStackTrace();
601.  }

602.  }//else

603.  }//actionPerformed
604.  void createDirctory(String strDirectoy)
605.  {
606.  try{

607.  File dir= new File(strDirectoy);
608.  deleteDir(dir);
609.  // Create one directory
610.  boolean success = (new File(strDirectoy)).mkdir();
611.  if (success) {
612.  System.out.println("Directory: " + strDirectoy + " created");
613.  }//if
614.  }catch (Exception e){//Catch exception if any
```

```
615.  System.err.println("Error: " + e.getMessage());
616.  }//catch
617.  }//createDirctory

618.  //Deletes all files and subdirectories under dir.
619.  //Returns true if all deletions were successful.
620.  //If a deletion fails, the method stops attempting to delete and returns false.
621.  public static boolean deleteDir(File dir) {
622.  if (dir.isDirectory()) {
623.  String[] children = dir.list();
624.  for (int i=0; i<children.length; i++) {
625.  boolean success = deleteDir(new File(dir, children[i]));
626.  if (!success) {
627.  return false;
628.  }
629.  }
630.  }


631.  // The directory is now empty so delete it
632.  return dir.delete();
633.  }
634.  };
635.  package org.apache.lucene.analysis;
636.  // yFiles Package
637.  // ============= [ Read Graph - Concept Map ] ==============
638.  import y.base.*;
639.  import y.io.*;
640.  import y.view.*;
641.  import java.io.IOException;
642.  import java.io.StringReader;
643.  import java.util.*;
644.  // Lucene packages

645.  import org.apache.lucene.analysis.PorterStemFilter;
646.  import org.apache.lucene.analysis.StopAnalyzer;
647.  import org.apache.lucene.analysis.TokenStream;
648.  import org.apache.lucene.analysis.tokenattributes.TermAttribute;
649.  import org.apache.lucene.util.Version;
```

```
650.  public

651.  class ReadGraph {
652.  //declare vector of type myNode to store information about each node in the graph
653.  String[]graphmlContent;
654.  //Constructor
655.  public ReadGraph(String path){

656.  IOHandler ioh = new GraphMLIOHandler(); // create new graph handler
657.  Graph2D graph=new Graph2D(); // create new 2D graph
658.  try {
659.  // read the sample "Mobile Application . graphml" into 2D graph for later manipulation
660.  ioh.read(graph,path);
661.  }

662.  catch (IOException e1) {
663.  e1.printStackTrace();

664.  //TODO handle
665.  }
666.  Node nodes[]=graph.getNodeArray();
667.  graphmlContent=new String [nodes.length];

668.  for(int i=0;i<nodes.length;i++)
669.  graphmlContent[i]=nodes[i].toString();

670.  }//constructor ReadGraph
671.  String [] getGraphmlContent()
672.  {
673.  return graphmlContent;
674.  }



675.  }//end class
676.  // ============== [ Calculate Graph Weights ] ===========
677.  package org.apache.lucene.analysis;
678.  // yFiles Package
679.  import y.base.*;
680.  import y.io.*;
681.  import y.view.*;
```

```
682.  import java.io.BufferedReader;
683.  import java.io.FileInputStream;
684.  import java.io.FileNotFoundException;
685.  import java.io.IOException;
686.  import java.io.InputStreamReader;
687.  import java.io.StringReader;
688.  import java.util.*;

689.  // Lucene packages
690.  import org.apache.lucene.analysis.PorterStemFilter;
691.  import org.apache.lucene.analysis.StopAnalyzer;
692.  import org.apache.lucene.analysis.TokenStream;
693.  import org.apache.lucene.analysis.tokenattributes.TermAttribute;
694.  import org.apache.lucene.util.Version;


695.  public class ClaculateGraphWeights {

696.  private Set inValidTokens = new HashSet();
697.  private StopAnalyzer stop = new StopAnalyzer(Version.LUCENE_CURRENT,
inValidTokens);

698.  //declare  vector of type myNode to store information about each node in the graph
699.  Vector <myNode> myNodes=new Vector <myNode>();

700.  //Constructor
701.  public ClaculateGraphWeights(){
702.  IOHandler ioh = new GraphMLIOHandler(); // create new graph handler
703.  Graph2D graph= new Graph2D(); // create new 2D graph
704.  try {


705.  // read the sample "Mobile Application . graphml" into 2D graph for later manipulation
706.  ioh.read(graph,readthePath());
707.  }
708.  catch (IOException e1) {
709.  e1.printStackTrace();
710.  //TODO handle
711.  }
```

```
712.  Node nodes[]=graph.getNodeArray(); // declare node array
713.  Edge edge[]=graph.getEdgeArray();       // declare edge array

714.  double nop=0; // store number of paths to the root
715.  int l=0;        // store number of levels from the root

716.  String tmp = ""; // temporarily hold temmed labels
717.  inValidTokens.add("XYZFH"); // dummy value for StopAnalyzer


718.  //loop through all nodes in the nodes array
719.  for(int i=0;i<nodes.length;i++)
720.  {
721.  nop=NOofPaths(nodes[i]); // find the number of direct and indirect paths from root to a single
node in the array
722.  l=Level(nodes[i]); // find the distance from the root to a single node


723.  // ====store the content of a node in a reader, then stem

724.  //Extract labels from graph
725.  StringReader reader = new StringReader(nodes[i].toString());
726.  //tokenize
727.  TokenStream stream = stop.tokenStream("test", reader);
728.  // stem all the labels of the graph
729.  stream =  new PorterStemFilter(stream);
730.  // extract text from stemmed stream
731.  TermAttribute termAtt = stream.getAttribute(TermAttribute.class);

732.  try {
733.  while (stream.incrementToken()){
734.  tmp = tmp + termAtt.term()+ " ";
735.  }
736.  } catch (IOException e) {
737.  // TODO Auto-generated catch block
738.  e.printStackTrace();
739.  }// end while
```

```
740.    //add node information to a new element in the myNodes vector
741.    //content,incoming links,outgoing links,levels/distance from the root,total num of paths to the
root
742.    myNodes.add(new myNode(tmp,nodes[i].inDegree(),nodes[i].outDegree(),l,nop));
743.    /* $$ System.out.println(myNodes.elementAt(i).content + " " +
744.    myNodes.elementAt(i).incoming+ " " +
745.    myNodes.elementAt(i).outgoing+ " " +
746.    myNodes.elementAt(i).EdgeToRoot+ " " +
747.    myNodes.elementAt(i).pathCount+ " " );
748.    */
749.    tmp = "";
750.    }// end for

751.    // normalize all hub/authority/upper node values in order to start the HITS algo
752.    normalizeWeight(myNodes, 0);
753.    normalizeWeight(myNodes, 1);
754.    normalizeWeight(myNodes, 2);

755.    for(int i=0;i<nodes.length;i++)
756.    {
757.    hupWeight(myNodes, nodes);
758.    authWeight(myNodes,nodes);
759.    upWeight(myNodes,nodes);

760.    //System.out.println(myNodes.elementAt(0).content+" auth weight :
"+myNodes.elementAt(0).auth +" hub weight  "+myNodes.elementAt(0).hub+" Up weight
"+myNodes.elementAt(0).up);

761.    }//i

762.    //$$for(int i=0;i<myNodes.size();i++)
763.    //$$System.out.println(myNodes.elementAt(i).content+" || "+myNodes.elementAt(i).auth +" ||
"+myNodes.elementAt(i).hub+" || "+myNodes.elementAt(i).up);

764.    // ---- Calculate Weights --------
765.    CRD(myNodes);
766.    HARD(myNodes);

767.    // ---- Normalize weights so as to be 0< weight < 1
```

```
768.    NormalizeCRDvalues();
769.    NormalizePCvalues();
770.    NormalizeHARDvalues();

771.    //++++++++++++++
772.    for(int i=0;i<myNodes.size();i++)
773.    System.out.println(myNodes.elementAt(i).content+"|"+myNodes.elementAt(i).CRDvalue+"|"
+myNodes.elementAt(i).HARDvalue +"|" + myNodes.elementAt(i).pathCount);
774.    }//constructor ReadGraph
775.    //===Read The path==
776.    String readthePath()
777.    {

778.    String outRSS="";

779.    InputStreamReader fstream;

780.    try {
781.    fstream = new InputStreamReader  (new FileInputStream("./Resources/Path.txt"));

782.    BufferedReader  in = new BufferedReader (fstream);

783.    try {
784.    int c;

785.    while ( (c=in.read()) != -1)
786.    {
787.    outRSS+=((char)c);

788.    }// end while
789.    } catch (IOException e) {
790.    // TODO Auto-generated catch block
791.    e.printStackTrace();
792.    }
793.    } catch (FileNotFoundException e1) {
794.    // TODO Auto-generated catch block
795.    e1.printStackTrace();
796.    }
```

```
797.    return outRSS;


798.    }//readthePath
799.    // === set values to be 0<Weight<1 =====
800.    void NormalizeHARDvalues()
801.    {
802.    double maxHARD = 0;
803.    for ( int i = 0 ; i < myNodes.size(); i++)
804.    {
805.    if ((myNodes.elementAt(i).HARDvalue) > maxHARD)
806.    maxHARD = myNodes.elementAt(i).HARDvalue;
807.    }// end for i

808.    for ( int i = 0 ; i < myNodes.size(); i++)
809.    {

810.    myNodes.elementAt(i).HARDvalue =(myNodes.elementAt(i).HARDvalue )/ maxHARD;
811.    }// end for i

812.    }// end NormalizeCRDvalues



813.    //=== set values to be 0<Weight<1 =====
814.    void NormalizeCRDvalues()
815.    {
816.    double maxCRD = 0;
817.    for ( int i = 0 ; i < myNodes.size(); i++)
818.    {
819.    if ((myNodes.elementAt(i).CRDvalue) > maxCRD)
820.    maxCRD = myNodes.elementAt(i).CRDvalue;
821.    }// end for i

822.    for ( int i = 0 ; i < myNodes.size(); i++)
823.    {

824.    myNodes.elementAt(i).CRDvalue =(myNodes.elementAt(i).CRDvalue )/ maxCRD;
825.    }// end for i
```

```
826.    }// end NormalizeCRDvalues


827.    //=== set values to be 0<Weight<1 =====
828.    void NormalizePCvalues()
829.    {
830.    double maxPC = 0;
831.    for ( int i = 0 ; i < myNodes.size(); i++)
832.    {
833.    if ((myNodes.elementAt(i).pathCount) > maxPC)
834.    maxPC = myNodes.elementAt(i).pathCount;
835.    }// end for i

836.    for ( int i = 0 ; i < myNodes.size(); i++)
837.    {

838.    myNodes.elementAt(i).pathCount  =(myNodes.elementAt(i).pathCount  )/ maxPC;
839.    }// end for i

840.    }// end NormalizePCvalues

841.    // --------- [ count all the possible paths from a node to the root node ] ----------
842.    int NOofPaths(Node node)
843.    {
844.    int paths=0;
845.    int z=0;


846.    if(node.inDegree()==0)// base condition , if the predecessor is the ROOT then count one path
847.    return 1;
848.    else // predecessor is not the root
849.    {
850.    //store all the predecessor nodes in a cursor
851.    NodeCursor nc=node.predecessors();
852.    //loop through predecessors
853.    while(z<nc.size()){
854.    // for every predecessor node, keep tracking the predecessors until you reach the root -->
base condition
855.    paths+=NOofPaths(nc.node());
856.    z++;
```

```
857.    nc.cyclicNext(); // next node in the cursor
858.    }//while
859.    return paths;
860.    }//else
861.    }//NOofPath

862.    //----- [ count the number of levels away from the root node this is equal to the distance ] ------
863.    int Level(Node node)
864.    {
865.    int distance=Integer.MAX_VALUE;
866.    int z=0;
867.    int temp=0;


868.    if(node.inDegree()==0) // base condition , root node has level of Zero
869.    return 0;
870.    else
871.    {
872.    //store all the predecessor nodes in a cursor
873.    NodeCursor nc=node.predecessors();
874.    //loop through predecessors
875.    while(z<nc.size()){
876.    temp=Level(nc.node());
877.    if(distance>temp)
878.    distance=temp+1;
879.    z++;
880.    nc.cyclicNext();
881.    }//while
882.    return distance;
883.    }//else
884.    }//LEVEL

885.    //----- [ initially hub/auth/up_node have values of one,, they should be normalized so that sum
of the squared value per node equals one] ------
886.    void normalizeWeight(Vector <myNode> Nodes, int choose)
887.    {
888.    double sum=0;

889.    //first -  for every parameter (hub, auth , up_node) sum up the corresponding weight for every
node
```

```
890.    for(int i=0;i<Nodes.size();i++)
891.    {

892.    if(choose==0)
893.    sum+=(Nodes.elementAt(i).auth);

894.    else if(choose==1)
895.    sum+=(Nodes.elementAt(i).hub);

896.    else if(choose==2)
897.    sum+=(Nodes.elementAt(i).up);

898.    }//for i

899.    //Second - assign the square root of the current weight over the total weights of all nodes for
the corresponding parameter
900.    // ---> this will make the sum of the square of all nodes' weights for a specific parameter
(hub.auth,up ) equals 1
901.    for(int i=0;i<Nodes.size();i++)
902.    {

903.    if(choose==0)

904.    Nodes.elementAt(i).auth=(Math.sqrt((Nodes.elementAt(i).auth/sum)));

905.    else if(choose==1)
906.    Nodes.elementAt(i).hub=(Math.sqrt((Nodes.elementAt(i).hub/sum)));

907.    else if(choose==2)
908.    Nodes.elementAt(i).up=(Math.sqrt((Nodes.elementAt(i).up/sum)));

909.    }//for i

910.    }//normalizeWeight



911.    // ----- [ calculate hub weight] ----------
912.    // ------[ hub weight for  node X is the sum of authority weights of nodes Y pointed to by X ] ----
-----
913.    void hupWeight(Vector <myNode> NodesVec,Node nodes[])
```

```
914.    {
915.    int z=0;
916.    double weight=0;
917.    int index=0;

918.    // loop through array of all nodes
919.    for(int i=0;i<nodes.length;i++)
920.    {
921.    //generate all the node successors, nodes pointed to by the current node
922.    NodeCursor nc=nodes[i].successors();
923.    //loop through successors of current node
924.    while(z<nc.size()){

925.    // retrieve the successor's index from the cursor BECAUSE indexes in the vector are the
same as indexes in the graph structure
926.    index=nc.node().index();
927.    //System.out.print(index+" ");
928.    weight+=NodesVec.elementAt(index).auth;

929.    z++;
930.    nc.cyclicNext();
931.    }//while


932.    NodesVec.elementAt(i).hub=weight; // assign the newlly calculated weight
933.    weight=0; // reset weight
934.    z=0; // reset counter
935.    normalizeWeight(NodesVec, 1); // normalize new weight

936.    //System.out.print("hub weight"+myNodes.elementAt(i).hub);
937.    //System.out.println();
938.    }// For i

939.    }//hupWeight

940.    //----- [ calculate auth weight] ----------
941.    //------[ auth weight for  node X is the some of hub weights of nodes Y pointing  to  X ] ---------
942.    void authWeight(Vector <myNode> NodesVec,Node nodes[])
943.    {
944.    int z=0;
```

```
945.    double weight=0;
946.    int index=0;

947.    // loop through array of all nodes
948.    for(int i=0;i<nodes.length;i++)
949.    {
950.    //generate all the node predecessors, nodes pointing to  the current node
951.    NodeCursor nc=nodes[i].predecessors();

952.    //loop through predecessors of current node
953.    while(z<nc.size()&& nc!=null){
954.    // retrieve the predecessor's index from the cursor BECAUSE indexes in the vector are the
same as indexes in the graph structure
955.    index=nc.node().index();
956.    weight+=NodesVec.elementAt(index).hub;

957.    z++;
958.    nc.cyclicNext();
959.    }//while
960.    NodesVec.elementAt(i).auth=weight;
961.    weight=0;
962.    z=0;
963.    normalizeWeight(NodesVec, 0);
964.    //System.out.print("auth weight"+myNodes.elementAt(i).auth);
965.    //System.out.println();
966.    }//i

967.    }//authWeight
968.    // ---- [ upper_node weight ] ----- [ how close a node is to the root] ------
969.    void upWeight(Vector <myNode> NodesVec,Node nodes[])
970.    {
971.    int z=0;
972.    double weight=0.0;
973.    int index=0;

974.    for(int i=0;i<nodes.length;i++)
975.    {
976.    NodeCursor nc=nodes[i].predecessors();
977.    if(nc.size()>0){
978.    while(z<nc.size()){
```

```
979.    index=nc.node().index();
980.    weight+=(NodesVec.elementAt(index).up*NodesVec.elementAt(index).up);

981.    z++;
982.    nc.cyclicNext();
983.    }//while

984.    NodesVec.elementAt(i).up=weight;
985.    weight=0;
986.    z=0;
987.    }//if
988.    else
989.    NodesVec.elementAt(i).up=1;

990.    normalizeWeight(NodesVec, 2);
991.    //System.out.print("upper node weight"+myNodes.elementAt(i).up);
992.    //System.out.println();
993.    }//i

994.    }//upWeight
995.    // ----------- [ CRD ] -------
996.    void CRD(Vector <myNode> NodesVec)
997.    {
998.    // best-fit values
999.    double Alph=9.0;
1000.   double beta=1.0;
1001.   double gama=1.0;

1002.   for(int i=0;i<NodesVec.size();i++)
1003.   {
1004.   //[(Alph*outgoin)+(beta*incoming)]*(1/(level+1))to the power of (1/gama)
1005.   NodesVec.elementAt(i).CRDvalue=((Alph*NodesVec.elementAt(i).outgoing)+(beta*NodesVec
.elementAt(i).incoming))
1006.   *(Math.pow((1.0/(NodesVec.elementAt(i).EdgeToRoot+1.0)), (1.0/(gama))));
1007.   }//for i
1008.   }//CRD

1009.   //------------ [ HARD ] ------
1010.   void HARD(Vector <myNode> NodesVec)
1011.   {
```

```
1012.  // best-fir values
1013.  double Alph=9.0;
1014.  double beta=1.0;
1015.  double gama=1.0;

1016.  for(int i=0;i<NodesVec.size();i++)
1017.  {

1018.  NodesVec.elementAt(i).HARDvalue=(Alph*NodesVec.elementAt(i).hub)
1019.  +(beta*NodesVec.elementAt(i).auth)
1020.  +(gama*NodesVec.elementAt(i).up);
1021.  }//for i
1022.  }//HARD

1023.  }//end class
```

```
1024. package org.apache.lucene.analysis;

1025. public class myNode {
1026. String content;
1027. int incoming;
1028. int outgoing;
1029. int EdgeToRoot;
1030. double pathCount;
1031. double hub;
1032. double auth;
1033. double up;
1034. double CRDvalue;
1035. double HARDvalue;
1036. public myNode(String con,int i,int o,int d,double id)
1037. {
1038. content=con;
1039. incoming=i;
1040. outgoing=o;
1041. EdgeToRoot=d;
1042. pathCount=id;
1043. hub=1;
1044. auth=1;
1045. up=1;
1046. CRDvalue=0;
1047. HARDvalue=0;
```

```
}//Node
}
package org.apache.lucene.analysis;

public class VecFCtable {
String word;
String docID;
int freq;
int pos[];
VecFCtable(String w,String doId,int frq,int po[])
{
word=w;
docID=doId;
freq=frq;
pos=po;

}//fctable
}//fctable

package org.apache.lucene.analysis;

public class VecFCValue {
float f;
//constructor
public VecFCValue(float v)
{
f=v;
}//end constructor

}

package org.apache.lucene.analysis;
import java.util.Vector;
public class VecWorddoc {
String word;
Vector <String> Doc;
Vector <int[]> poistion;
public VecWorddoc(String w)
{
word=w;
Doc= new Vector <String>();
```

```
poistion= new Vector <int[]>();
}
void addDoc(String doc)
{
Doc.add(doc);
}
void addPosition(int [] p)
{
poistion.add(p);
}
boolean check(String str)
{
for(int i=0;i<Doc.size();i++)
if(str.equalsIgnoreCase(Doc.elementAt(i)))
return false;

return true;
}//end check
}//end check

// ========== [ Read RSS ] ==========
package org.apache.lucene.analysis;
import java.io.*;
import java.net.MalformedURLException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.Vector;
import org.apache.lucene.analysis.StopAnalyzer;
import org.apache.lucene.util.Version;
// rome packages
import com.sun.syndication.feed.synd.SyndContent;
import com.sun.syndication.feed.synd.SyndContentImpl;
import com.sun.syndication.feed.synd.SyndEntry;
```

```
1122.  import com.sun.syndication.feed.synd.SyndEntryImpl;
1123.  import com.sun.syndication.feed.synd.SyndFeed;
1124.  import com.sun.syndication.feed.synd.SyndFeedImpl;
1125.  import com.sun.syndication.io.FeedException;
1126.  import com.sun.syndication.io.SyndFeedInput;
1127.  import com.sun.syndication.io.SyndFeedOutput;
1128.  import com.sun.syndication.io.XmlReader;
1129.  //Lucene packages
1130.  import org.apache.lucene.analysis.PorterStemFilter;
1131.  import org.apache.lucene.analysis.StopAnalyzer;
1132.  import org.apache.lucene.analysis.TokenStream;
1133.  import org.apache.lucene.analysis.tokenattributes.TermAttribute;
1134.  import org.apache.lucene.util.Version;
1135.  public class ReadRSS {
1136.  static HashMap CFHash ;
1137.  Vector <RSSiTem> rssITEM;
1138.  int NoOfRss=0;
1139.  ReadRSS(int noOfRss, int sortMethod){
1140.  NoOfRss=noOfRss;
1141.  rssITEM= new Vector <RSSiTem> ();
1142.  StopAnalyzer stop = new StopAnalyzer(Version.LUCENE_CURRENT);
1143.  CreateHashTable();
1144.  String urlRSS[]=getRSSurl();
1145.  //http://www.google.com/alerts/feeds/01286230342152329225/17070834082521743714
1146.  for(int R=0;R<urlRSS.length;R++){
1147.  System.out.println(urlRSS[R]);
1148.  URL url = null;
1149.  try {
1150.  url = new URL(urlRSS[R]);
1151.  } catch (MalformedURLException e1) {
1152.  // TODO Auto-generated catch block
1153.  e1.printStackTrace();
1154.  }
1155.  XmlReader reader = null;
1156.  String original = "";

1157.  try {
1158.  reader = new XmlReader(url);
1159.  SyndFeed feed = new SyndFeedInput().build(reader);
1160.  FileWriter fstream = new FileWriter("out.doc");
```

```
1161.  BufferedWriter out = new BufferedWriter(fstream);
1162.  String tmp_T = "";
1163.  String tmp_C = "";
1164.  for (Iterator i = feed.getEntries().iterator(); i.hasNext();) {
1165.  SyndEntry entry = (SyndEntry) i.next();
1166.  tmp_T = entry.getTitle();
1167.  tmp_T=RemoveHTMLtag( tmp_T);
1168.  if(feed.getFeedType().contains("atom")){ //--- ATOM ----
1169.  Iterator contentIter = entry.getContents().iterator();
1170.  while (contentIter.hasNext())
1171.  {
1172.  // Target the description node
1173.  SyndContent content = (SyndContent) contentIter.next();
1174.  content.setType("text/plain");
1175.  // Create and set a footer-appended description
1176.  original += content.getValue();
1177.  }//while
1178.  }//end if - atom
1179.  else // --- RSS ----
1180.  {
1181.  original=entry.getDescription().getValue();
1182.  }
1183.  tmp_C=RemoveHTMLtag(original);
1184.  // $$tmp_C = original;
1185.  // ====stem a single Item of RSS
1186.  String stem = "";
1187.  //Extract labels from graph
1188.  StringReader reader1 = new StringReader(tmp_T + tmp_C);
1189.  //tokenize
1190.  TokenStream stream = stop.tokenStream("test", reader1);
1191.  // stem all the labels of the graph
1192.  stream =  new PorterStemFilter(stream);
1193.  // extract text from stemmed stream
1194.  TermAttribute termAtt = stream.getAttribute(TermAttribute.class);
1195.  try {
1196.  while (stream.incrementToken()){
1197.  stem = stem + termAtt.term()+ " ";
1198.  }
1199.  } catch (IOException e) {
1200.  // TODO Auto-generated catch block
```

```
1201.  e.printStackTrace();
1202.  }// end while
1203.  // ===========
1204.  //$$rssITEM.add(new RSSiTem (tmp_T,original, ( tmp_T + tmp_C ), stem,entry.getLink()) );
1205.  rssITEM.add(new RSSiTem (entry.getTitle(),original, ( tmp_T + tmp_C ), stem,entry.getLink())
);
1206.  out.append(tmp_T + tmp_C);
1207.  original="";
1208.  tmp_T = "";
1209.  tmp_C= "";
1210.  }//for i
1211.  out.close();
1212.  } catch (IOException e) {
1213.  // TODO Auto-generated catch block
1214.  e.printStackTrace();
1215.  } catch (IllegalArgumentException e) {
1216.  // TODO Auto-generated catch block
1217.  e.printStackTrace();
1218.  } catch (FeedException e) {
1219.  // TODO Auto-generated catch block
1220.  e.printStackTrace();
1221.  } finally {
1222.  if (reader != null)
1223.  try {
1224.  reader.close();
1225.  } catch (IOException e) {
1226.  // TODO Auto-generated catch block
1227.  e.printStackTrace();
1228.  }
1229.  }//end try finally
1230.  }//end for
1231.  System.out.println("Title:  " + rssITEM.elementAt(0).Original_Title);
1232.  System.out.println("Content:  " + rssITEM.elementAt(0).Original_Content);
1233.  System.out.println("Conact:  " + rssITEM.elementAt(0).Concat);
1234.  System.out.println("Stem:   " + rssITEM.elementAt(0).Stem);
1235.  System.out.println("Vec Size"+rssITEM.size());
1236.  //concept Map
1237.  ClaculateGraphWeights CM= new ClaculateGraphWeights();
1238.  // fuzzy association between each node and a single RSS item
1239.  double [] FuzzyAssociation= new double[ CM.myNodes.size()];
```

```
1240.  double  CRDFuzzy=1;
1241.  double HARDFuzzy=1;
1242.  double PCFuzzy= 1;
1243.  double CRDdegreeOfSimilarity=0;
1244.  double HARDdegreeOfSimilarity= 0;
1245.  double PCdegreeOfSimilarity=0;
1246.  for(int k=0;k<rssITEM.size();k++){
1247.  //single words in RSS item
1248.  String RSSWords[]=rssITEM.elementAt(k).Stem.split(" ");
1249.  for(int i=0;i<CM.myNodes.size();i++)
1250.  {
1251.  double tmp = 1;
1252.  String []NodeContent=CM.myNodes.elementAt(i).content.split(" ");
1253.  for(int j=0;j<RSSWords.length;j++)
1254.  {
1255.  double CFavg=0;
1256.  for (int h = 0 ; h< NodeContent.length ; h++)
1257.  {
1258.  if(CFHash.get(NodeContent[h]+ " " +RSSWords[j] )==null && CFHash.get(NodeContent[h]+ "
" +RSSWords[j] )==null)
1259.  {
1260.  CFavg+=0;
1261.  }// IF
1262.  else
1263.  {
1264.  if(CFHash.get(NodeContent[h]+ " " +RSSWords[j] )==null)
1265.  CFavg+= Double.parseDouble(""+CFHash.get(NodeContent[j] + " " +RSSWords[h] ));
1266.  else
1267.  CFavg+= Double.parseDouble(""+CFHash.get(NodeContent[h] + " " +RSSWords[j] ));
1268.  }//else
1269.  }//h
1270.  CFavg=CFavg/(double)NodeContent.length;
1271.  tmp  =tmp*(1-CFavg);
1272.  }//j
1273.  FuzzyAssociation[i]=1.0-tmp;
1274.  CRDFuzzy= FuzzyAssociation[i]*CM.myNodes.elementAt(i).CRDvalue;
1275.  HARDFuzzy= FuzzyAssociation[i]*CM.myNodes.elementAt(i).HARDvalue;
1276.  PCFuzzy=FuzzyAssociation[i]*CM.myNodes.elementAt(i).pathCount;
1277.  CRDdegreeOfSimilarity=CRDdegreeOfSimilarity+CRDFuzzy;
1278.  HARDdegreeOfSimilarity= HARDdegreeOfSimilarity+HARDFuzzy;
```

```
1279.  PCdegreeOfSimilarity=PCdegreeOfSimilarity+PCFuzzy;
1280.  }//i
1281.  // calculate the similarity measure
1282.  CRDdegreeOfSimilarity=CRDdegreeOfSimilarity/CM.myNodes.size();
1283.  HARDdegreeOfSimilarity= HARDdegreeOfSimilarity/CM.myNodes.size();
1284.  PCdegreeOfSimilarity=PCdegreeOfSimilarity/CM.myNodes.size();
1285.  rssITEM.elementAt(k).setRank(CRDdegreeOfSimilarity,
HARDdegreeOfSimilarity,PCdegreeOfSimilarity);
1286.  // reset
1287.  CRDdegreeOfSimilarity=0;
1288.  HARDdegreeOfSimilarity= 0;
1289.  PCdegreeOfSimilarity=0;
1290.  //System.out.println(rssITEM.elementAt(k).CRDrank);
1291.  //System.out.println(rssITEM.elementAt(k).HARDrank);
1292.  //System.out.println(" PC" + rssITEM.elementAt(k).PCrank);
1293.  }//k
1294.  if(sortMethod==0)
1295.  {
1296.  SortCRDrank(rssITEM);
1297.  GenerateRankedFeed(rssITEM,"CRDrank");
1298.  }
1299.  else if(sortMethod==1)
1300.  {
1301.  SortHARDrank(rssITEM);
1302.  GenerateRankedFeed(rssITEM,"HARDrank");
1303.  }
1304.  else if(sortMethod==2)
1305.  {
1306.  SortPCrank(rssITEM);
1307.  GenerateRankedFeed(rssITEM,"PCrank");
1308.  }
1309.  }//end ReadRSS constructor
1310.  String[] getRSSurl()
1311.  {
1312.  String outRSS="";
1313.  InputStreamReader fstream;
1314.  try {
1315.  fstream = new InputStreamReader  (new FileInputStream("./Resources/RSSFeeds.txt"));
1316.  BufferedReader  in = new BufferedReader (fstream);
1317.  try {
```

```
1318.  int c;
1319.  while ( (c=in.read()) != -1)
1320.  {
1321.  outRSS+=((char)c);
1322.  }// end while
1323.  in.close();
1324.  } catch (IOException e) {
1325.  // TODO Auto-generated catch block
1326.  e.printStackTrace();
1327.  }
1328.  } catch (FileNotFoundException e1) {
1329.  // TODO Auto-generated catch block
1330.  e1.printStackTrace();
1331.  }
1332.  return outRSS.split("#");
1333.  }//getRSSurl
1334.  //=======================
1335.  void GenerateRankedFeed(Vector<RSSiTem> feeds,String title)
1336.  {
1337.  DateFormat DATE_PARSER = new SimpleDateFormat("yyyy-MM-dd");

1338.  try {
1339.  String feedType = "rss_2.0";
1340.  //String feedType = "atom_1.0";
1341.  String fileName = "./Resources/feed_" + title + ".xml";
1342.  SyndFeed feed = new SyndFeedImpl();
1343.  feed.setFeedType(feedType);
1344.  feed.setTitle(title);
1345.  //http://www.rgagnon.com/howto.html
1346.  feed.setLink("http://www.aus.edu");
1347.  feed.setDescription("Top Ten RSS Feeds Related to Mobile Applications");
1348.  List entries = new ArrayList();
1349.  SyndEntry entry;
1350.  SyndContent description;
1351.  for(int i=0;i<NoOfRss;i++){
1352.  entry = new SyndEntryImpl();
1353.  entry.setTitle(feeds.elementAt(i).Original_Title);
1354.  entry.setLink(feeds.elementAt(i).Link);
1355.  entry.setPublishedDate(DATE_PARSER.parse("2004-06-08"));
1356.  description = new SyndContentImpl();
```

```
1357.  description.setType("text/plain");
1358.  description.setValue(feeds.elementAt(i).Original_Content );
1359.  entry.setDescription(description);
1360.  entries.add(entry);
1361.  feed.setEntries(entries);
1362.  }//end for
1363.  OutputStreamWriter writer = new OutputStreamWriter(new
FileOutputStream(fileName),"UTF-8");
1364.  //Writer writer = new FileWriter(fileName);
1365.  SyndFeedOutput output = new SyndFeedOutput();
1366.  output.output(feed,writer);
1367.  writer.close();
1368.  }
1369.  catch (Exception ex) {
1370.  ex.printStackTrace();
1371.  }
1372.  }// end generate RSS
1373.  //================
1374.  void SortCRDrank(Vector<RSSiTem> feeds)
1375.  {
1376.  RSSiTem temp;
1377.  for(int j=0;j<feeds.size()-1;j++){
1378.  for(int i=0;i<feeds.size()-1;i++)
1379.  {
1380.  if(feeds.elementAt(i).CRDrank<feeds.elementAt(i+1).CRDrank)
1381.  {
1382.  temp=feeds.elementAt(i);
1383.  feeds.set(i,feeds.elementAt(i+1));
1384.  feeds.set(i+1,temp);
1385.  }
1386.  }//for i
1387.  }//for j
1388.  }// end sort CRD
1389.  //============
1390.  static void SortHARDrank(Vector<RSSiTem> feeds)
1391.  {
1392.  RSSiTem temp;
1393.  for(int j=0;j<feeds.size()-1;j++){
1394.  for(int i=0;i<feeds.size()-1;i++)
1395.  {
```

```
1396.  if(feeds.elementAt(i).HARDrank<feeds.elementAt(i+1).HARDrank)
1397.  {
1398.  temp=feeds.elementAt(i);
1399.  feeds.set(i,feeds.elementAt(i+1));
1400.  feeds.set(i+1,temp);
1401.  }
1402.  }//for i
1403.  }//for j
1404.  }// end SortHARDrank(
1405.  //======================
1406.  void SortPCrank(Vector<RSSiTem> feeds)
1407.  {
1408.  RSSiTem temp;
1409.  for(int j=0;j<feeds.size()-1;j++){
1410.  for(int i=0;i<feeds.size()-1;i++)
1411.  {
1412.  if(feeds.elementAt(i).PCrank<feeds.elementAt(i+1).PCrank)
1413.  {
1414.  temp=feeds.elementAt(i);
1415.  feeds.set(i,feeds.elementAt(i+1));
1416.  feeds.set(i+1,temp);
1417.  }
1418.  }//for i
1419.  }//for j
1420.  }
1421.  String RemoveHTMLtag(String s)
1422.  {
1423.  // Remove <a ,,,,,,,,,> www.htmll.com </a>
1424.  s=removelink(s);
1425.  //Remove nonsens
1426.  s = s.replace("&#39;","'");
1427.  s = s.replace("&lt;br /&gt;&lt;br /&gt;","");
1428.  s=s.replace(".", "");
1429.  s=s.replace( "&quot;","");
1430.  s=s.replace("&amp","");
1431.  s=s.replace("href","");
1432.  s=s.replace("<a","");
1433.  //s=s.replace("<a","");
1434.  s=s.replace("style=color","");
```

```
1435.  //remove any thing between <>

1436.  while(s.contains("<")||s.contains(">")){
1437.  int indo= s.indexOf("<");
1438.  int indc= s.indexOf(">");
1439.  s=removetag(s,indo,indc);
1440.  }//while
1441.  return s;
1442.  }//RemoveHTMLtag
1443.  //Remove <a ,,,,,,,,,> www.htmll.com </a>
1444.  public  String removelink(String s)
1445.  {
1446.  while(s.contains("<a")||s.contains("</a>")){
1447.  int linkfirst=s.indexOf("<a");
1448.  int linklast=s.indexOf("</a>");
1449.  s=removetag(s,linkfirst,linklast+3);
1450.  }//while
1451.  return s;
1452.  }//removelink
1453.  public static String removetag(String s,int openPrc,int closePrc)
1454.  {
1455.  String x= s.substring(0,openPrc)+s.substring(closePrc+1,s.length());
1456.  return x;
1457.  }//remove
1458.  public  void  CreateHashTable()
1459.  {
1460.  //------------- [ Load HashTable --------------------
1461.  try {
1462.  FileInputStream fileIn = new FileInputStream("./CrFcHashMap.ser");
1463.  ObjectInputStream in =new ObjectInputStream(fileIn);
1464.  CFHash = (HashMap)in.readObject();
1465.  in.close();
1466.  fileIn.close();
1467.  }
1468.  catch (ClassNotFoundException e) {
1469.  e.printStackTrace();
1470.  }
1471.  catch(FileNotFoundException e) {
1472.  e.printStackTrace();
1473.  }
```

```
1474.  catch (IOException e) {
1475.  e.printStackTrace();
1476.  }// end try catch
1477.  } // end func CreateHashTable
1478.  }///end class
1479. package org.apache.lucene.analysis;
1480. public class RSSiTem {
1481. String Original_Title;
1482. String Original_Content;
1483. String Concat;
1484. String Stem;
1485. String Link;
1486. double CRDrank;
1487. double HARDrank;
1488. double PCrank;
1489. public RSSiTem(String orT,String orC, String con,String stem,String
l)
1490. {
1491. Original_Title = orT;
1492. Original_Content = orC;
1493. Concat = con;
1494. Stem = stem;
1495. Link=l;
1496. CRDrank=0;
1497. HARDrank=0;
1498. PCrank=0;
1499. }
1500. //constructor
1501. public void setRank (double c,double h,double p )
1502. {
1503. CRDrank=c;
1504. HARDrank=h;
1505. PCrank=p;
1506. }
1507. //end setRank
1508. }
1509. // end class
1510.  // ========== [ Setup RSS Form ] ============
1511.  package org.apache.lucene.analysis;
1512.  import java.awt.event.ActionEvent;
1513.  import java.awt.event.ActionListener;
```

```
1514. import java.io.BufferedReader;
1515. import java.io.BufferedWriter;
1516. import java.io.File;
1517. import java.io.FileInputStream;
1518. import java.io.FileNotFoundException;
1519. import java.io.FileOutputStream;
1520. import java.io.IOException;
1521. import java.io.InputStreamReader;
1522. import java.io.OutputStreamWriter;
1523. import javax.swing.AbstractButton;
1524. import javax.swing.JButton;
1525. import javax.swing.JComboBox;
1526. import javax.swing.JFileChooser;
1527. import javax.swing.JFrame;
1528. import javax.swing.JLabel;
1529. import javax.swing.JPanel;
1530. import javax.swing.JTextField;
1531. public class SetupRssForm implements ActionListener {
1532. JFrame frame;
1533. JTextField RSSnametxtf;
1534. JLabel numRSSlab;
1535. int NoRSSFeeds=0;
1536. JButton Add;
1537. JButton Save;
1538. JButton Reset;
1539. BufferedWriter out;
1540. SetupRssForm(){
1541. frame = new JFrame("Setup RSS Feeds");
1542. JPanel mainPanel = new JPanel();
1543. JLabel label = new JLabel("RSS Feeds:");
1544. Add= new JButton("Add RSS", null);
1545. Save= new JButton("Save RSS", null);
1546. Reset= new JButton("Reset", null);
1547. RSSnametxtf = new JTextField(40);
1548. Reset.setEnabled(false);
1549. Save.setEnabled(false);
1550. if(Check()==true)
1551. {
1552. Save.setEnabled(false);
1553. RSSnametxtf.setEnabled(false);
```

```
1554.  Add.setEnabled(false);
1555.  Reset.setEnabled(true);
1556.  }//if (there is RSS feeds)
1557.  else{
1558.  try {
1559.  OutputStreamWriter fstream;
1560.  fstream= new OutputStreamWriter(new FileOutputStream("./Resources/RSSFeeds.txt"));
1561.  out = new BufferedWriter(fstream);
1562.  } catch (FileNotFoundException e) {
1563.  // TODO Auto-generated catch block
1564.  e.printStackTrace();
1565.  }
1566.  }//else (there is No RSS feeds)
1567.  Save.addActionListener(this);
1568.  Add.addActionListener(this);
1569.  Reset.addActionListener(this);
1570.  numRSSlab= new JLabel("You have "+NoRSSFeeds+" RSS feeds stored");
1571.  mainPanel.add(label);
1572.  mainPanel.add(RSSnametxtf);
1573.  mainPanel.add(Add);
1574.  mainPanel.add(numRSSlab);
1575.  mainPanel.add(Save);
1576.  mainPanel.add(Reset);
1577.  frame.add( mainPanel);
1578.  frame.setSize(700,150);
1579.  frame.setVisible(true);
1580.  }//
1581.  public void actionPerformed(ActionEvent actionEvent)
1582.  {
1583.  AbstractButton aButton = (AbstractButton) actionEvent.getSource();
1584.  String label = aButton.getText();
1585.  if(label.equals("Add RSS"))
1586.  {
1587.  try {
1588.  out.append(RSSnametxtf.getText()+"#");
1589.  NoRSSFeeds++;
1590.  numRSSlab.setText("You have "+NoRSSFeeds+" RSS feeds stored");
1591.  } catch (IOException e) {
1592.  // TODO Auto-generated catch block
1593.  e.printStackTrace();
```

```
1594.  }//catch
1595.  Save.setEnabled(true);
1596.  RSSnametxtf.setText("");
1597.  }
1598.  else if(label.equals("Save RSS"))
1599.  {

1600.  try {
1601.  OutputStreamWriter fstream2;
1602.  fstream2= new OutputStreamWriter(new FileOutputStream("./Resources/noRSS.txt"));
1603.  BufferedWriter out2 = new BufferedWriter(fstream2);
1604.  out2.write(""+NoRSSFeeds);
1605.  out2.close();
1606.  out.close();
1607.  } catch (FileNotFoundException e) {
1608.  // TODO Auto-generated catch block
1609.  e.printStackTrace();
1610.  } catch (IOException e) {
1611.  // TODO Auto-generated catch block
1612.  e.printStackTrace();
1613.  }
1614.  Add.setEnabled(false);
1615.  }//else if
1616.  else if(label.equals("Reset"))
1617.  {
1618.  NoRSSFeeds=0;
1619.  numRSSlab.setText("You have "+NoRSSFeeds+" RSS feeds stored");
1620.  File file=new File("./Resources/RSSFeeds.txt");
1621.  File file2=new File("./Resources/noRSS.txt");
1622.  file.delete();
1623.  file2.delete();
1624.  try {
1625.  OutputStreamWriter fstream;
1626.  fstream= new OutputStreamWriter(new FileOutputStream("./Resources/RSSFeeds.txt"));
1627.  out = new BufferedWriter(fstream);
1628.  } catch (FileNotFoundException e) {
1629.  // TODO Auto-generated catch block
1630.  e.printStackTrace();
1631.  }//catch
1632.  Save.setEnabled(true);
```

```
1633.  RSSnametxtf.setEnabled(true);
1634.  Add.setEnabled(true);
1635.  Reset.setEnabled(false);
1636.  }//elseif
1637.  }//actionPerformed
1638.  boolean Check()
1639.  {
1640.  String num="";
1641.  File file=new File("./Resources/noRSS.txt");
1642.  boolean exists = file.exists();
1643.  if (!exists) {
1644.  NoRSSFeeds=0;
1645.  // It returns false if File or directory does not exist
1646.  return false;
1647.  }else{
1648.  InputStreamReader fstream;
1649.  try {
1650.  fstream = new InputStreamReader  (new FileInputStream("./Resources/noRSS.txt"));
1651.  BufferedReader  in = new BufferedReader (fstream);
1652.  try {
1653.  num=in.readLine();
1654.  } catch (IOException e) {
1655.  // TODO Auto-generated catch block
1656.  e.printStackTrace();
1657.  }
1658.  } catch (FileNotFoundException e) {
1659.  // TODO Auto-generated catch block
1660.  e.printStackTrace();
1661.  }
1662.  NoRSSFeeds= Integer.parseInt(num)  ;
1663.  // It returns true if File or directory exists
1664.  return true;
1665.  }
1666.  }
1667.  }//
1668.  package org.apache.lucene.analysis;
1669.  import java.awt.Color;
1670.  import java.awt.GridLayout;
1671.  import java.awt.event.ActionEvent;
1672.  import java.awt.event.ActionListener;
```

```
1673.  import java.awt.event.ItemEvent;
1674.  import java.awt.event.ItemListener;
1675.  import java.io.BufferedWriter;
1676.  import java.io.File;
1677.  import java.io.FileNotFoundException;
1678.  import java.io.FileOutputStream;
1679.  import java.io.IOException;
1680.  import java.io.OutputStreamWriter;
1681.  import javax.swing.AbstractButton;
1682.  import javax.swing.ButtonGroup;
1683.  import javax.swing.JButton;
1684.  import javax.swing.JComboBox;
1685.  import javax.swing.JFileChooser;
1686.  import javax.swing.JFrame;
1687.  import javax.swing.JLabel;
1688.  import javax.swing.JPanel;
1689.  import javax.swing.JRadioButton;
1690.  import javax.swing.JScrollPane;
1691.  import javax.swing.JTextArea;
1692.  import javax.swing.JTextField;
1693.  public class viewRecentUpdateForm implements ActionListener {
1694.  JPanel mainPanel;
1695.  JComboBox comboNoRSS;
1696.  JComboBox comboSort;
1697.  ReadRSS rssRead;
1698.  brows browser;
1699.  JFrame frame;
1700.  viewRecentUpdateForm(){
1701.  mainPanel= new JPanel();
1702.  browser= new brows();
1703.  frame= new JFrame("View Recent Update Setup");
1704.  JPanel Panel = new JPanel();
1705.  String noOfRSS[] = {"5","10","15","20"};
1706.  String sortType[] = {"CRD Rank","HARD Rank","PC Rank"};
1707.  comboNoRSS= new JComboBox(noOfRSS);
1708.  comboSort= new JComboBox(sortType);
1709.  JButton ok= new JButton("OK", null);
1710.  //ReadRSS rssRead=new ReadRSS();
1711.  //  RSSPanel();
1712.  Panel.add(comboNoRSS);
```

```
1713.  Panel.add(comboSort);
1714.  Panel.add(ok);
1715.  ok.addActionListener(this);
1716.  frame.add( Panel);
1717.  frame.setSize(400,150);
1718.  frame.setVisible(true);
1719.  }//
1720.  void RSSDisplay()
1721.  {
1722.  JFrame frame2= new JFrame("RSS");
1723.  frame2.add( mainPanel);
1724.  //JScrollPane scrollPane = new JScrollPane(mainPanel);
1725.  //frame2.add(scrollPane );

1726.  frame2.setSize(400,1000);

1727.  frame.setVisible(false);
1728.  frame2.setVisible(true);
1729.  }//RSSDisplay
1730.  void RSSPanel(String txt,int i)
1731.  {
1732.  JPanel panel = new JPanel( new GridLayout(0,2));
1733.  JButton view= new JButton("View Page "+i, null);
1734.  view.addActionListener(this);
1735.  JTextArea RSStxtArea = new JTextArea(5, 20);
1736.  JScrollPane scrollPane = new JScrollPane(RSStxtArea);
1737.  RSStxtArea.setEditable(false);
1738.  RSStxtArea.setText(txt);
1739.  panel.add(scrollPane);
1740.  panel.add(view);
1741.  mainPanel.add(panel);
1742.  }//
1743.  public void actionPerformed(ActionEvent actionEvent)
1744.  {
1745.  AbstractButton aButton = (AbstractButton) actionEvent.getSource();
1746.  String label = aButton.getText();
1747.  if(label.equals("OK"))
1748.  {
1749.  int rssNoarr[]={5,10,15,20};
```

```
1750.  int rssN=comboNoRSS.getSelectedIndex();  // number of items to display in the result 5, 10 , 15 , 20
1751.  int sort=comboSort.getSelectedIndex(); // sorting method CRD, HARD , PC
1752.  rssRead=new ReadRSS(rssNoarr[rssN],sort); // instantiate new object to read and rank RSS feed
1753.  for(int i=0;i<rssNoarr[rssN];i++)
1754.  {
1755.  RSSPanel(rssRead.rssITEM.elementAt(i).Original_Content,(i+1));
1756.  }//for
1757.  RSSDisplay();
1758.  }
1759.  else if(label.contains("View Page"))
1760.  {
1761.  //index 10 start the number
1762.  int c= Integer.parseInt(label.substring(10));
1763.  c--;
1764.  browser.openURL(rssRead.rssITEM.elementAt(c).Link);

1765.  }
1766.  }//actionPerformed
1767.  };
```

APPENDIX B

Kappa Count Tables

Appendix B- Kappa Count Tables

Count Tables For Experts' Multiple-Raters Test Using Fleis's Kappa Coefficient

Fleiss Kappa for multiple raters – All Experts
Raters=5 Categories=20 Subjects=20
Table : subjects(rows) by categories(cols)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1
3	2	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
5	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0
6	1	0	1	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
7	0	1	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1
11	0	0	0	0	0	0	0	0	0	1	1	0	0	3	0	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0
13	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0
14	0	0	1	0	0	0	1	0	0	0	0	0	1	0	2	0	0	0	0	0
15	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0
16	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	2	0	1	0	0	0	1	0	0	0
18	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	2
19	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1
20	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	2	0

Overall Kappa = 0.0158 SE=0.0162
95% CI=-0.0160 to 0.0476

Appendix B- Kappa Count Tables

Count Tables For Experts' Two-Raters Test Using Cohen's Kappa Coefficient

Cohen Kappa for two rators – E3 & E4
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
13	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
18	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
20	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.1429 se=0.1422
95% CI=-0.1359 to 0.4216

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E3 & E2
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
12	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
16	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.1429 se=0.1422
95% CI=-0.1359 to 0.4216

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E3 & E5
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
6	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
13	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.1128 se=0.1422
95% CI=-0.1660 to 0.3915

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E3 & E1
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.1429 se=0.1422
95% CI=-0.1359 to 0.4216

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E1 & E5
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
13	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
19	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=-0.0526 se=0.1422
95% CI=-0.3314 to 0.2261

Appendix B- Kappa Count Tables

Cohen Kappa for two rators –E1 & E2
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
16	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
18	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.3383 se=0.1422
95% CI=0.0596 to 0.6171

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E1 & E4
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
8	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
14	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
18	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.0376 se=0.1422
95% CI=-0.2412 to 0.3163

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E4 & E5
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
11	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
17	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
20	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.0827 se=0.1422
95% CI=-0.1960 to 0.3615

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E4 & E2
Minimum score = 1
Maximum score = 20

Count matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Tot
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
12	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
15	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
18	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=0.0226 se=0.1422
95% CI=-0.2562 to 0.3013

Appendix B- Kappa Count Tables

Cohen Kappa for two rators – E2 & E5
Minimum score = 1
Maximum score = 20

Count matrix

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**Tot**
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
9	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
17	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
18	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
19	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
Tot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20

Kappa=-0.0827 se=0.1422
95% CI=-0.3615 to 0.1960

Count Table For Feis's Kappa - All Topological Models (CRD, HARD, PC)

Fleiss Kappa for multiple raters – all Models CDR , HARD & PC , with all parameters' levels (3,7) & (1,9)
Raters=17 Categories=20 Subjects=20

Table subjects(rows) by categories(cols)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
3	0	0	0	0	0	0	0	0	0	0	0	1	11	3	2	0	0	0	0	0
4	0	0	1	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	1	5	1	8	1	0	0	0	0	0	0	0
6	0	0	0	0	10	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	6	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	6	0	0	2	5	1	3	0	0	0	0	0	0	0
9	0	1	1	0	5	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	16	0
12	0	0	0	0	0	0	1	13	2	1	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	9	3	5	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	9	1	0	4	3	0	0	0	0	0	0	0	0	0
15	0	10	4	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	1	8	8	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0
18	0	0	0	0	0	0	1	0	0	1	1	0	1	6	7	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	1	0
20	0	0	0	0	0	0	0	2	5	1	2	7	0	0	0	0	0	0	0	0

Overall Kappa = 0.5553 SE=0.0044
95% CI=0.5467 to 0.5640

Count Table For Fliess'' Kappa - Experts Vs. PC Model

Fleiss Kappa for multiple raters – Experts vs PC
Raters=6 Categories=20 Subjects=20
Table subjects(rows) by categories(cols)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	3	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	2
3	2	2	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	1	2	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
5	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	0
6	1	0	1	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
7	0	1	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	1	0	1	2	0	1	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	1	0	0	2	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	1	1
11	0	0	0	0	0	0	0	0	0	1	1	0	0	3	0	0	0	1	0	0
12	0	0	0	1	0	0	0	0	0	2	1	0	0	0	0	0	1	1	0	0
13	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	1	0	1	0	0
14	0	0	1	0	0	0	1	1	0	0	0	0	1	0	2	0	0	0	0	0
15	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0
16	0	0	0	1	0	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	2	0	1	0	0	1	1	0	0	0
18	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	2
19	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	2	1
20	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	2	0

Overall Kappa = 0.0175 SE=0.0132
95% CI=-0.0084 to 0.0435

Count table for Fliess'' Kappa - Experts Vs. HARD (3,7) model

Fleiss Kappa for multiple raters – Experts vs HARD (3,7)
Raters=9 Categories=20 Subjects=20

Table subjects(rows) by categories(cols)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	5
3	2	2	0	0	0	1	0	0	0	0	0	0	3	1	0	0	0	0	0	0
4	0	0	1	5	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
5	0	1	0	0	0	0	0	0	2	3	0	0	0	0	0	0	1	1	1	0
6	1	0	1	0	5	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
7	0	1	4	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	1	0	1	2	0	1	3	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	4	0	0	1	0	0	2	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	4	0	1	1
11	0	0	0	0	0	0	0	0	0	1	1	0	0	3	0	0	0	0	4	0
12	0	0	0	1	0	0	0	4	0	1	1	0	0	0	0	0	1	1	0	0
13	0	0	0	0	1	0	0	0	3	1	0	1	0	1	0	1	0	1	0	0
14	0	0	1	0	0	0	5	0	0	0	0	0	1	0	2	0	0	0	0	0
15	0	4	0	0	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0
16	0	0	0	1	0	0	0	0	1	1	1	0	0	0	4	1	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	2	0	1	0	0	4	1	0	0	0
18	0	0	0	0	1	0	0	0	0	1	0	1	1	3	0	0	0	0	0	2
19	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	4	1	1
20	0	0	1	0	0	1	1	0	0	0	1	3	0	0	0	0	0	0	2	0

Overall Kappa = 0.1520 SE=0.0085
95% CI=0.1353 to 0.1688

Appendix B- Kappa Count Tables

Count table for Fliess'' Kappa - Experts Vs. HARD (1,9) model

Fleiss Kappa for multiple raters – Experts vs HARD (1,9)
Raters=9 Categories=20 Subjects=20

Table subjects(rows) by categories(cols)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	5
3	2	2	0	0	0	1	0	0	0	0	0	0	3	1	0	0	0	0	0	0
4	0	0	2	4	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
5	0	1	0	0	0	0	0	1	1	2	1	0	0	0	0	0	1	1	1	0
6	1	0	1	0	5	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
7	0	2	3	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	1	0	1	2	0	2	2	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	4	0	0	1	0	0	2	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	4	0	1	1
11	0	0	0	0	0	0	0	0	0	1	1	0	0	3	0	0	0	0	4	0
12	0	0	0	1	0	0	1	3	0	1	1	0	0	0	0	0	1	1	0	0
13	0	0	0	0	1	0	0	0	4	0	0	1	0	1	0	1	0	1	0	0
14	0	0	1	0	0	0	4	0	0	0	1	0	1	0	2	0	0	0	0	0
15	0	3	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0
16	0	0	0	1	0	0	0	0	1	1	1	0	0	1	3	1	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	2	0	1	0	0	4	1	0	0	0
18	0	0	0	0	1	0	0	0	0	2	0	1	0	2	1	0	0	0	0	2
19	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	4	1	1
20	0	0	1	0	0	1	1	0	0	0	0	4	0	0	0	0	0	0	2	0

Overall Kappa = 0.1272 SE=0.0085
95% CI=0.1104 to 0.144

Appendix B- Kappa Count Tables

Count table for Fliess'' Kappa - Experts Vs. CRD (1,9) model

Fleiss Kappa for multiple raters – Experts vs CRD (1,9)
Raters=9 Categories=20 Subjects=20
Table subjects(rows) by categories(cols)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	5
3	2	2	0	0	0	1	0	0	0	0	0	1	2	0	1	0	0	0	0	0
4	0	0	1	5	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
5	0	1	0	0	0	0	0	0	1	0	0	3	1	0	0	0	1	1	1	0
6	1	0	1	0	2	3	0	0	0	0	0	0	0	0	2	0	0	0	0	0
7	0	3	2	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	1	0	4	2	0	1	0	0	1	0	0	0	0	0	0	0
9	0	1	0	0	2	1	0	0	1	0	0	2	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	4	0	1	1
11	0	0	0	0	0	0	0	0	0	1	1	0	0	3	0	0	0	0	4	0
12	0	0	0	1	0	0	0	3	1	1	1	0	0	0	0	0	1	1	0	0
13	0	0	0	0	1	0	0	0	1	1	2	1	0	1	0	1	0	1	0	0
14	0	0	1	0	0	0	2	0	0	2	1	0	1	0	2	0	0	0	0	0
15	0	1	2	0	1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0
16	0	0	0	1	0	0	0	0	1	1	1	0	0	4	0	1	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	2	0	1	0	0	4	1	0	0	0
18	0	0	0	0	1	0	0	0	0	1	1	1	0	0	3	0	0	0	0	2
19	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	4	1	1
20	0	0	1	0	0	1	1	1	2	1	0	0	0	0	0	0	0	0	2	0

Overall Kappa = 0.1038 SE=0.0085
95% CI=0.0870 to 0.1206

Appendix B- Kappa Count Tables

Count table for Fliess' Kappa - Experts Vs. CRD (3,7) model

Fleiss Kappa for multiple raters – Experts vs CRD (3,7)
Raters=9 Categories=20 Subjects=20
Table subjects(rows) by categories(cols)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	5
3	2	2	0	0	0	1	0	0	0	0	0	0	3	0	1	0	0	0	0	0
4	0	0	1	5	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
5	0	1	0	0	0	0	0	0	1	0	0	4	0	0	0	0	1	1	1	0
6	1	0	1	0	1	4	0	0	0	0	0	0	0	0	2	0	0	0	0	0
7	0	4	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	1	0	4	2	0	2	0	0	0	0	0	0	0	0	0	0
9	0	0	1	0	3	0	0	0	1	0	0	2	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	4	0	1	1
11	0	0	0	0	0	0	0	0	0	1	1	0	0	3	0	0	0	0	4	0
12	0	0	0	1	0	0	0	3	1	1	1	0	0	0	0	0	1	1	0	0
13	0	0	0	0	1	0	0	0	1	1	2	1	0	1	0	1	0	1	0	0
14	0	0	1	0	0	0	2	0	0	2	1	0	1	0	2	0	0	0	0	0
15	0	1	2	0	1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0
16	0	0	0	1	0	0	0	0	1	1	1	0	1	3	0	1	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	2	0	1	0	0	4	1	0	0	0
18	0	0	0	0	1	0	0	0	0	1	0	1	0	1	3	0	0	0	0	2
19	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	4	1	1
20	0	0	1	0	0	1	1	1	2	0	1	0	0	0	0	0	0	0	2	0

Overall Kappa = 0.1170 SE=0.0085
95% CI=0.1002 to 0.1337

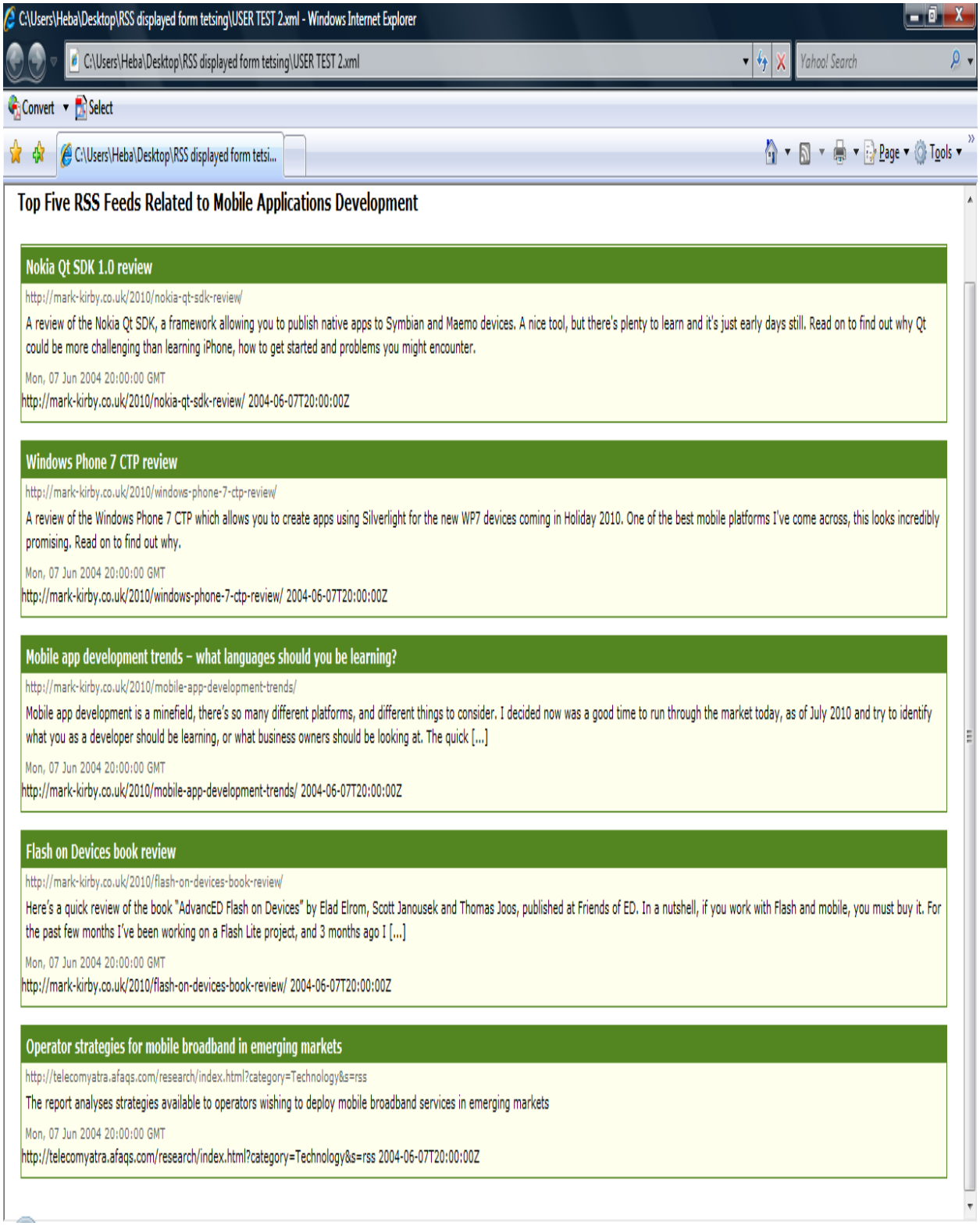
APPENDIX C

User Tests' Results

TEST # 1


C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 1.xml - Windows Internet Explorer
C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 1.xml
Yahoo! Search
C:\Users\Heba\Desktop\RSS displayed form tetsi...
Page
Tools
Top Five RSS Feeds Related to Mobile Applications Development
Nokia Qt SDK 1.0 review
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/
A review of the Nokia Qt SDK, a framework allowing you to publish native apps to Symbian and Maemo devices. A nice tool, but there's plenty to learn and it's just early days still. Read on to find out why Qt could be more challenging than learning iPhone, how to get started and problems you might encounter.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/ 2004-06-07T20:00:00Z
Windows Phone 7 CTP review
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/
A review of the Windows Phone 7 CTP which allows you to create apps using Silverlight for the new WP7 devices coming in Holiday 2010. One of the best mobile platforms I've come across, this looks incredibly promising. Read on to find out why.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/ 2004-06-07T20:00:00Z
Mobile app development trends – what languages should you be learning?
http://mark-kirby.co.uk/2010/mobile-app-development-trends/
Mobile app development is a minefield, there's so many different platforms, and different things to consider. I decided now was a good time to run through the market today, as of July 2010 and try to identify what you as a developer should be learning, or what business owners should be looking at. The quick [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/mobile-app-development-trends/ 2004-06-07T20:00:00Z
Flash on Devices book review
http://mark-kirby.co.uk/2010/flash-on-devices-book-review/
Here's a quick review of the book "AdvancED Flash on Devices" by Elad Elrom, Scott Janousek and Thomas Joos, published at Friends of ED. In a nutshell, if you work with Flash and mobile, you must buy it. For the past few months I've been working on a Flash Lite project, and 3 months ago I [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/flash-on-devices-book-review/ 2004-06-07T20:00:00Z
Optimising visual components for Flash Mobile
http://mark-kirby.co.uk/2009/optimising-visual-components-for-flash-mobile/
I've been working on some a visual component for a flash mobile app, and thought I'd use it to give you a quick case study on how important it is to optimise everything for mobile. Although this example came from an app for Flash Mobile and Flash Lite, the same problem is suffered by iPhone [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2009/optimising-visual-components-for-flash-mobile/ 2004-06-07T20:00:00Z

TEST # 2


C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 2.xml - Windows Internet Explorer
C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 2.xml
Yahoo! Search
Convert
Select
C:\Users\Heba\Desktop\RSS displayed form tetsi...
Page
Tools
Top Five RSS Feeds Related to Mobile Applications Development
Nokia Qt SDK 1.0 review
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/
A review of the Nokia Qt SDK, a framework allowing you to publish native apps to Symbian and Maemo devices. A nice tool, but there's plenty to learn and it's just early days still. Read on to find out why Qt could be more challenging than learning iPhone, how to get started and problems you might encounter.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/ 2004-06-07T20:00:00Z
Windows Phone 7 CTP review
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/
A review of the Windows Phone 7 CTP which allows you to create apps using Silverlight for the new WP7 devices coming in Holiday 2010. One of the best mobile platforms I've come across, this looks incredibly promising. Read on to find out why.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/ 2004-06-07T20:00:00Z
Mobile app development trends – what languages should you be learning?
http://mark-kirby.co.uk/2010/mobile-app-development-trends/
Mobile app development is a minefield, there's so many different platforms, and different things to consider. I decided now was a good time to run through the market today, as of July 2010 and try to identify what you as a developer should be learning, or what business owners should be looking at. The quick [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/mobile-app-development-trends/ 2004-06-07T20:00:00Z
Flash on Devices book review
http://mark-kirby.co.uk/2010/flash-on-devices-book-review/
Here's a quick review of the book "AdvancED Flash on Devices" by Elad Elrom, Scott Janousek and Thomas Joos, published at Friends of ED. In a nutshell, if you work with Flash and mobile, you must buy it. For the past few months I've been working on a Flash Lite project, and 3 months ago I [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/flash-on-devices-book-review/ 2004-06-07T20:00:00Z
Operator strategies for mobile broadband in emerging markets
http://telecomyatra.afaqs.com/research/index.html?category=Technology&s=rss
The report analyses strategies available to operators wishing to deploy mobile broadband services in emerging markets
Mon, 07 Jun 2004 20:00:00 GMT
http://telecomyatra.afaqs.com/research/index.html?category=Technology&s=rss 2004-06-07T20:00:00Z

TEST # 3


C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 3.xml - Windows Internet Explorer
C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 3.xml
Yahoo! Search
C:\Users\Heba\Desktop\RSS displayed form tetsi...
Page
Tools
Top Five RSS Feeds Related to Mobile Applications Development
Nokia Qt SDK 1.0 review
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/
A review of the Nokia Qt SDK, a framework allowing you to publish native apps to Symbian and Maemo devices. A nice tool, but there's plenty to learn and it's just early days still. Read on to find out why Qt could be more challenging than learning iPhone, how to get started and problems you might encounter.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/ 2004-06-07T20:00:00Z
Oracle sues Google for Android Java use
http://www.i-programmer.info/news/83-mobliephone/1163-oracle-sues-google-for-android-java-use.html
Oracle wants to stop Android in its tracks because it alleges that the OS uses Java-related patents. Could this be the end of Android and a retreat into proprietary mobile operating systems like Windows Phone 7? Read more...
Mon, 07 Jun 2004 20:00:00 GMT
http://www.i-programmer.info/news/83-mobliephone/1163-oracle-sues-google-for-android-java-use.html 2004-06-07T20:00:00Z
Windows Phone 7 CTP review
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/
A review of the Windows Phone 7 CTP which allows you to create apps using Silverlight for the new WP7 devices coming in Holiday 2010. One of the best mobile platforms I've come across, this looks incredibly promising. Read on to find out why.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/ 2004-06-07T20:00:00Z
The Official Ubuntu Book (Prentice Hall)
http://www.i-programmer.info/book-watch-archive/1154-the-official-ubuntu-book-prentice-hall.html
In this 5th edition the authors explain Ubuntu 10.04 from start to finish: installation, configuration, desktop productivity, games, management, support, and much more. Among the many topics covered in this edition: Kubuntu, Ubuntu Netbook Edition, and Ubuntu Server. <ASIN:0137081308>
Mon, 07 Jun 2004 20:00:00 GMT
http://www.i-programmer.info/book-watch-archive/1154-the-official-ubuntu-book-prentice-hall.html 2004-06-07T20:00:00Z
Mobile app development trends – what languages should you be learning?
http://mark-kirby.co.uk/2010/mobile-app-development-trends/
Mobile app development is a minefield, there's so many different platforms, and different things to consider. I decided now was a good time to run through the market today, as of July 2010 and try to identify what you as a developer should be learning, or what business owners should be looking at. The quick [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/mobile-app-development-trends/ 2004-06-07T20:00:00Z

TEST # 4


C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 4.xml - Windows Internet Explorer
C:\Users\Heba\Desktop\RSS displayed form tetsing\USER TEST 4.xml
Yahoo! Search
C:\Users\Heba\Desktop\RSS displayed form tetsi...
Page
Tools
Top Five RSS Feeds Related to Mobile Applications Development
Nokia Qt SDK 1.0 review
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/
A review of the Nokia Qt SDK, a framework allowing you to publish native apps to Symbian and Maemo devices. A nice tool, but there's plenty to learn and it's just early days still. Read on to find out why Qt could be more challenging than learning iPhone, how to get started and problems you might encounter.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/nokia-qt-sdk-review/ 2004-06-07T20:00:00Z
Windows Phone 7 CTP review
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/
A review of the Windows Phone 7 CTP which allows you to create apps using Silverlight for the new WP7 devices coming in Holiday 2010. One of the best mobile platforms I've come across, this looks incredibly promising. Read on to find out why.
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/windows-phone-7-ctp-review/ 2004-06-07T20:00:00Z
Mobile app development trends – what languages should you be learning?
http://mark-kirby.co.uk/2010/mobile-app-development-trends/
Mobile app development is a minefield, there's so many different platforms, and different things to consider. I decided now was a good time to run through the market today, as of July 2010 and try to identify what you as a developer should be learning, or what business owners should be looking at. The quick [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/mobile-app-development-trends/ 2004-06-07T20:00:00Z
Flash on Devices book review
http://mark-kirby.co.uk/2010/flash-on-devices-book-review/
Here's a quick review of the book "AdvancED Flash on Devices" by Elad Elrom, Scott Janousek and Thomas Joos, published at Friends of ED. In a nutshell, if you work with Flash and mobile, you must buy it. For the past few months I've been working on a Flash Lite project, and 3 months ago I [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2010/flash-on-devices-book-review/ 2004-06-07T20:00:00Z
Optimising visual components for Flash Mobile
http://mark-kirby.co.uk/2009/optimising-visual-components-for-flash-mobile/
I've been working on some a visual component for a flash mobile app, and thought I'd use it to give you a quick case study on how important it is to optimise everything for mobile. Although this example came from an app for Flash Mobile and Flash Lite, the same problem is suffered by iPhone [...]
Mon, 07 Jun 2004 20:00:00 GMT
http://mark-kirby.co.uk/2009/optimising-visual-components-for-flash-mobile/ 2004-06-07T20:00:00Z

APPENDIX D

Experts' Test Document

American University of Sharjah

College of Engineering

ESM Department - ITM

Just-In-Time Knowledge Management

This research is conducted as part of a Master Thesis in Knowledge Management in the College of Engineering in the American University of Sharjah. The thesis proposes a new algorithm for dynamically capturing knowledge from the web that's tailored to specific user's interest by utilizing concept maps and RSS feeds. This research is conducted to validate the effectiveness of the proposed algorithm

Using the attached concept map about '*What You Need To Know About When You Develop A Mobile Application'* and a test RSS feed with 20 articles, a ranking is generated using the proposed method. As an expert in the development of mobile applications, you need to go through the test RSS feed and evaluate each articles relevance to the context of the concept map. Your ranking will be compared to the ranking generated using the new algorithm.

How to evaluate the RSS feeds.

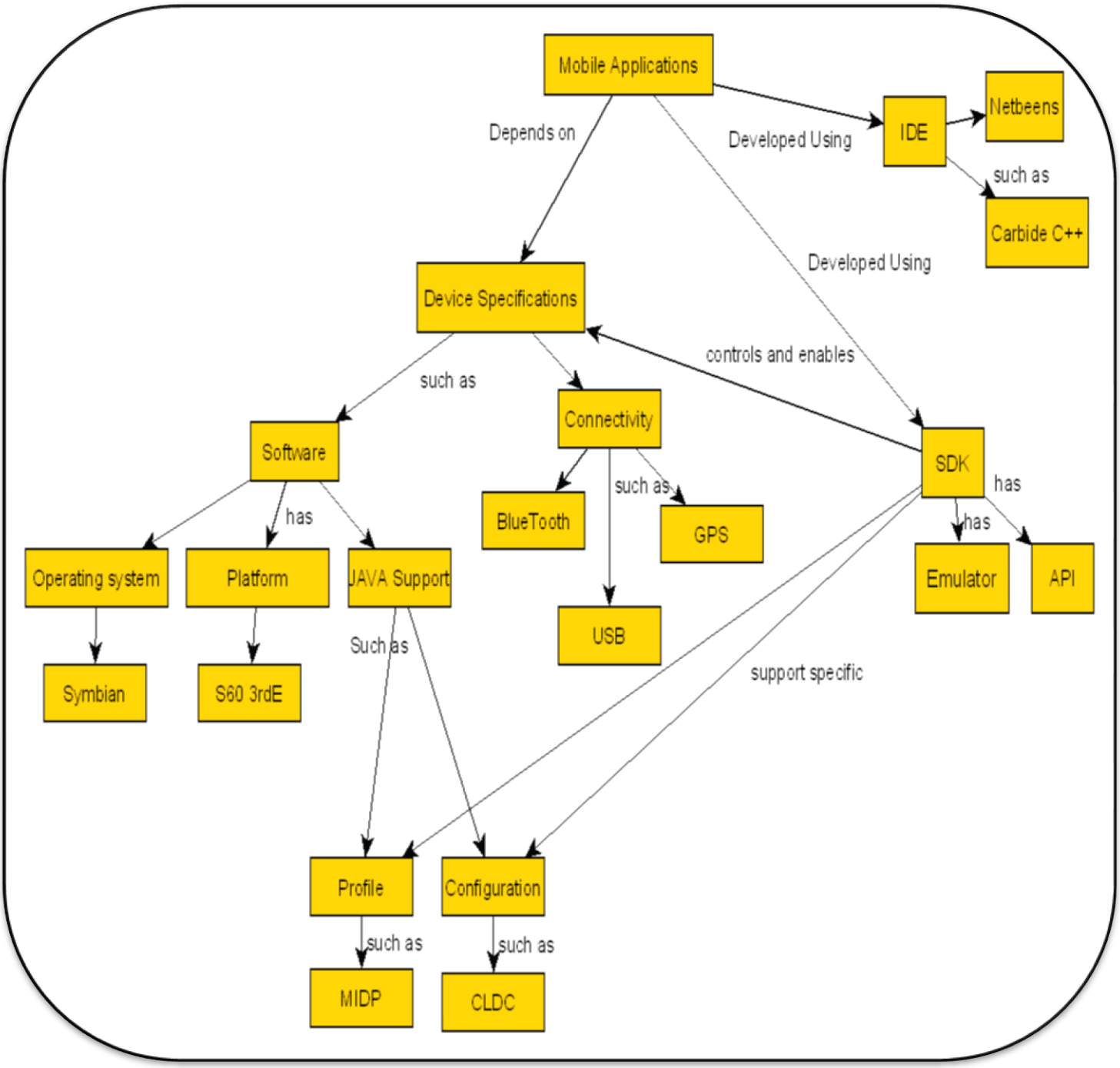
1) First go through the attached concept map and realize the concepts used.
   ➔ The concept map highlights important aspects of mobile application development

2) Read all the articles in the attached RSS feed.
3) Evaluate each feed separately
4) For the given feed rank the 20 articles from 1 to 20.
   ➔ (1) indicates the most relevant article and (20) indicates the least relevant article to the context of the concept map

Please if you have any question or you need any clarification don't hesitate to ask.

Your participation is appreciated.

What You Need To Know About When You Develop A Mobile Application



Test RSS Feed output

RSS URL: http://www.google.com/alerts/feeds/01286230342152329225/17070834082521743714

Date :

Saturday,,June 19, 2010

Please rank the below 20 articles from most relevant to "*What You Need To Know About When You Develop A Mobile Application*" . Assign 1 to the most relevant article

---

Download NetBeans IDE 6.9 Free – SoftSailor

Rank

Yesterday, June 18, 2010, 10:30:10 PM | Marius Maria

NetBeans **IDE** comes with a full-featured package off **applications**. It gives you all you need to create professional cross-platform desktop, enterprise, web and **mobile applications**. The program allows you to design GUIs intuitively. **...** NetBeans **IDE** 6.5 for JavaFX 1.0 supports developing, debugging, and deploying JavaFX **applications**. It is a full-featured development environment that includes the JavaFX 1.0 **SDK**, building block samples, and the JavaFX 1.0 Plugin for NetBeans **...**
SoftSailor - http://www.softsailor.com/how-to/9583-watch-youtube-movies-in-cinema-mode-using-youtube-cinema-firefox-extension.html

iPhone and Mobile App Developers (Kingsway) | Telecommute Job List

Rank

Yesterday, June 18, 2010, 2:38:25 PM | Software Job

An new Start up seeks dexterous iPhone Developers with excellent comfort and efficiency with Cocoa, iPhone **SDK** in XCode and other **Mobile Application** Platforms such as BB, Android with Eclipse **IDE** and jdk. The ability to apply and **...**
Telecommute Job List - Free list... - http://telecommutejoblist.com/

Free Download NetBeans IDE 6.9 / FileHorse.com

Rank

Thursday, June 17, 2010, 2:28:06 AM | (author unknown)

Visual **Mobile** Development Create, test and debug GUI **applications** that run on **mobile** phones, set-top boxes, and PDAs. The NetBeans **IDE** supports JavaFX **Mobile** and the Java ME **SDK** 3.0 Platform. C and C++ Development **...**
FileHorse.com - http://www.filehorse.com/

NetBeans IDE 6.9 Release | MidSofts.COM

Rank	

Thursday, June 17, 2010, 2:28:06 AM | King

The NetBeans project consists of an open-source **IDE** and an **application** platform that enable developers to rapidly create web, enterprise, desktop, and **mobile applications** using the Java platform, as well as JavaFX, PHP, JavaScript and Ajax, Ruby and Ruby on **...** Additional highlights include OSGi interoperability for NetBeans Platform **applications** and support for developing OSGi bundles with Maven; support for JavaFX **SDK** 1.3, PHP Zend framework, and Ruby on Rails 3.0; **...**
MidSofts.COM - Live Software... - http://midsofts.com/

Free AnexTEK moboDA 3160 C# IDE Mobile Software Download

Rank	

Tuesday, June 15, 2010, 5:04:04 AM | (author unknown)

EXE, C# **IDE Mobile** doesn't require you to have the .NET **SDK** (which includes the **...** Then you can compile into native PPC **applications** using Microsoft's **...**
www.pocketpc-freeware.net/.../c-ide-mobile-download-7526.h...

Online (Stream) iPhone SDK 4 beta 4 Build 10M2252 [Intel] | GSM ...

Rank	

Tuesday, June 15, 2010, 5:04:04 AM | (author unknown)

Free download **mobile** phone repair files software cellphone firmware **...** 2.05 GB iPhone **SDK** 4.0 provides support for developing iPhone **applications** and includes the **...** the Xcode **IDE** and the Instruments analysis tool among many others. **...**
www.deltagsm.net/online-stream-iphone-sdk-4-beta-4-build-1...

Free MoSync IDE 2.3 Beta r767 Download

Rank	

Tuesday, June 15, 2010, 5:04:04 AM | (author unknown)

MoSync is a **mobile application** development **SDK** based on common programming **...** Version tracker: You can track versions of MoSync **IDE** 2.3 Beta r767 by visit **...**
just-download.com/software-development/...ide.../17d80

Open source webAndroidiPhoneiPad kit betas for BlackBerry

Rank

Tuesday, June 15, 2010, 4:39:33 AM | Register

The open-source Titanium is a means of building native desktop and **mobile applications** using traditional web-development tools, including JavaScript, **...**
See all stories on this topic

Iphone – the Next Wonder of the World or Just a Flop? | Smart Cellular

Rank

Monday, June 14, 2010, 1:58:34 PM | admin

We focus on the development of **applications** for the Apple iPhone, and we asked Nick Dalton, iPhone **SDK** specialist, to give Elance buyers a few tips on how to scope and post an iPhone **Application** project. **.....** You can either download the Free iPhone **SDK** which include the Xcode **IDE**, Instruments, iPhone simulator, frameworks and samples, compilers, Shark analysis tool, and more. Or you can choose to enroll into the iPhone developers program at standard of enterprise program **...**
Smart Cellular - http://www.cell-mobile.net/

Iphone ? the Next Wonder of the World or Just a Flop? | Bzyk

Rank

Monday, June 14, 2010, 1:58:34 PM | admin

HSUPA does promise a future path for faster data, and will no doubt unlock new **applications** for future phones. Along with the confusing acronym soup of HSDPA and HSOPA, HPUPA competes against other **mobile** 3G standards. **....** You can either download the Free iPhone **SDK** which include the Xcode **IDE**, Instruments, iPhone simulator, frameworks and samples, compilers, Shark analysis tool, and more. Or you can choose to enroll into the iPhone developers program at standard of **...**
Bzyk - http://www.bzyk.net/

Setting the Environment - Aspose.BarCode for Java

Rank

Monday, June 14, 2010, 2:41:19 AM | (author unknown)

We will use Netbeans **IDE** and Java Micro Edition **SDK** 3.0 emulator in our examples **...** Select "Java ME" from the categories and "**Mobile Application**" from the **...**
www.aspose.com/.../java...for.../setting-the-environment.html

NetBeans web-content: Web Content Subversion: trunk: www: features ...

Rank

Monday, June 14, 2010, 2:41:19 AM | (author unknown)

features.css"> </head> <body> **...** JavaFX **SDK** 1.2.x supports JavaFX **Mobile** and comes with a JavaFX **Mobile** Emulator. JavaFX **Mobile applications** run directly on the Java Micro Edition platform: **...**
share.java.net/projects/web-content/sources/.../index.html?...

Beginner's Guide for DotNet

Rank

Monday, June 14, 2010, 2:41:19 AM | (author unknown)

NET Framework **SDK** You can develop such varied types of **applications**. **...** In such cases you require some Integrated Development Environment (**IDE**) that allows for Rapid **....** NET can start developing for **mobile** devices almost instantly. **...**
www.docstoc.com/docs/.../Beginners-Guide-for-DotNet

iPhoneDevCon - San Diego

Rank

Monday, June 14, 2010, 2:41:19 AM | (author unknown)

The iPhone has enable a new class of **application** that is both aware of a user's **...** Apple's Xcode **IDE** and the iPhone **SDK** for building iPhone **applications**. **....** make the most sense in the context of **mobile application** development. **...**
www.iphonedevcon.com/classes.html

some thing which is new to globe: Overall info in our com science

Rank

Sunday, June 13, 2010, 10:52:26 PM | pvvijay

IDE-integrated development environment 1. Git 2. Net weaver 3. Eclipse **Mobile** os 1. Android 2. Windows **mobile** os 3. Linux **Mobile** apps **sdk**-s/w development kit 1. Symbian 2. Iphone **sdk** 3. Java 4. Blackberry **sdk**. Browser engines **...**
some thing which is new to globe - http://pvvijay.blogspot.com/

Adding Application Icons | MoSync

Rank	

Sunday, June 13, 2010, 7:56:36 PM | (author unknown)

Please keep in mind that this feature of the MoSync **SDK** is experimental. **... Mobile** World Congress in Barcelona, a success for **application** developers **...**
www.mosync.com/documentation/.../adding-application-icons

Blackberry Java Application Development | Free Download Verizon ...

Rank	

Sunday, June 13, 2010, 7:56:36 PM | (author unknown)

The BlackBerry Widget **SDK** (Software Development Kit) v1.0 and the **...** Software Development: Sub Category: **Mobile Apps**: Skills: PHP java BlackBerry . **...**
verizonsaga.com/.../blackberry+java+application+development

netbeans IDE 6.8 Payroll Program (Java) - WorkingBase.com

Rank	

Sunday, June 13, 2010, 7:56:36 PM | (author unknown)

We have created a **mobile application** that runs on java enabled handsets like Nokia, Motorola etc **...** Hi, I need a working sample of Facebook Android **SDK**. **...**
www.workingbase.com/.../netbeans-IDE-6.8-Payroll-Program...

Software Application Developer jobs - Dice.com

Rank	

Sunday, June 13, 2010, 7:56:36 PM | (author unknown)

Authoring **Mobile SDK**. - Developers must be proven software engineers experienced **...** 3) Widget **application** development Expert Required 4) **Mobile** device **IDE ...**
seeker.dice.com/jobsearch/servlet/JobSearch?...

Download Droid Apps Software - Downloads Free Droid Apps : Xentrik

Rank	

Sunday, June 13, 2010, 7:56:36 PM | (author unknown)

AppStore **SDK** scrapes all apps from **App** Store AppStore **SDK** enables you to scrape all **...** Add a Calculator Property Editor to the Rave **IDE** and a Coloured Text **...** Esperanto-English Dictionary by Ultralingua for Windows **Mobile** Pro 6.2 **...**
www.xentrik.net/software/downloads/droid_apps.html

VITA

Heba Mahmoud Ismail was born on March 26,1985, in Al Ain, United Arab Emirates (U.A.E). She was educated in local public schools and graduated from Al-Manaseer High School in 2003. She received Sheikh Hamdan Al-Nahyan Best Achievers Scholarship to The University of Abu Dhabi in Al Ain, UAE and graduated suma cum laude in 2007 with Bachelor of Science degree in Information Systems Technology. After Her graduation Ms. Ismail joined Emirates Group-IT as a Software Engineer for one year. In 2008 Ms. Ismail joined the Master's program in Engineering Systems Management at the American University of Sharjah and received a graduate teaching assistantship. She was awarded the Master of Science degree in Engineering Systems Management in 2011.