An IR-Based Semantic Search Approach on Portable Document Format (PDF) Files Using Similarity Measure

A Thesis
Presented to the
Department of Computer Science
Institute of Information and Computing Sciences
University of Santo Tomas

In Partial Fulfilment
of the Requirements for the Degree
Bachelor of Science in Computer Science

by

Agsunod, John Mark Robert M.
Banting, Carl Jayson M.
Brar, Harjit S.
Cunanan, Patrick Bryan F.

Adviser:

Ms. Charmaine S. Ponay

December 2017

University of Santo Tomas
Institute of Information and Computing Sciences
Department of Computer Science

Certificate of Authenticity and Originality

We, the authors of this thesis, "An IR-Based Semantic Search Approach on Portable Document Format (PDF) Files Using Similarity Measure", hereby certify and vouch that the contents of this research work is solely our own original work; that no part of this work has been copied nor taken without due permission or proper acknowledgment and citation of the respective authors; that we are upholding academic professionalism by integrating intellectual property rights laws in research and projects as requirements of our program.

If found and proven that there is an attempt or committed an infringement of copyright ownership, we are liable for any legal course of action sanctioned by the University and the Philippine laws.

John Mark Robert M. Agsunod

Carl Jayson M. Banting

Harjit S. Brar

Patrick Bryan F. Cunanan

Program: BS Computer Science

Date: December 06, 2017

University of Santo Tomas

Institute of Information and Computing Sciences

Department of Computer Science

ACKNOWLEDGEMENT OF UNDERTAKING

This is to acknowledge the great efforts contributed by each member in our thesis group towards completing this research output.

In view of our commitment to make a significant imprint toward research and innovation, we strongly adhere that our research be presented in a public research forum or conference and/or published in a reputable research journal where a greater number of researchers may find our research endeavors meaningful and useful.

In this undertaking, we support that any member of our group including our thesis adviser (as co-author) may represent our group for presentation and/or publication of our research work sole for the purpose of further academic research and not for any material gains. It is therefore understood that anyone of us intending to present and/or publish our thesis must fully acknowledge that the copyright remains with the original authors. Henceforth, proper acknowledgement, recognition and credit must be duly accorded to all members of the group.

Agsunod, John Mark Robert M.
Date Signed 12/06/17

Banting, Carl Jayson M.
Date Signed 12/06/17

Brar, Harjit S.
Date Signed 12/06/17

Cunanan, Patrick Bryan F.
Date Signed 12/06/17

Ms. Charmaine Ponay
Date Signed 12/6/17

Table of Contents

List of Figures

Page

List of Tables

Page

List of Equations

Page

ACKNOWLEDGEMENTS

The researchers would like to extend their earnest gratitude to Almighty God,

For bestowing upon them wisdom and intelligence to complete this work,

To our panel members,

For giving their prized time to provide utmost support and advices,

To our thesis adviser, Ms. Charmaine S. Ponay,

For her guidance and valuable opinions for the betterment of our thesis,

To our peers and acquaintances,

For their assistance and accompaniment in favorable and trying times,

And to our ever supportive parents and family members,

For providing for our needs and understanding our endeavors.

Thank you very much.

The Researchers

ABSTRACT

Information Retrieval is finding material, usually documents, of an unstructured nature, usually text, that satisfies an information need from within large collections, usually stored in computer databases.

This research introduces a semantic search engine that is able to generate a list of relevant documents that are related to a certain user query faster than current search engines by implementing the similarity measure in its search process. A familiar user interface would be adopted by the implemented system that is generally known to almost everyone to ensure ease of use, efficiency, and usability.

As for existing semantic search engines, the implemented system would be redesigned in order to perform better in terms of speed and accuracy. This is done by introducing the similarity measure, along with several algorithms into its search process. The evaluation of these factors were calculated using the precision, recall, and F1-measure tools in order to ensure the reliability and efficiency of the implemented system.

The evaluation results show that the accuracy of the implemented system stayed the same and the actual time or speed is decreased by 30%, compared to the existing system.

Chapter I The Problem and Its Background

A. Introduction

The idea of searching is changing. From the time when search engines were first introduced, a plethora of advancements and additions have been applied to traditional search engines in order to accommodate the rapid influx of new information being made available, both in the physical and digital world. In order to cope with this rapid data growth, several techniques were created to find information that the user needed as swiftly and accurately as possible. One of the developed search techniques is information retrieval. Information retrieval is a technique that mixes computer science and information science. Christopher Manning wrote that Information retrieval (IR) is "finding material, usually documents, of an unstructured nature, usually text, that satisfies an information need from within large collections, usually stored on computers". Certain applications of Information Retrieval in our activities and in the industry include activities done by paralegals, librarians and other professional searchers (Manning et al., 2009).

Information Retrieval and Information Extraction may sound like the same concept, but it differs by the way each concept works. As stated by G. Chowdhury, Information Retrieval is different from information extraction in a way that IR is getting a set of results based on a set of keywords given by the user. Information extraction, on the other hand, is more on getting facts inside a document based on a given query. "The goal of making an information retrieval (IR) system is to aid in recovering and accessing documents". G. Chowdhury further explains, "IR systems are designed to analyze, process and store sources of information and retrieve those that match the user's preferred requirements" (Chowdhury, 2010).

B. Background of the Study

The field of information retrieval has been progressively developing along with the development of search engines and the World Wide Web. According to Wall, in the year 1990, the first search engine was brought to life by Alan Emtage. Alan Emtage, as cited by Wall, is a member of the Internet Society and has lead projects of the Internet Engineering Task Force (IETF) which is the standard-setting body for the Internet. Moreover, Emtage's creation, named *Archie*, is the world's first search engine ever built. Wall wrote, "It was a pre-Web search engine for locating material in specific public FTP archives. Though it only operated for a short while and updates were ceased later that same year, similar search engines were formed in likeness to *Archie*, such as *Jughead*, produced by the University of Minnesota, and *Veronica*, which was developed by the University of Nevada to search on plain text files" (Wall, 2006).

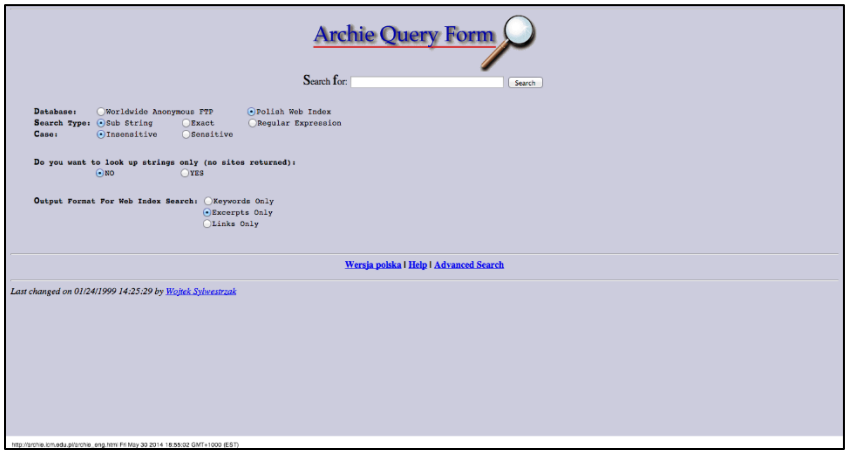


Figure I-1. Archie Search Query Form
(Source: http://www.searchenginehistory.com/)

The use of information retrieval has been used in many fields such as medicine and engineering, considering the large amounts of jargons and expansive data gathered and documented on those fields. Information retrieval can greatly help generate a compact and sound compilation of relevant topics whenever they are needed. One area that the research

team aims to improve knowledge on is data specific to technology, specifically in the categories of Word Disambiguation, Information Extraction, Image Processing, Data Compression, and Semantic Search.

With technology specific to Word Disambiguation, Information Extraction, Image Processing, Data Compression, and Semantic Search as the research domain, the team can implement an approach to develop a search engine agent that generates a set of selected documents from a database, which is then processed with IR algorithms to produce a collected list of documents related to the terms determined in the user query. In the study, where most of our study is based upon, entitled "*A semantic search technique with Wikipedia-based text representation model*" (Hong et al., 2016), a 3rd-order tensor model was made by using the user query to produce a concept window wherein terms are selected and run through the concept space using the TFDIF equation, in order to produce a concept vector for those concepts in the concept space (Hong et al., 2016). This study then proceeds in using information retrieval (IR) strategy and three searching algorithms, namely *Keyword only, Query expansion*, and *Concept only* algorithms, to come up with a list of documents that scores the most value computed.

The research team identified that while this approach produced a list with high accuracy and correctness in terms of similarity function value, the 3rd-order tensor model can be much more time efficient and compact. The efficiency is achieved by introducing a model pool and by removing irrelevant documents among the concepts in the selected model.

As further explained in the conceptual framework; see Figure I-3, the user query undergoes a process to construct query matrix where each value in the matrix tells the

relevance value between the query and the 3rd-order tensor model before it is compared to the query matrix of previous queries recorded in the system. By this comparing process, the engine can decide whether the user query undergoes a process for constructing a new 3rd-order tensor model or not.

For the construction of a new 3rd-order tensor model, the Wikipedia pages are considered as the concepts in which the concept vector uses to construct a concept matrix, and these Wikipedia pages are compared to the documents for constructing a concept space composed of those concepts or Wikipedia pages that has a high relational value with the corpus of document set. At the same time, a document matrix is constructed using the concepts and the terms from all the documents in the set. The values in the matrix are created using TFIDF value of terms in the concepts. The list of terms in the documents are again used for constructing a list of concept window for all the terms. Concept window is a set of terms that surrounds the terms in the list. Using these surrounding terms, the meaning of the terms can be discerned in the process of comparing them with the concepts and thus constructing the concept matrix. The concept matrix and document matrix are combined to create the 3rd-order tensor model.

After that, the lists of concepts in the 3rd-order tensor model are arranged in a descending order. This is done for the preparation of removing unwanted concepts in the searching process of the system. After the sorting process of the model, the model is summarized in to a 2nd-order tensor model. The other term for 2nd-order tensor model is super matrix. The super matrix is used for the comparison in searching relevant documents in the 3rd-order tensor model.

In the searching process, the user query is used to construct user query concept space is created by extracting concepts from the concept space created in the construction phase of the system. The size of this user query concept space is used to remove unwanted concepts in the models created in the construction phase. The refined super matrix is then used to create a query matrix that is used to search in the refined 3rd-order tensor model and in the result set pool. Query matrix is a matrix created by searching through terms in the super matrix and marking unmatched terms' value zero. Result pool is a database of previously made queries. It stores information produced in the construction phase and in the searching phase. This pool is used before searching through 3rd-order tensor model in hopes for finding similar query for the current query that is being searched. If a matched query is found, then its produced output is shown to the user. If not, then the query matrix is used to search through the 3rd-order tensor model.

C. Theoretical Framework

This study is based from a study entitled "*A Semantic Search Technique with Wikipedia-based Text Representation Model*" (Hong et al., 2016)

Information Retrieval (IR) is an application of Natural Language Processing (NLP), which is a field under Computer Science. It deals with the representation, storage, organization of, or access to information items. These information items refer to real documents, Web pages, spoken documents, images, pictures, music, video, etc. (Hong et al., 2016). There are three Information Retrieval models, namely; the *Boolean Model*, the *Vector-Space Model*, and the *Probabilistic Model*.

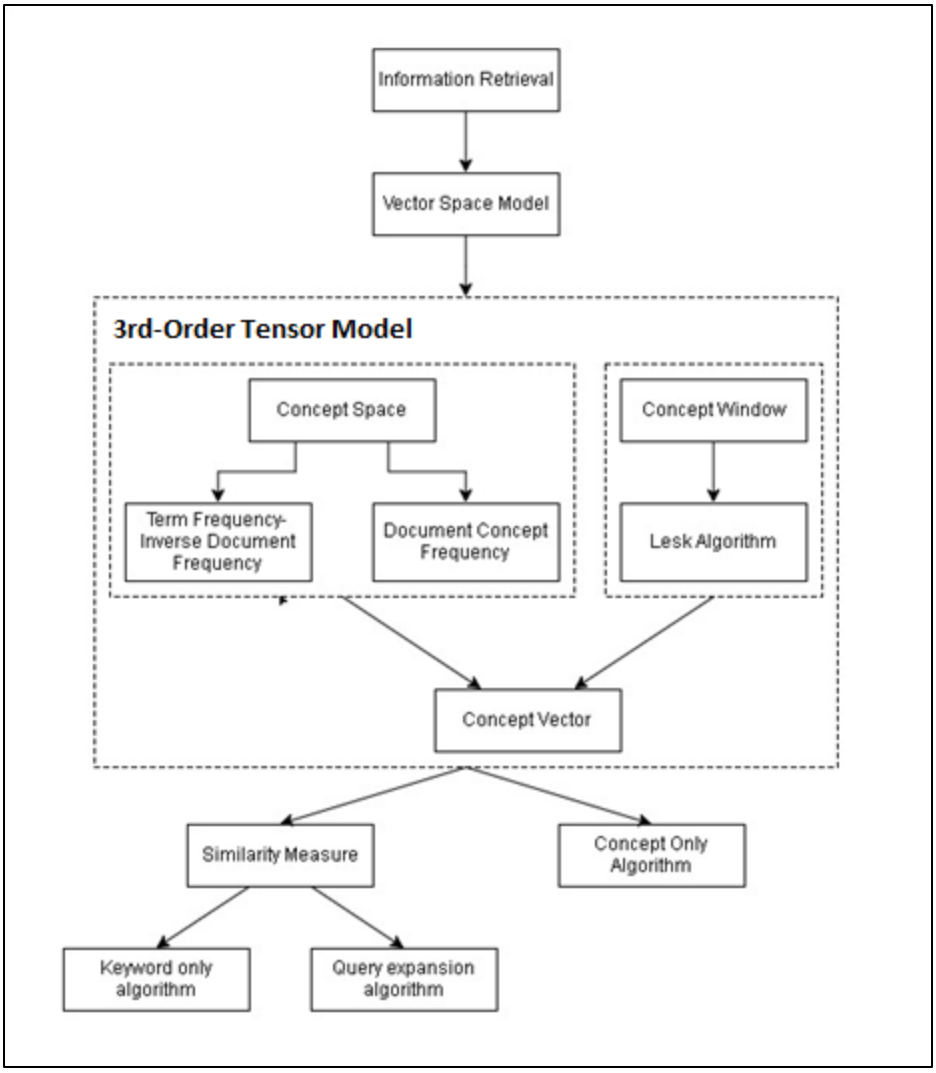


Figure I-2. 3rd-order Tensor Model Framework

The researchers' study is based on *Vector-Space Model*. According to Salton, "vector-space model, documents and queries are represented by vectors in a high dimensional space in which each indexing term corresponds to one dimension". Furthermore, Salton explained that the "elements of the vectors may be binary, indicating the presence or absence of the term, or fractional weights indicating the relative importance of the term in the document or query" (Salton, 1971).

A study made by Ki-Joo Hong and Han-Joon Kim utilized this vector model using the definitions of the terms and concepts defined in the formal concept theory in forming a 3rd-order tensor model. The model was created by constructing three components and combining them. These components are concept space, concept window, and concept vector.

The concept space was formed by mapping pages from Wikipedia with the sets of documents using a third party system called *Apache Lucene*. In order to map these documents, they used an algorithm called *term frequency-inverse document frequency* (TFIDF) which determines the importance of a word or text from the given document. To extract significant concepts, the researchers only selected concepts with high DCF (document concept frequency). Document concept frequency is the frequency of a particular concept in the entire set of documents.

The concept window was constructed by using *Lesk algorithm*; see Figure II-3. The *Lesk algorithm* works by returning a set of target terms where each target terms have a set of definitions or glosses and a set of surrounding words. After that, the concept space is then compared to the concept window to form 3rd-order tensor model. The process is done by having each terms in the glosses of a target term compared with the concepts in the concepts space using TFIDF.

The search engine used three algorithms, namely; the *Keyword Only* Algorithm, *Query Expansion* Algorithm, and the *Concept Only* Algorithm, along with similarity measure function. The first algorithm, which is the *Keyword Only* Algorithm, used simply the similarity measure on each pair of the term by document matrices. The second one, which is the *Query Expansion* Algorithm, used a concept from the concept space that has

a high importance factor with the query, and combines it with the similarity measure function. Finally, the third algorithm, called the *Concept Only* Algorithm, used the selected concept that has the high importance factor to compare it to the 3rd-order tensor model.

D. Conceptual Framework

The conceptual framework; see Figure I-3, is derived from the existing 3rd-order tensor model but modified in order to try to improve the time spent by the searching algorithm to find the relevant documents per concept and to maintain the accuracy of the previous research.

In Figure I-3, the conceptual framework of the system has two phases, namely; Tensor Construction Phase and Search Phase. Tensor construction phase consists of processes that constructs and sorts the models which are used as resources for the search phase. It accepts documents and wiki pages to create a concept space, and concept window. The concept space's and concept window's outputs are used to create 3rd-order tensor model with the help of concept vector process. After the creation of 3rd-order tensor model, the list of concepts in the tensor are sorted using the merge sort algorithm. The sorted 3rd-order tensor model is then used to produce the 2nd-order tensor model or the super matrix. These models are stored in text files for the second phase.

The search phase uses the outputs of the tensor construction phase and the query from the user to produces set of documents to the user. This is done by reading the resources and constructing a user query concept space using the user query. User query concept space are list of concepts searched from the concept space produced from the tensor construction phase.

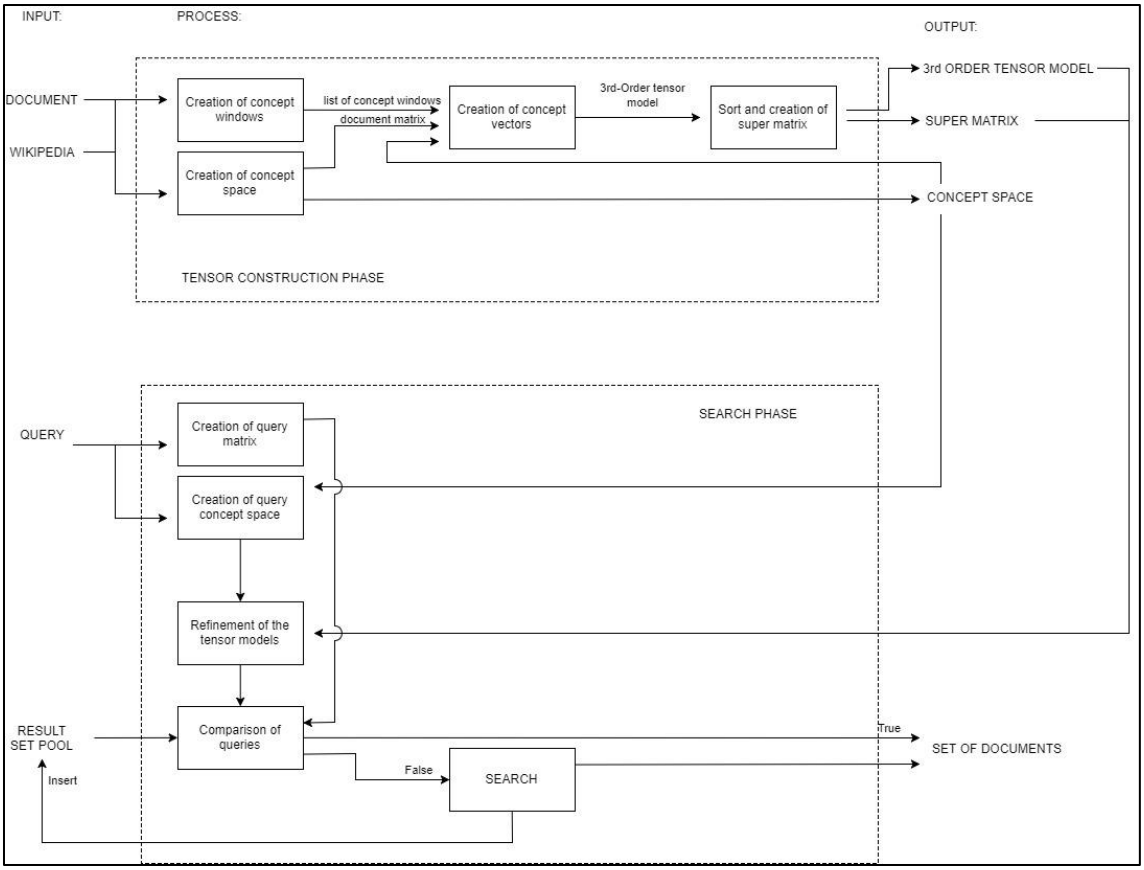


Figure I-3. Conceptual Research Framework

The size of the user query concept space is used to remove unwanted concepts in the list of concepts in the 3rd-order tensor model and in the super matrix. After the process of removing unwanted concepts, the query is used to create a query matrix from the super matrix. Query matrix is the same as the super matrix except that the values of terms in the query are retained and the others are changed to zeros. The query matrix is first used to search through the result set pool and if there is a match in the result set pool, then the resources stored in that specific record are returned to the user. However, if the match is not found, the search phase uses the model read from the first phase and searches inside the 3rd-order tensor model. At the end of the process of searching the model, the resources found are stored in the pool for later use.

E. Statement of the Problem

There are a number of studies made in order to improve information retrieval engine. The basis for the researchers' study, "*A semantic search technique with Wikipedia-based text representation model*" by Ki-Joo Hong and Han-Joon Kim, is one of these studies. However, the research team has observed that the search algorithm used is run on entire concepts, which, theoretically, costs more time and resources.

The research team aims to further decrease the size of the concepts by applying the similarity measure of the query to each of the concept in the model and filter out irrelevant concepts on these documents in the process. In this way, the search algorithm can theoretically run faster as less terms or concepts are being processed.

The study would look into these problems:

1. What is the actual amount of running time of the researchers' study on documents related to Word Disambiguation, Information Extraction, Image Processing, Data Compression, and Semantic Search technologies, as compared to the actual time of the existing study?
2. What is the accuracy of the retrieved documents as compared to the accuracy of the approach of the existing study by Hong and Kim?

F. Objectives

This study aims to improve the existing techniques used by Kim and Hong in their study entitled "*A semantic search technique with Wikipedia-based text representation model*". The study intends to decrease the time it takes to access data or produce results. To do so, it should be able to accomplish the following specific objectives:

1. to use cosine similarity measure function in the records of the result set pool for the reduction of extra process in searching through the model.
2. to compare the actual amount of running time of the researchers' study against the existing study for semantic search.
3. to compare the accuracy of the existing study against the researchers' implemented study.

G. Scope and Limitations

The study is limited by the following scopes:

1. The search engine that is produced only runs offline or on a localhost and is not web-based.
2. The database of the search engine is also maintained offline.
3. The database of documents that the search engine consists only of .PDF (Portable Document Format) files.
4. The database contains PDF files that are approximately 15,000 documents taken from Wikipedia and 250 documents taken from IEEE Database.
5. The PDF files are all in the scope of the English language and uses the English alphabet and grammar rules.
6. The user query is limited to use the English language, its alphabet and corresponding grammar rules.
7. The user query need not form a complete sentence to be accepted and processed by the search engine.
8. The user query must not be in a question form, but must either be a phrase or a sentence in declarative form.

9. The user query's length, in characters and word forms, is not restricted in this study, but must contain at least one word that is not a stop word (see *Appendix F* for the Table of Stop Words and Symbols)
10. The significance of accuracy and the reliability of query results along with the precision and recall measures are quantitative measures.
11. This study is applied to documents on technology specific to Word Disambiguation, Information Extraction, Image Processing, Data Compression, and Semantic Search only.
12. The user input can contain punctuation marks or other symbols such as comma (,), hyphen (-), etc. but these marks are removed during input processing since they do not hold any meaning compared to the vocabulary words in the query.
13. If the user uses the keywords "and" or "or" in their user query, the search topics involved must be related to one another.
14. The PDF documents contain only the abstract part of the documents.
15. The PDF documents should be linear in formatting, without any charts or graphs at the sides and only single-columned.

Limitations of the Study

Some limitations that are out of the researchers' control are the speed and performance of hardware devices such as laptops, desktops, and the like and erroneous and inappropriate contents and data in the sample documents. Sample data gathered for the experiment also does not represent the entire available domain and resources stored on the IEEE data repository.

H. Significance of the Study

This study aims to compare the search accuracy and improve the response time of the current semantic search engine. This study is a significant endeavor for students who would like to seek for documents regarding the domain on technology related documents specific to Word Disambiguation, Information Extraction, Image Processing, Data Compression, and Semantic Search; it may also be beneficial for students, researchers, technologists and professors who can use it as an effective way of looking for certain techniques and functions relating to these technology concepts in order to conduct their own researches and studies. This study can be used as a reference and as a basis for analysis and comparison for their future works.

I. Definition of Terms

3rd-order tensor model. A model of a geometric object with a dimensionality of 3 that describe linear relations, such as similarity and cross products, between geometric vectors and scalars.

Algorithm. A set of rules for solving a problem in a finite number of steps, as for finding the greatest common divisor.

Cognitive synonyms. Synonyms that are so similar in meaning that they cannot be differentiated either denotatively or connotatively, that is, not even by mental associations, connotations, emotive responses, and poetic value.

Concept space. A geometric structure that represents a number of quality dimensions.

Concept vector. A matrix of one row containing TF-IDF values on a particular concept captured by the concept window.

Concept window. A set of general terms derived from the query of the user using an algorithm, such as the *Lesk Algorithm*.

CRUD. expanded as *Create, read, update, and delete*, these are the four basic functions of persistent storage in computer programming.

File Transfer Protocol (FTP). File Transfer Protocol; a standard Internet protocol for transmitting files between computers on the Internet over TCP/IP connections.

Frobenius Product. also called *Frobenius inner product*, a binary operation that takes two matrices and returns a number.

Hash Map. also called hash table, a data structure which implements an associative array abstract data type, a structure that can map keys to values and relative addresses.

Information Retrieval. It is a method of finding and fetching information from stored data.

Localhost. A hostname that means the user's own personal computer. It is used to access the network services that are running on the host via its loopback network interface.

Metadata. Data that provides information about other data.

Model. A simulation to reproduce the behavior of a system.

Module. An interchangeable, independent part of a functionality, each containing everything necessary to execute only one aspect of the desired functionality.

Ontology. A formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse.

Paralegal. An attorney's assistant, not admitted to the practice of law but trained to perform certain legal tasks.

Parameter. Generally, any characteristic that can help in defining or classifying a particular system.

PDF. expanded as Portable Document Format, a file format developed by Acrobat Systems, that is usually used to save documents which have more complex contents rather that simple text.

Plethora. An overabundance or excess of something.

Query. A question or an inquiry.

Research domain. A field of action, thought, influence, etc.

Semantics. The study of meanings.

Semantic Search. A searching technique which does not concern only about the keywords, but also the meaning and context of the words on the query.

Similarity Measure. A measure that defines similarities between two objects.

Vector. An element of a vector space, which is also called a linear space.

Chapter II Review of Related Literature and Studies

This chapter includes the related studies and literatures used by the researchers for the study. These include the ideas and concepts about semantic search, algorithms used on semantic searching and existing applications of semantic search engines. Ideas on information retrieval, methods of IR, and current use of information retrieval on the modern world are also discussed.

A. Semantic Search

Merriam-Webster dictionary define semantics as meaning or relationship of meanings of a sign or set of signs**.** Applied to search, "semantics" essentially relates to the study of words and their logic. Semantic search seeks to improve search accuracy by understanding a searcher's intent through contextual meaning (Sander, 2016). According to Amerlands, semantic search is like a searchlight because it picks up all the different data nodes of the Web and follows them around creating a picture of how they link up, who they belong to, who created them, what else they created, who they are, who they were, and what they do (Amerland, 2014).

Teodora Petkova defined semantic search as a technique that, "reaches out beyond keywords and seeks to understand the semantics of the search query. It improves search accuracy by looking at both data and their connections". She also added that, "instead of more links, which are only a single kind of relation, the algorithm presents you with a networked view of relations, facts, information, you might not know even existed" (Petkova, 2016). Figure II-1 shows how the semantic search process works.

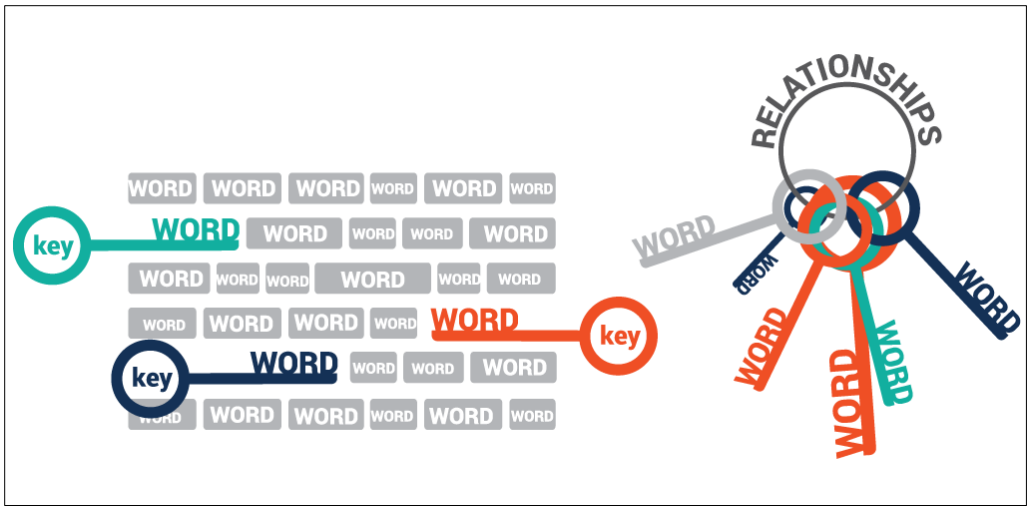


Figure II-1. Semantic Search Process Diagram
(Source: http://ontotext.com/semantic-search-the-paradigm-shift-from-results-to-relationships/)

Brief History of Semantic Search

Semantic search has evolved throughout the decades of intensive research and heavy investments of time and knowledge by researchers, professors, scientists, and engineers. As stated in *semanticweb.rs*, that as early as the 1980s, significant research appeared in information science literature about the development of expert systems for improving search results. Hundreds of universities, start-up companies, and major corporations have published research and filed patents on various algorithmic techniques for machine-aided searching over three decades (Petrovic, 2010).

In 1998, Tim Berners Lee, during the infancy of the World Wide Web, thought of the idea of semantic searching when he published his report entitled “Semantic Web Road Map”. According to Samuel Edwards, Lee’s introductory sentence to his report, which states that “a road map for the future, an architectural plan untested by anything except thought experiments”, has well foreseen the future before us with much more than just

mere thought experiments backing up the acceptance of the public to semantic search (Edwards, 2015).

Modern Application of Semantic Search

Semantic search has been applied to modern search engines and sites many times, due to its convenience and accuracy to the user query. As web search engines continue to improve, good results to such a query have become a reality (Petrovic, 2010). According to Petrovic, this type of Q & A often makes use of a semantic technology called "natural language processing," one of many related technologies that comprise the semantic software technology landscape (Petrovic, 2010). Figure II-2 shows a sample semantic search output using Google.

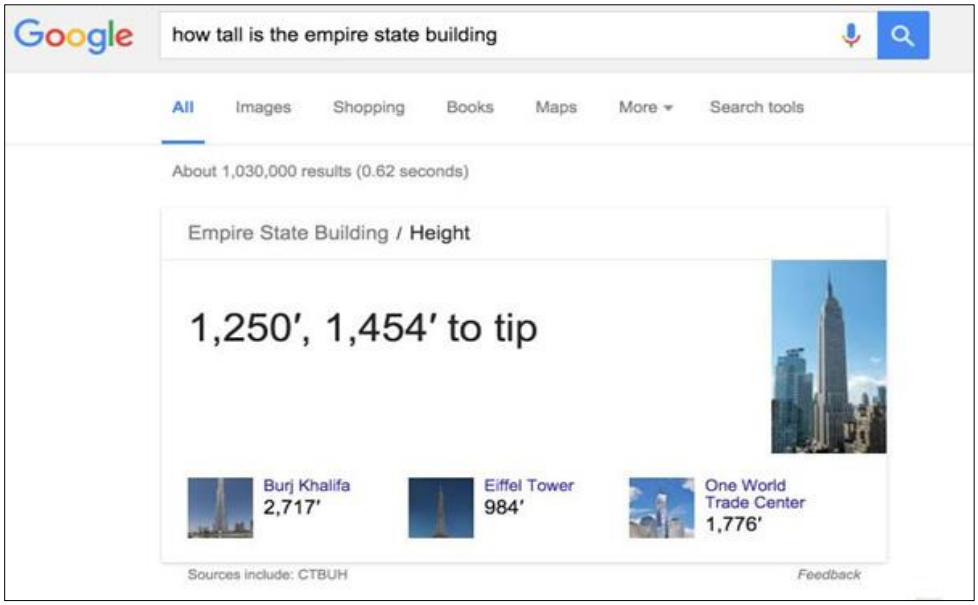


Figure II-2. Sample Semantic Search Engine Output from Google retrieved May 19, 2017
(Source: http://www.semanticweb.rs/Article.aspx?iddoc=32&id=65&lang=2/)

However, in the industrial world, expectations for relevant search results are much higher than for finding content already optimized for e-commerce on the web. Each business unit in an organization has specialized requirements for finding information needed to do its work more efficiently. This is where other types of semantic processing to be implemented can give organizations a competitive edge to other enterprises by getting workers to answer more swiftly, with more conceptual relevance, and even with pinpoint, on-the-dot accuracy. The idea is to get only the right information, only relevant, and all the right information, everything in a document that is relevant.

Challenges to Semantic Search Engines

Based on the paper, "New Framework for Semantic Search Engine" by Arooj Fatima et.al; they noted that "a semantic web search engine should be able to search data in a data store or database with maximum precision and accuracy and be able to link related data through data analysis and usage of the search algorithms." The research team has considered the following criteria in the creation of their project semantic search engine: user experience, efficiency, file ranking, scalability, and cost effectiveness (Fatima et. al., 2014).

User Experience

The user interface (UI) of the semantic search engine must be friendly to all users. Veteran players in the search engine industry, such as Google, Yahoo, and Bing have rigorously and continuously modified and developed their interfaces so as to accommodate more audiences in the most comfortable and friendly way possible (Fatima et. al., 2014).

Efficiency

A search engine's efficiency may depend on several factors. One factor is the size of the data store where the keywords are searched and analyzed. The larger the database size, the slower the time it takes to locate and pair each keyword with its corresponding meaning or context. Another factor could be the environment where the data store runs on. The efficiency of a search engine really depends on the factors in its environment (Fatima et. al., 2014).

File Ranking

Page ranking is a way of grading documents so that the documents with the highest ranking are presented at the top of a list of search summary results Syntax-based search engines use various algorithms to calculate relevancy of a web page to the search terms and arrange the results sorted with highest relevancy at the top (Fatima et. al., 2014).

Scalability

Scalability is the ability of the search engine to handle and adjust a rapidly growing amount of data. In a technology-savvy world today, the amount of data that constantly moves around us is gradually, maybe even exponentially burgeoning (Fatima et. al., 2014).

Cost Effectiveness

A good search system must provide a solution that is cost effective. It should be cost effective to both the developer of the program to the end user, without ever affecting the accuracy, integrity and correctness of the data (Fatima et. al., 2014).

B. Information Retrieval

As defined by the Cambridge Press, *information retrieval* or *IR* is finding material, usually documents, of an unstructured nature, which are usually text that satisfies an information need from within large collections that are usually stored on computers

(Manning et. al., 2006). IR used to be an activity that only a few people really engaged in, but as the times change and technology continuously shapes the modern world, hundreds of millions of people engage in information retrieval daily, such as using web search engines like Bing or Google, and searching personal emails (Manning et al, 2006). Information retrieval is basically concerned with the retrieval of relevant information or documents from data stores or databases (Tan et. al., n.d.). IR is basically concerned with facilitating the access of a user to huge amounts of information, which are predominantly textual in nature (Tan et. al, n.d.). The documents can be books, journals, reports, or other records of thought, or any parts of such records; namely articles, chapters, sections, tables, diagrams, or even particular words. The retrieval devices can range from a bare list of contents to a large digital computer and its accessories. The range of the operations can be from simple visual scanning to the most detailed programming methods (Vickery, 1959).

An information retrieval system stores units of information. Each unit of information stored is linked in the system to specifications of one or more documents or parts of documents. The user specifies particular units of information and the system is designed to provide him with knowledge of all relevant items recorded in the system (Vickery, 1959).

A retrieval system can be studied at three levels:

1. First, the way in which units of information, and relevant relations between them, are defined in the system. This is the semantic level of subject analysis.

2. Second, the general structural features of the system considered as a network of units of information linked to each other and to documentary items. This may be called structural analysis.

3. Lastly, the physical mechanisms (hardware) in which the structure is embodied (Vickery, 1959).

Brief History of Information Retrieval

As historians very well know, watching the progression of a field often provides valuable insights or understandings into the nature of its problems and possible solutions.

The history of information retrieval is as blooming as the creation of the World Wide Web. Though, the long history of IR does not begin with the internet. According to Sanderson, it was only within in the last decade and a half of IEEE's hundredth year that web search engines have become used and searching has become one of the most used and interweaved function of mobile and desktop operating systems. He adds that, "conventional approaches to managing large collections of information originate from the discipline of librarianship. Commonly, items such as books or papers were indexed using cataloguing schemes. Eliot and Rose claim this approach to be millennia old: declaring Callimachus, a 3rd century BC Greek poet as the first person known to create a library catalogue" (Sanderson et. al., n.d.).

The papyrus scrolls that were invented and predominantly used in ancient Greeks and Romans were "not the most efficient and convenient way of storing information in a written form and of retrieving them", as stated by Forsythe. "It was difficult to manage and maintain. Yet, as Greek and Roman scholars began to write large scholarly works that were compilations of data of various sorts, they found it very useful to devise and think of various means of organizing the material to make locating certain passages easier for the reader" (Forsythe, n.d.).

In the United States, the development of information retrieval started when Hollerith's invention of tabulating machines was used to analyze the census of the US population (Dubin & Smith, 2000). At this time, there was a perceived explosion on the amount of information, mostly scientific that time that were made available. In 1948, Holmstrom described a "Univac machine" that is capable of text references searching that were associated with a subject code (Forsythe, n.d.). The code, along with the text, was stored on a magnetic tape and could be processed by the machine at the rate of 120 words per minute, according to Holmstrom. This appears as the first reference to a machine, specifically a computer that was used to search for content stored in a data store (Forsythe, n.d.).

Modern Application of Information Retrieval

The application of information retrieval on the turn of the century might not be all that different from how often it was used then. According to Tan, "In the modern era, where huge amount of information in multiple domains are made available, the use of information retrieval has been applied many times such as *text information retrieval* and *multimedia information retrieval"* (Tan et. al., n.d.). He defined text information retrieval as, "perhaps the most common and well known applications of IR. It deals with the retrieval of text documents from the internet, which is the database or can be other data stores as well", (Tan et. al., n.d.). As the web is swiftly becoming the medium of communications in business and academic information, it is essential and a-must to be able to locate the appropriate documents from the vast array of information. Such examples of text information retrieval systems include *Isoquest NetOwl* and *EUROSPIDER*. He defined multimedia information retrieval as, "on the other hand, deals with multimedia data such

as video and voice files. In addition to textual IR, other forms of representations should be used in order to accurately describe an object. The system should be able to interact with the user so as to recursively refine the query through machine learning and natural language processing". Such retrieval system is the *PArallel Multimedia Information Retrieval* (PAMIR) which performs context-based searching on images stored in a database containing almost 2100 images in all (Tan et. al., n.d.).

C. Lesk Algorithm

In 1986, Michael E. Lesk introduced an algorithm for dealing with word sense disambiguation called *Lesk Algorithm*, which has then been considered a classical algorithm. This algorithm is based on the assumption that words that were provided in a given block of text will tend to share a common general topic (WSD, 2015). A simplified version of the *Lesk algorithm* is produced to compare the dictionary definition of an ambiguous word with the terms contained in its neighborhood. Versions have been adapted to use WordNet (WSD, 2015). Figure II-3 shows the pseudocode of the *Lesk algorithm.*

For example, given an ambiguous word and the context in which the word occurs, the *Lesk algorithm* returns a Synset (synonyms set) with the highest number of overlapping words between the context sentence and different definitions from each Synset.

Unfortunately, this method of word disambiguating is very sensitive to the exact wording of definitions, so the absence of a certain word can radically change the results of the query. In addition, Lesk's algorithm determines overlaps only among the glosses of the senses being considered. This is a significant limitation in that dictionary glosses tend to be fairly short and do not provide sufficient vocabulary to relate fine-grained sense distinctions (WSD, 2015).

Currently, more works appeared and are underway which offer different modifications of this algorithm. These works use other resources for analysis such as thesauruses, synonym dictionaries or morphological and syntactic models; For instance, it may use such information as synonyms, different derivatives, or words from definitions of words from definitions (WSD, 2015).

```
let best_score_till_now = 0
loop until all candidate combinations are done
        let w[i...N] = get_next_candidate_combination(w)
        let combination_score = 0
        for i ranging over 1 <= i < N
                for j ranging over i < j <= N
                        for r1 ranging over (self hypernym hyponym holonym meronym troponym attribute)
                                for r2 ranging over (self hypernym hyponym holonym meronym troponym attribute)
                                        let combination_score = combination_score + get_score(gloss(r1(w[i])), gloss(r2(w[j])));
                                end for
                        end for
                end for
        end for
                if combination_score > best_score_till_now
                        let best_score_till_now = combination_score
                        let best_candidate_till_now[1...N] = w[1...n]
                end if
        end loop
        output best_candidate_till_now
```

Figure II-3. *Lesk Algorithm* Pseudocode
(source: *Adapting the Lesk Algorithm for Word Sense Disambiguation to WordNet* by Satanjeev Banerjee)

D. TF-IDF (Term Frequency-Inverse Document Frequency)

In information retrieval, TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. This statistic is often used as a weighting factor in information retrieval, text mining, and user modeling. Its value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Up to today, TF-IDF is one of the most well-known term-weighting schemes.

Variations of the TF-IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. It can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

In Equation 2.1, based on the Apache Lucene Documentation, *TF* correlates to the term's frequency, defined as the number of times the term appears in the currently scored document.

$$TF = \sqrt{termFreq} \qquad (2.1)$$

In Equation 2.2, *IDF*'s value correlates to the inverse of the number of documents in which the term appears. It means that rarer items give higher contribution to the total score. *IDF* appears for both the query and the document, therefore it is squared in this equation.

$$IDF = 1 + \log\left(\frac{number\ of\ documents}{docFreq + 1}\right) \qquad (2.2)$$

In Equation 2.3, the values of *TF* and *IDF* are multiplied with each other to complete the *TF-IDF* value.

$$TF - IDF = TF * IDF \qquad (2.3)$$

```
function TERM-FREQUENCY (term, document) return tf
        termInDoc : Number of times term t appears in a document
        docSize : Total number of terms in the document
        termInDoc <- 0
        for each word in the document do
                if word is equal to term
                        then termInDoc++
        tf <- count/docSize
        return tf

function INVERSE-DOCUMENT-FREQUENCY (term, documents) return idf
        docWithTerm : Number of documents with term t in it
        numDocs : Total number of documents
        docWithTerm <- 0
        for each documents do
                for each word in the document do
                        if word is equal to term
                                then docWithTerm++
                                break
        idf <- log (numDocs/docWithTerm)
        return idf

function TERM-FREQUENCY-INVERSE-DOCUMENT-FREQUENCY (document, documents, term) return tfidf
        return TERM-FREQUENCY (term, document) + INVERSE-DOCUMENT-FREQUENCY (term, documents)
```

Figure II-4. TF-IDF Pseudocode
(Source: http://onlinecourses.science.psu.edu/stat857/node/3)

E. Similarity Function

The similarity function is one of the most crucial algorithms used in the implemented study. As defined by Pennsylvania State University, "the similarity functions, also called *similarity or distance measures*, are essential to solve many pattern recognition problems such as classification and clustering". "Different measures are available in researches and databases to compare two different data distributions. As the names suggest, a similarity measures how close two distributions are", they added (Pennsylvania State

University, 2014). Similarity measure is a numerical measure of how alike two data objects are and often uses 1 and 0 to indicate whether a complete similarity or no similarity at all.

According to Ashby, “The concept of similarity has a fundamental importance to almost every scientific field. Topological methods are applied in fields such as semantics. Graph theory is widely used for assessing cladistic similarities in taxonomy”. “Fuzzy set theory has also developed its own measures of similarity, which find application in areas such as management, medicine and meteorology. An important problem in molecular biology is to measure the sequence similarity of pairs of proteins”, she added (Ashby et. al., 2007).

As stated by Ashby, similarity is of utmost importance to all scientific fields. So much so as she wrote that, “listing down all the uses of similarity is impossible. Similarity is a core element in achieving an understanding of variables that motivate behavior and mediate affect. The use of the similarity function in semantic searching is limitless and is now the basis on developing web search engines globally” (Ashby et. al., 2007).

F. Cosine Similarity Measure

As stated by Hong and Kim, “In the traditional Vector Space Model, documents are represented as term vectors and their similarity is generally measured by calculating the cosine similarity.” The study used the *Cosine Similarity Measure* to define the similarity function between two term-by-concept matrices of documents.

$$\mathrm{sim}(\mathbb{d}, \mathbb{q}) = \frac{<\mathbb{d},\mathbb{q}>_F}{\|\mathbb{d}\|_2 \cdot \|\mathbb{q}\|_2} \qquad (2.4)$$

In Equation 2.4, *d* is equal to the term-by-concept matrix while *q* represents the query matrix. In Equation 2.5 <*d*, *q*> denotes the *Frobenius product*, which is equal to the trace of the matrices *d* and *q*.

$$< \mathbb{d}, \mathbb{q} >_F = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbb{d}_{ij} \cdot \mathbb{q}_{ij} \quad (2.5)$$

In Equation 2.4, $\|d\|_2$ denoted the L2-norm of matrix *d*, is computed as shown below.

$$\|\mathbb{d}\|_2 = \sqrt{< \mathbb{d}, \mathbb{d} >_F} \quad (2.6)$$

```
double cosineSimilarity (array1: double type double array, array2: double type double array)
        let frob = frobProd(array1, array2)
        let l2norm1 = SquareRoot(frobProd(array1,array1))
        let l2norm2 = SquareRoot(frobProd(array2,array2))

        return frob/ l2norm1 * l2norm2
```

Figure II-5. Cosine Function Pseudocode
(Source: http://onlinecourses.science.psu.edu/stat857/node/3)

```
double frobProd (array1: double type double array, array2: double type double array)
        let result = 0
        for i = 0 until array1.rowLength, i++
                for j = 0 until array1.colLength, j++
                        result = result + (array1[i][j] * array2[i][j])
        return result
```

Figure II-6. *Frobenius Product* Pseudocode
(Source: http://onlinecourses.science.psu.edu/stat857/node/3)

G. Existing Studies and Applications of Semantic Search Engine

Semantic search engines have evolved from the dawn of the World Wide Web up to the present times. Along with this, different methods of information retrieval have been continuously developed and studied through research and studies of different experts, academes, and institutions. Some examples of existing studies on semantic search are explained and tackled in the succeeding paragraphs.

As stated from the research of Mulles et al., entitled "What You Search is What You Get: A Query-Based Automatic Document Searching Using Semantic Search", semantic search produces particular answer to various user queries by optimizing the availability of semantics present in the information given. Mulles expounded, "This research presents a search engine that is able to generate list of suggested pediatric medicine related documents faster than most of the semantic search engines present today. It also takes advantage of user familiarity through the simplicity and ease of use of the system. The system sets to hide the complexity of the semantic search from end users effectively" (Mulles et al., 2011).

In contrast with existing keyword-based semantic search engines which typically functions as that of the common search engines present in the market nowadays, this research aims to bring the level of understanding deeper into the semantic analysis part. Further, the research provides comprehensive means to produce precise answers that is evaluated using the evaluation tool called precision, recall and f - measure to further validate the results shown during the implementation phase of the system (Mulles et al., 2011).

Similarities of this existing study to the researchers' study is the use of user query to produce the list of documents through semantic searching, the use of precision and recall as evaluation tools, and the use of Information Retrieval (IR) methods while differences include the research domain and different model and method applications on IR.

As per the research study of Libut et. al., entitled "*Text-Based Project Document File (PDF) Search Engine Using XRank Algorithm*" takes a reference of the previous study "What You Search Is What You Get: A Query - Based Automatic Document Searching Using Semantic Search" by Mulles et. al., using XRank algorithm as a sorting/ranking algorithm. Libut stated that their study "wants to surpass the previous study's relevance and accuracy in output generation. This will help find the most suitable and relatable documents for your input query and become more reliable" (Libut, 2013). Similarities and differences of this study is the same as in Mulles et. al.'s existing study.

Based on the study of Hong et. al. entitled, "*A semantic search technique with Wikipedia-based text representation model*", semantic search is known as a series of activities and techniques to improve the search accuracy by clearly understanding users' search intent. Usually, a semantic search engine requires ontology and semantic metadata to analyze user queries. Hong added that, "However, building a particular ontology and semantic metadata intended for large amounts of data is a very time-consuming and costly task. In order to resolve this problem, this research paper implements a novel semantic search method that does not require ontologies and semantic metadata by taking advantage of semantically enriched text model. Through extensive experiments using the OSHUMED document collection and SCOPUS library data, the proposed method improves users' search satisfaction" (Hong et al., 2016). Similarities between this study and the researchers'

study are the use of the 3rd-order tensor model and the vector space model. Differences include the search algorithms running in the concept space, therefor having unrelated concepts to the user query. The researchers' study develops an approach where the search algorithm uses a model that consists of concepts and documents in each of them that are highly related to the query, therefore improving the search capability of the model. The researchers' study also improves the time complexity where a query can extract models that were previously built by the system if the query is related to that model.

The study of Ma et. al., entitled "*Material Hub: A Semantic Search Engine with Rule Reasoning*" tackles that, "some special domains, if search engines only take advantage of keywords, the searchers need to matter lots of professional terms". "Moreover, to refine the search interests, the professional rules and reasoning of those rules should be used. To resolving above problems, this paper focuses on extending search engines via ontology extraction and modeling, semantic rules reasoning algorithm, and feature-based ordering algorithm and related semantic technologies.", Ma added. A semantic search engine *Material Hub* has been set up based on above research points and applied in industry. Four experiments are conducted which reveal the improvements of the accuracy and efficiency for retrieval as well as semantic rules reasoning algorithm satisfyingly contribute to an improvement in the responding time compared to other semantic reasoning algorithms (Ma et. al., 2012).

Based on the research study of Yadav et. al., entitled "*A novel approach for precise search results retrieval based on semantic web technologies*", they discussed that "Web documents are growing exponentially and the results retrieved by traditional search engines are marked by irrelevant and inconsistent results. In order to alleviate this problem and

increase degree of relevance, there is need to move towards development of semantic search engines". Yadav explained further, "such engines produce results by focusing on meaning of context rather than structure of content. This paper describes the implemented architecture of semantic search engine that takes web search results as input and refines them with notion of semantics. The output produced would be precise and relevant results in context of user's query" (Yadav et. al., 2016).

H. Synthesis

Improvements and developments in the field of Information Retrieval and Semantic Search provide users more freedom and accuracy when searching for documents they need. While improving on some limitations and hurdles on these fields, recent studies also uncover more hurdles and limitations that future researchers need to address such as the lack of a more powerful and efficient sorting and searching algorithm for Information Retrieval and improvement of the concept window for the user query. With this, machine learning and intelligence is measured by the ability of the system to adjust its functionalities and behavior in favor of the user's behavior and personal preferences.

Semantic search is a technique to search information that is beyond the keywords and seeks to understand the semantics of the search query. It has been applied to modern search engine and sites many times, because of its accuracy. It often makes use of a semantic technology called "natural language processing" in order to understand what the user really want. Information retrieval is used in semantic search in order to retrieve relevant information or documents stored in a database depending on the user query.

The user query undergoes a process to construct a concept in which the query belongs. The concept is then compared to the concepts of the previous queries already

made and recorded in the system. A new 3rd-order tensor model pool is constructed if the current does not match requirements for models in the cache. Derived concept space and concept window are needed to construct a 3rd-order tensor model. Similarity measure is used to construct the derived concept space and *Lesk algorithm* for the concept window. Each term in the concept window is compared to each of the concepts in the derived concept space using the TFIDF. Afterwards, the 3rd-order tensor model is used to find the number of relevant documents related to the query among the concepts in the model.

The following table, Table II-1, shows the summary of the following studies that were used to support the researchers study.

Table II-1. Summary of Existing Studies

Title	Author, Year	Approach	Efficiency
What You Search Is What You Get: A Query-Based Automatic Document Searching Using Semantic Search	(Mulles et. al., 2011)	Conceptual graph matching algorithm, applied in a semantic search engine and breadth-first search as the searching strategy.	The methods used were successful in retrieval of information, as the use of precision, recall and f - measure as evaluation tools yield remarkable results. The computed averages of these three tools are 90%, 83.6% and 84.64% respectively.
Text-Based Project Document File (PDF) Search Engine Using XRank Algorithm	(Libut et. al., 2013)	XRank Algorithm as the sorting/ranking algorithm, improved from "What You Search Is What You Get".	The current system has been an improvement to the previous study, fulfilling its problems. In addition, the use of precision, recall and f-measure as the study's evaluation tools gave it remarkable results. The computed averages of the precision are 90%,

			recall is 90% and f-measure is 90% respectively.
A semantic search technique with Wikipedia-based text representation model	(Hong et. al., 2016)	Use of ontology and sematic metadata to analyze search queries and the use of 3rd-order tensor text representation model.	Research study proved that the proposed method improves users' search satisfaction reasonably. The implemented method improved the baseline method by about 2~15% in terms of the nDCG (Normalized discounted cumulative gain) as a measure of ranking performance.
Material Hub: A Semantic Search Engine with Rule Reasoning	(Ma et. al., 2012)	Focused on extending search engines via ontology extraction and modeling, semantic rules reasoning algorithm, and feature-based ordering algorithm and related semantic technologies.	In the experiment, they compared Material Hub with keywords based search engine Sphider and semantic search engine without rule reasoning. collect about 2G product data, and build 20 m index, in the knowledge base there 1875 concepts, 16 ontology files,133 semantic rules. And we execute 1000 times query with parameter restrictions and calculate the accuracy and recall rate. Material Hub is better than others with a retrieval accuracy of 90% and recall rate of 89%.

A novel approach for precise search results retrieval based on semantic web technologies	(Yadav et. al., 2016)	The use of semantic web, WordNET, and JADE ontology methods such as RDF model and SSE framework.	The paper describes proposed architecture of SSE which incorporates GOOGLE search results as input and enhances them with the help of SW technologies. Modules like query processing, import existing ontology, extraction of knowledge and many more have been introduced in proposed framework. At last, relation based ranking algorithm is being applied to compare query and document graphs. It leads to enhanced ranked results that are presented to user.

Chapter III Research Design and Methodology

This chapter includes the research design that is implemented by the researchers for the study. These include the research instruments, sources of data, and the statistical treatment of data. The hypothesis of the study is presented.

A. Hypothesis

The researchers believe that the introduction of similarity function on the concept vector of the 3rd-order tensor model can further improve the search time taken by the computation of the search algorithm from the previous model. If it is deemed possible, a search engine can accurately search and gather documents through information retrieval with lesser time whilst not affecting the accuracy and precision of the current model.

If these assumptions are proven, a program can be developed to use as an alternative search engine model implementing these functionalities. Since queries are compared to saved queries, it takes *O(n x m)*, where ***m*** is the running time of the cosine similarity for two matrices and **n** is the number of query matrix in the result set pool, to extract the 3rd-order tensor model of that stored query which was stored in cache. If there are no matching queries in the cache with the process of using TFIDF and Sim measure applied to the 3rd-order tensor model, the researchers should gather more relevant documents and rank the files based on their relevance to the user query. If these assumptions are proven, the user can garner a more tailored list of relevant documents that have been filtered out to be based on their query.

H_0: The use of an algorithm that removes unwanted concepts in the 3rd-order tensor model for information retrieval has no significant difference on accuracy and running time between the existing and the new approach.

H_1: The use of an algorithm that removes unwanted concepts in the 3rd-order tensor model for information retrieval has significant difference on accuracy and running time between the existing and the new approach.

B. Research Methods

The main method that the study uses is the experimental or scientific method. This is used to prove whether or not the application of the similarity measure on the concepts vector does lessen the time it takes for the search algorithm to find relevant documents from each of the document sets. As such, a program is developed to test and apply this assumption.

C. Research Design

There are 5 steps that make up this research project:

1. Researching about the basics of semantic search and information retrieval, as well as existing studies regarding semantic search engines. Find a model search engine and find ways to improve it.
2. Developing new approaches which can improve the accuracy and speed of this existing semantic search engine, therefore enhancing its overall performance.
3. Programming/Coding the new implemented approaches and implementing it into the existing architecture in order to build a new implemented system.
4. Produce adequate test cases to examine the new implemented system.
5. Analysis of the results on whether there is improvement between the model system and the new system.

In the first step, the researchers conduct investigation and research works regarding the concepts involved such as semantic search and information retrieval. Knowledge on past researches were also gathered and different approaches were taken into consideration in this phase. In the second step, the researchers converge all of the gathered information and find the pros and cons of these models. If improvement on a model is detected, a new approach is considered for that system. This new approach is then made into a pseudocode, ready for programming phase. In the third step, the approach's pseudocode is converted into a program, as well as the user interface which end users will use. In the fourth step, multiple test cases that cover several scenarios of input into the system are conducted to test the performance of the new approach that is being introduced. In the fifth and final step, the results are analyzed and weighed using the evaluation tools (precision, recall, and F1-measure) in order to know whether the system fares better than the existing system.

D. Research Instruments

Research Instruments

The researchers used a laptop with the following specifications for this study:

- Acer Aspire ES-15
- Intel Core i3-6006U @ 2.00GHz
- 4GB RAM @ 1063MHz
- Windows 10 Home Single Language 64-bit

The researchers used the following software in this study:

- Java 8
- Eclipse Oxygen ver. 3.7.1

- XAMPP ver. 7.1.11
- WordNet ver. 2.1

All software used for the study are free to get from the Internet.

Sources of Data

The researchers gathered the majority of documents and data sets from the IEEE Online Database and Wikipedia.

E. Data Gathering and Statistical Treatment of Data

The training data in this study is gathered using random sampling. PDF documents are randomly gathered from a database into the data store which the system uses. But the documents are limited on technology related documents specific to Word Disambiguation, Information Extraction, Image Processing, Data Compression, and Semantic Search that are only written in English. The database that this study uses to gather information to store into the data store is the IEEE Online Database. Sample documents can be seen on Appendix C. The total number of documents needed from the IEEE data repository related to technology specific to Word Disambiguation, Information Extraction, Image Processing, Data Compression, and Semantic Search is 250 and these were retrieved from November 13, 2017 to November 20, 2017. The IEEE data repository has an advance search where different categories are presented. The researchers then get the total number of documents related to the categories.

The total number of documents needed from Wikipedia is approximately 15,000 and these were retrieved from September 11, 2017 to November 9, 2017.

The researchers created two sets of queries, namely, Set A, and Set B. Each set is divided into five categories, namely; Data Compression, Image Processing, Information

Extraction, Semantic Search, and Word Disambiguation. The categories contain five queries each. Each category is described as follows: 1) the first query is the definition of each particular category, 2) the second query contains a query that is highly related to the category, 3) the third query contains a query that is good or related to the category, 4) the fourth query comprise of a query that is somewhat related to the category, and 5) the fifth query contains a query that is not related to the category. There are two sets used to test the data, namely; Set A is used to compare the actual running time of the implemented system with the sorted concepts and pruning feature compared to the original system, and Set B is used to compare the actual running time between the implemented system with result pool and without the result pool.

To determine if the accuracy of the implemented system is maintained, the researchers used same Set A queries to both the original and the implemented system. The accuracy of the implemented system is then compared to the accuracy of the original system. As for Set B, the queries are used to the implemented system and the accuracy is then compared to the accuracy when using the result set of Set A.

On identifying whether a document is relevant to the query or not, the researchers used the Apache Lucene search engine. Each query is used in the Apache Lucene search engine and the resulting result set are then used as the set of relevant document for that particular query.

For the evaluation on the accuracy of the results given from the system, the researchers use the precision, recall and F – measure. Precision measurement is the measure of the percentage of retrieved documents that are relevant. Recall is the percentage of

relevant documents retrieved. F1 – measure is a measure that combines precision and recall using the harmonic mean of precision and recall.

In measuring the precision and recall, the number of True Positives, False Positives and False Negatives are needed to be known. The True Positives are the set of documents that are correctly retrieved from the set of documents which are described to be "correct". The False Positives are the set of documents which are not supposed to be retrieved but retrieve. The False Negatives are the set of documents which are supposed to be retrieved but are not returned.

In Equation 3.1 for the precision measurement, T_p is the number of true positives and F_p is the number of false positives.

$$\boldsymbol{P = \frac{T_p}{T_p + F_p}} \qquad (3.1)$$

In Equation 3.2 for the recall measurement, F_n is the number of false negatives.

$$\boldsymbol{R = \frac{T_p}{T_p + F_n}} \qquad (3.2)$$

In Equation 3.3, the F1-measure is computed using the precision and recall values, computed using Equations 3.1 and 3.2 respectively.

$$\boldsymbol{F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}} \qquad (3.3)$$

Significance testing is conducted on the results of the actual running times on both Set A (Unsorted and Sorted) and Set B (With and Without result pool). The researchers used T-test in testing the hypothesis. A T-test is an analysis of two population means through the use of statistical examination. A T-test consisting of two samples is commonly used along with small sample sizes, preferably less than or equal to 30, and testing the difference between the samples when the variances of two normal distributions are

unknown. The T-test looks at the T-statistic; which is the test statistic; the t-distribution, and degrees of freedom in order to determine the probability of difference between populations. The researchers utilized the t-test since the sample size of the data is below 30, which is specifically 25, and the population standard deviation is unknown. The researchers used a specific T-test, which is Paired T-test, in the significance testing. A paired t-test is used to compare two population means where there would be two samples in which observations in one sample can be paired with observations in the other sample. During a paired t-test, it is assumed that the data are continuous and not discrete, the data (i.e. the differences for matched-pairs) are following a normal probability distribution, and that the sample of pairs is a simple random sample from its population. Each individual part of the population should have an equal probability of being selected in the sample. The researchers determined that the test is going to be a one-tailed test for both Set A and Set B, since the researchers would like to test whether the values indicate a more effective system.

The researchers determined the level of confidence that is most fitted for the significance testing. The level of confidence shows the sureness and certainty of the researchers in their claim in the alternative hypothesis. It is expressed as a percentage and represents how often the true percentage of the population who would pick an answer lies within the confidence interval. The 95% confidence level means you can be 95% certain. In this significance testing, the researchers selected a 95% confidence level as the level of confidence. As a result of this, the alpha level is determined. Alpha value specifies a value for the probability of a Type-I error. A TypeI error occurs when a true null hypothesis is rejected. Values considered for the alpha level must be between zero and one. Historically,

the value of 0.05 has been used for alpha. This means that about one test in 20 will falsely reject the null hypothesis. The value for alpha should be picked wherein that value represents the risk of a Type-I error the researchers are willing to take in the experimental situation. Possible range of values for the alpha can be considered, such as 0.01 0.05 0.10 or 0.01 to 0.10 by 0.01. The researchers used 0.05 because of the confidence level being 95%, which resulted in as computation 100% - 95% = 5% or 0.05.

The researchers utilized Microsoft Excel 2016 for the computation of the Paired T-Test thorough the Data Analysis option. The data for the computation of the Paired Two Sample for Means of Set A (Unsorted and Sorted) that are inputted are the actual running time (in seconds) of Set A, the hypothesized mean difference; which is set to 0 since the null hypothesis states that there is no significant difference between results and that the difference between population means is 0; and the alpha value of 0.05. The same process would be done for Set B (With and Without result set pool). The results for both Set A and B can be found in Chapter IV, in the section on Test Results. The researchers also computed for the mean difference, standard deviation of the difference, standard error of the difference, and the lower and upper confidence levels.

The mean difference is the average of the difference between each pairs of values, X_i and Y_i, of the two sample sets, X and Y.

In Equation 3.4, the standard deviation, S_x, of difference is computed by the standard deviation formula, wherein n is the number of data points, $\bar{X}$ is the mean of the X_i, and X_i are each of the values of the data.

$$Sx = \sqrt{\frac{\sum_{i=1}^{n}(Xi - \bar{X})^2}{n-1}} \quad (3.4)$$

In Equation 3.5, the standard error of difference is computed using the formula, wherein S_x is the standard deviation of difference, and *n* is the number of data points

$$std\ error = \frac{Sx}{\sqrt{n}} \quad (3.5)$$

In Equations 3.6 and 3.7, the upper and lower confidence levels are computed using the formulas, wherein *md* is the mean difference, *std error* is the standard error of difference, and α is the alpha value resulting from the t-test from Data Analysis option.

$$upper = (md + std\ error) * \alpha \quad (3.6)$$

$$lower = (md - std\ error) * \alpha \quad (3.7)$$

Chapter IV Presentation and Analysis of Data

This chapter includes the system architecture of the project for the study. This includes the main modules and submodules about the different functions and algorithms, including equations and computation procedures used on the program. The methods of data gathering, a prototype of the program, and summary of results and findings are also contained in this chapter.

A. System Architecture

The system architecture for the project is divided into two (2) phases: the creation phase and search phase. The creation phase is the phase that creates resources for the search phase depending on the documents and Wikipedia pages as inputs of the phase. This phase creates the resources only once. However, if changes are applied to the documents or the Wikipedia pages, then it is necessary to run this phase again before using the search phase. The search phase is the phase where it uses the query from the user and the resources extracted from the "Conversion of Text Files into Objects" module to generate the result set and present it back to the user. The "Conversion of Text Files into Objects" module is a module that is run once to prepare the resources needed for the search phase. However, due to the web application technology in Java, the first request to the web application runs the "Conversion of Text Files into Objects" module. In other words, the first request in the web application has an overhead of extracting resources generated from the creation phase. Figure IV-1 shows the system architecture of creation phase and search phase.

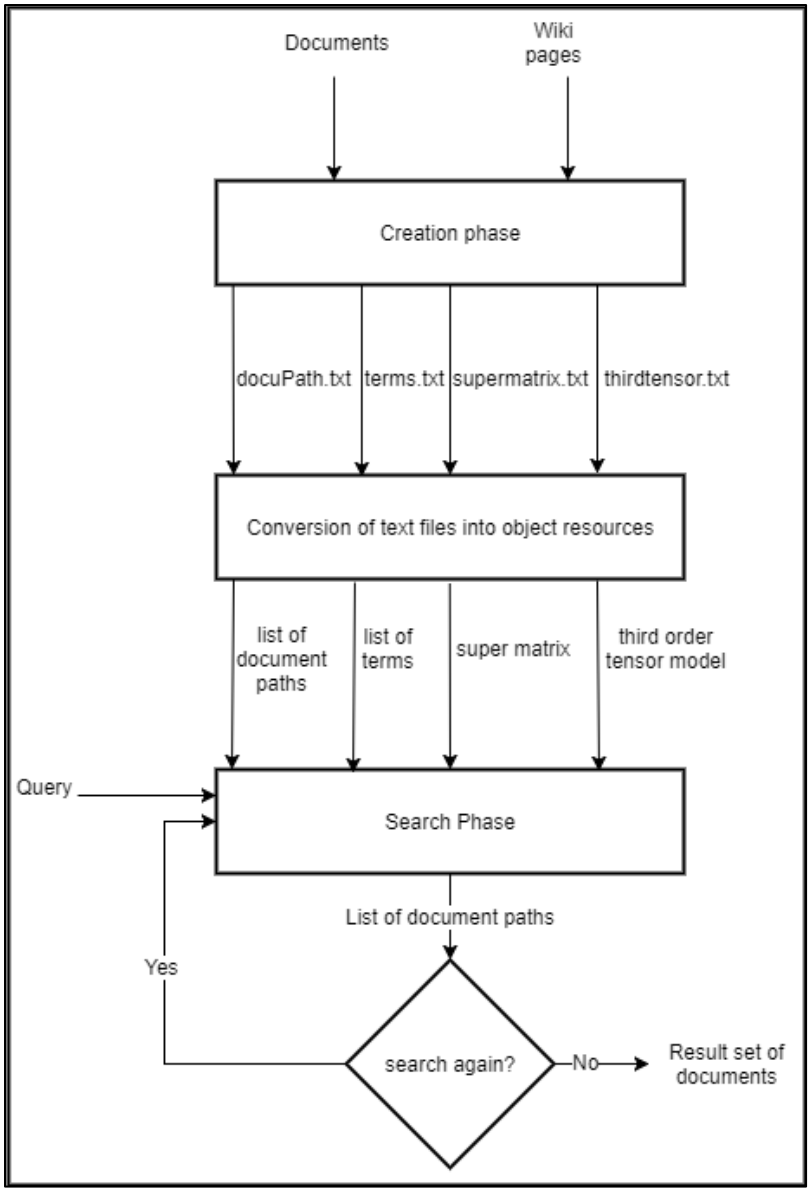


Figure IV-1. System Architecture of the System

A detailed system architecture for the creation phase is presented in Figure IV-2. Modules shown in the figure that are grouped using dashed bordered boxes are the new implemented features added in the system and modules that are not grouped with dash bordered boxes are the features/modules introduced by Ki-Joo Hong and Han-Joon Kim. The creation phase contains 7 modules, which starts with the “Generate Concept Space” module that accepts documents and Wikipedia pages as inputs.

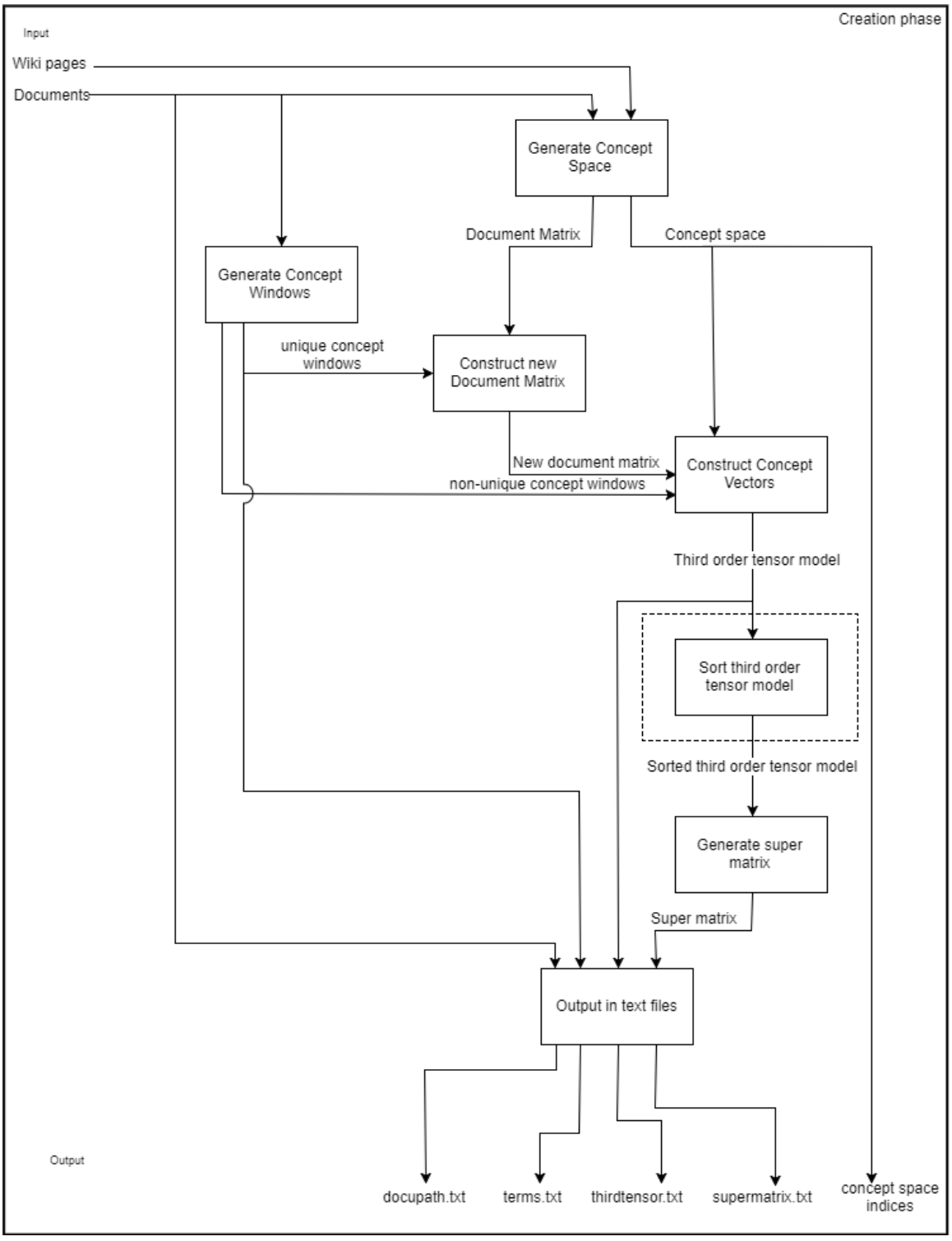


Figure IV-2. Level-2 System Architecture of the System (Creation Phase)

This module produces Document Matrix and Concept Space to be used as required inputs in Concept Vector module. It also produces Concept Space Indices where it is used as the resource inputs for the search phase. After the "Generate Concept Space" module, the "Generate Concept Windows" module runs and accepts documents as the input of the module. This module outputs a list of non-unique Concept Windows and a list of unique Concept Windows. These outputs are used for the "Construct New Document Matrix" module and the "Construct Concept Vectors" module. After the "Generate Concept Windows" module, the Document Matrix from the "Generate Concept Space" module is transformed into a new Document Matrix using the list of unique Concept Windows produced by the "Generate Concept Windows" module. The new Document Matrix, list of non-unique Concept Windows, and Concept Space are used in "Construct Concept Vectors" module. This module constructs the 3^{rd} – order tensor model and this output is passed to the "Sort Third Order Tensor Model" module.

The "Sort Third Order Tensor Model" sorts the input and output the sorted 3^{rd} – order tensor model. This output is used in the "Generate Super Matrix" module to produce the Super Matrix. The documents, list of unique Concept Windows, Sorted 3^{rd} – order tensor model, and Super Matrix are passed to the "Output in Text Files" module at the end of the architecture. The resources used for the search phase are the text files produced by the "Output in Text Files" module and the Concept Space Indices produced by the "Generate Concept Space" module.

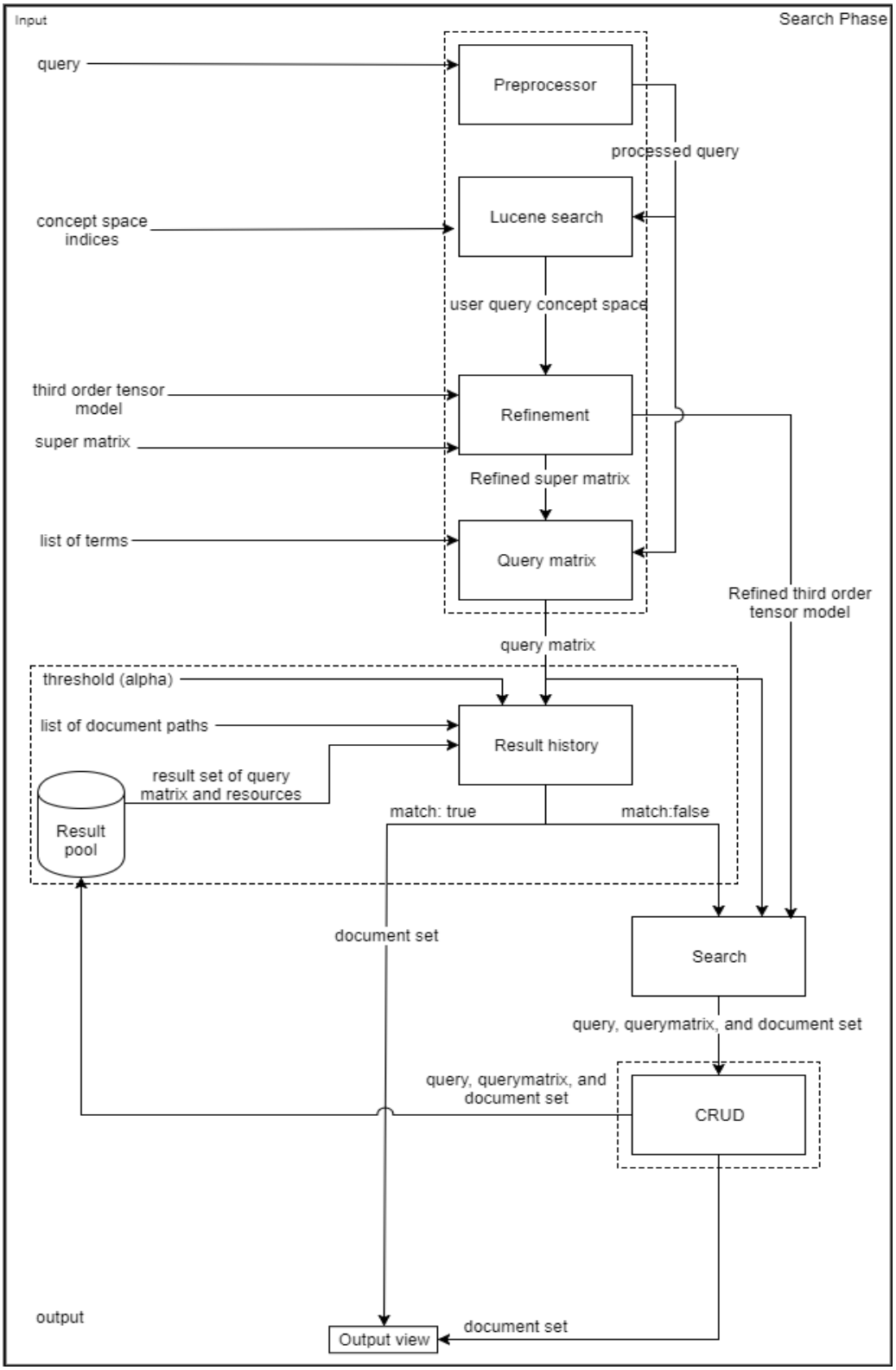


Figure IV-3. Level-2 System Architecture of the System (Search Phase)

Figure IV-3 shows the detailed System Architecture of the search phase. The search phase accepts queries from the user and preprocesses it. After that, it passes the preprocessed query to the Lucene Search module where it produces a list of concepts related to the query. This list of concepts of the user query concept space is used to refine the 3rd – order tensor model and Super Matrix by using the Refinement module. The outputs of the Refinement module are the Refined 3rd – order tensor model and the Refined Super Matrix. The Refined 3rd - order tensor model is used later in the Search module if the result history module does not find a query similar to the current query being searched.

The Refined Super Matrix and the preprocessed query are passed to the Query Matrix module and the output of this module is a Query Matrix for the current query. This Query Matrix and the threshold is used in the Result History module to search similar query in the result set pool and decide whether to select a record in the pool or search through the tensor model.

If a match was found in the pool, then the result set of document of the matched query is passed to the output view where it will display each documents. However, if a match was not found, then the Refined 3rd - order tensor model, produced by the Refinement module, is used along with the Query Matrix, constructed from the Query Matrix module, in the Search module.

The result set of documents searched by the Search module is stored in the result set pool along with the current query and the Query Matrix. After storing the resources in the result set pool, the result set of documents is passed to the output view to present each documents to the user.

B. Main Modules

1. Generate Concept Space

This module prepares the document matrix and the concept space by using the documents and wiki pages as inputs for the process.

Input:

A folder path to the list of documents, and a folder path to the list of Wikipedia pages.

Process:

1. For each document, the TF-IDF value is computed for every term using the TF-IDF algorithm
2. Then the heavy weighted terms or the top 20% of the terms is used as a query for Apache Lucene search engine
3. The result set which contains a list of concepts is then stored in a data structure that contains all the result set of each document
4. After getting all the result set for each document, the DCF is then computed for each concept
5. DCF or document concept frequency is the frequency of a particular concept in the entire set of documents.
6. After computing the DCF for each concept, the concepts are sorted from highest to lowest according to their DCF
7. The top 200 concepts according to their DCF value are then selected as the final concept space for a particular document corpus
8. The Concept Space is then indexed using Apache Lucene

Output:

A concept space and a document matrix with a data structure of a hash map where the keys are the document ID and the values are array list of terms and TF-IDF values.

2. Generate Concept Windows

This module constructs list of unique concept windows and list of non-unique concept windows where these output are used for the construction of the new document matrix and the construction of 3rd-order tensor model.

Input:

A list of document paths with a data structure of array list of strings.

Process:

1. Extraction of documents – In this process, each documents are extracted using the document path. These documents are arranged and collated to form a list of documents with a data structure of an array list consisting string values.
2. For each document in the list of documents, the document selected is split or tokenized into sentences. After the splitting process, the sentences are grouped.
3. The grouping is done by having a maximum of 2 succeeding sentences and a maximum of 2 preceding sentences of each sentences tokenized from the document. Thus, forming a list of windows where each window holds the current sentence and its succeeding and preceding sentences.
4. For each window in the list of windows, the sentence is extracted from the window and tokenized into words and a selection process is done for each word.

5. The selection process is divided into two process. First, the succeeding and proceeding sentences are used to find synonym words for a selected word or target term. These synonyms are stored in an array called the concept window. The concept window has a size of 2n + 1 and it is limited by a maximum of 9. This is because a concept window with a size of 9 gives the peak performance and any size more than 9 gives less performance than with the size of 9 (2002, Banarjee).
6. The second process is done if the size of the concept window produced from the first phase is not enough and if there are available nearby terms from the target term's sentence. This process selects nearby terms from the target term's sentence and fills up the concept window.
7. After filling up the concept window of a target term or selected word, the concept window is stored or added in a list data structure

Output:

A list of concept windows with a data structure of array list that contains concept window. Concept window is a data structure that contains attributes like target term and terms that filled up the concept window.

3. Construct New Document Matrix

This module transforms the document matrix produced from the 'Generate concept space' module into a new document matrix where its terms are synced with the target terms produced from the 'Generate concept window' module.

Input:

A hash map where its keys is an integer indicating the document number and its value is an array list that contains terms of the document and their TF-IDF value. A list of unique concept window where each concept window contains a target term.

Process:

1. The list of unique concept window is sorted using merge sorted algorithm. The comparison of values done in the merge sort are target terms contained in the concept windows.
2. For each concept window in the list of sorted unique concept window, the target term is extracted from the selected concept window and an array list of terms is extracted from the hash map for a given document. A selection process is done for the target term compared with the array list.
3. The selection process is done by comparing the target term to every term in the array list. If there is no match between the target term and a selected term, then a zero value is stored in the new document matrix and if there is a match, the TF-IDF value of the selected term is extracted and stored in the new document matrix.

Output:

A 2-D array, document matrix, that contains TF-IDF value selected from the hash map produced by the ‘Generate concept space’ module. The first element of the array indicates the term ID and the second element of the array indicates the document ID.

4. Construct Concept Vector

This module constructs 3rd-order tensor model by the use of the list of non-unique concept windows generated from the ‘Generate concept windows’ module and the new document matrix constructed from the ‘Construct new document matrix’ module.

Input:

A 2-D array, document matrix, that contains TF-IDF value of terms compared with their corresponding document, a list of non-unique concept windows where there exist a pair of concept window that has equal target terms, and a concept space where it contains concepts selected from the ‘Generate concept space’ module.

Process:

1. For each concept window in the list of non- unique concept windows, the terms that filled up the concept windows or, simply, the surrounding terms are extracted from the selected concept window and are compared to each of the concepts in the concept space using the TF-IDF algorithm.
2. The TF-IDF value of each terms in the surrounding term are averaged and stored in a 2-D array called the concept matrix.
3. After filling the concept matrix, duplicate target remains in the matrix so a process of summarization is done to the concept matrix.
4. The summarization process is done by searching a group of duplicate terms and selecting maximum values in each column of the group. Thus, summarizing the group into a 1-D array.

5. After the summarization process, a selected column from the document matrix added to the concept matrix. Thus, producing a new 2-D matrix or a term by concept matrix. Each row in this 2-D matrix is called the concept vector.
6. Collating these newly produced terms by concept matrix, creates the 3rd-order tensor model.

Output:

A 3-D array, 3rd-order tensor model, that contains added TF-IDF value from the document matrix and the concept matrix. The first element of the array indicates the document ID, the second indicates the term ID, and the third indicates the concept ID.

5. Sort 3rd - order Tensor Model

This module sorts the 3rd-order tensor model using the tensor model produced by the 'Construct concept vectors' module.

Input:

A 3-D array, 3rd-order tensor model, that contains values computed from the 'Generate concept vectors' module.

Process:

1. For each document in the 3rd-order tensor model, a 2-D matrix, term by concept matrix, is selected and a sorting process is done using this matrix.
2. A sorting process is done by arranging the sum of each column using merge sort sorting algorithm.

Output:

A sorted 3rd-order tensor model where the model is described using the 3-D array data structure.

6. Super Matrix Module

This module converts the 3rd-order tensor model into a 2nd order tensor model or a 2-D array.

Input:

A 3-D array, 3rd-order tensor model, where the first element is the document number, the second element is the term number, and the third number is the concept number.

Process:

For all slices of term by concept array in the 3-D array, their values are added and averaged. Thus, creating a new term by concept matrix containing average values.

Output:

A 2-D array, super matrix, where the first element is the term number and the second element is the concept number. The values in this are average values from the 3rd-order tensor model.

7. Preprocessor

This module preprocesses the query sent by the user.

Input:

A query from the user with a data type of a string.

Process:

1. Each terms of the query are processed by removing the stop words.
2. After that, each term is also processed by transforming them into base form. This is done by searching the base form in the WordNet dictionary.
3. If a term has too many base form, the original form is retained.

Output

A processed query with a data type of a string.

8. Lucene Search Module

This module searches a list of wiki pages using the query from the user and this feature is implemented using a feature from Apache Lucene as a black box.

9. Refinement Module

This module cuts or limits the 3rd-order tensor model and the super matrix.

Input:

A 3rd-order tensor model and a super matrix. An integer, alpha or threshold, that indicates the size of the concepts returned by the Lucene search.

Process:

1. Decides if the size returned by the Lucene search is greater than the size of concepts in the 3rd-order tensor model.
2. If the it is greater than the threshold, then there is no limit.
3. However, if it is less than the threshold, then there is a limit.

Output:

A size that indicates the limit of process of the proceeding modules.

10. Search Module

This module uses the 3rd-order tensor model and the query matrix to search the set of documents to be returned to the user.

Input:

A 3rd-order tensor model, an integer that indicates the limit of concepts, and a query matrix.

Process:

1. For each term by concept slices of the 3rd-order tensor model, a selected term by concept matrix is compared with the query matrix using the similarity measure.
2. After comparing each term by concept matrix, these matrices are ranked up based on their similarity value with the query matrix. The ranking is descending and the term by concept matrices that have zero similarity value with the query matrix are not included in the ranked list.

Output:

A set of documents that are ranked according to their similarity value with the query matrix.

11. Result History Module

This module searches in the result pool and retrieves a record if the highest similarity value in the pool meets the threshold or alpha value.

Input:

A query matrix with a data structure of 2-D double array.

Process:

1. Extraction of result set in the result pool.
2. For each record in the result set, a query matrix is extracted from a selected record. The extracted query matrix is compared with the input query matrix and their similarity value is stored in an array list.
3. This array list is sorted using the merge sort using the descending order.

4. Using the first element of the sorted list. If the similarity of the first element is greater than or equal to the alpha then the result set of documents of that record is extracted, if not then the flow of process proceeds to search module.

Output:

A result set of documents or null that indicates there was no match.

12. Database CRUD Module

This module stores the resources provided by the search phase if there was no match in the result history module.

Input:

A query from the user, a result set of documents, and a query matrix constructed from the query matrix module

Process:

1. The resources are collated into one object
2. Serialization of the object and the process of storing the serialized object in the result set pool

Output:

A confirmation message indicating a successful operation.

C. Test Data

The program used an estimated total of 15,000 Wikipedia pages in PDF format. They were gathered using a Wiki tool that utilizes the Mediawiki API in order to gather Wikipages that links to certain defined titles in the documentation and save them into the

local repository. A total of 250 IEEE documents in PDF format were also gathered through the IEEE Explore Online database portal.

For the testing of the program's accuracy, F1-measure, precision and recall methods are utilized. The formulas for the following methods were defined in Chapter III.

D. Sample System Simulation of Test Data

This section shows the simulation of these two phases, namely; creation phase and search phase. The simulation starts with the flow of the creation phase because creation phase has the resources that the search phase needs.

The creation phase starts with the 'Generate concept space' module. This module needs documents and wiki pages as inputs. Shown in Tables IV-1 and IV-2 are portion of the documents and wiki pages used to build the output of this module which are document matrix and concept space. Description on the content of the document and wiki pages are also listed in the tables.

The output of the "Generate Concept Space" is the concept space and document matrix. Figure IV-4 shows a sample list of concepts selected from the list of wiki pages in the dataset and Figure IV-5 shows a sample portion of the document matrix.

Table IV-1. Sample List of Documents in the Dataset

ID	Document title	Content	Image	Symbols	File format
1	Generating Rules with Common Knowledge A Framework for Sentence Information Extraction	Abstract	None	Yes	PDF
2	Medical image processing A review	Abstract	None	Yes	PDF
3	Lossless Data Compression via Substring Enumeration for k-th Order Markov Sources with a Finite Alphabet	Abstract	None	Yes	PDF
4	Semantic Search Meets the Web	Abstract	None	Yes	PDF
5	Improving Persian POS tagging using the maximum entropy model	Abstract	None	Yes	PDF

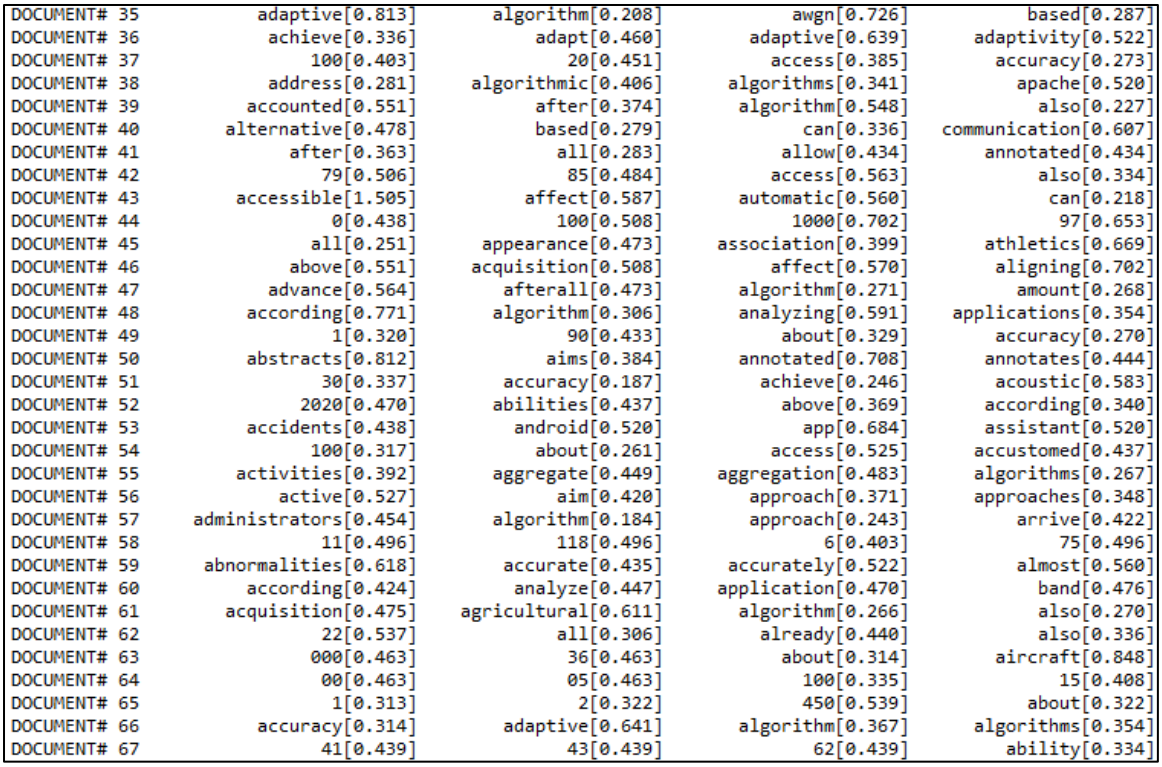
Table IV-2. Sample List of Wikipedia Pages in the Dataset

ID	Document title	Tables	Images	Symbols	File format
1	Pacificair	Yes	None	None	PDF
2	Bubunaoan River	None	None	None	PDF
3	Transparency (data_compression)	None	None	None	PDF
4	Occam's razor	None	Yes	None	PDF
5	Semantic Web	None	Yes	None	PDF

The format shown in the Figure IV-4 follows the URL format where the file directive to the file is shown first then the file name of the target file. In Figure IV-5, the values are presented by showing the terms in each document first followed by their TF-IDF value.

```
C:\Users\harji\Documents\DataSet\Concepts\Calligraphy.pdf
C:\Users\harji\Documents\DataSet\Concepts\A._P._J._Abdul_Kalam.pdf
C:\Users\harji\Documents\DataSet\Concepts\William_Salesbury.pdf
C:\Users\harji\Documents\DataSet\Concepts\Touch_(manga).pdf
C:\Users\harji\Documents\DataSet\Concepts\STAR_(student_association).pdf
C:\Users\harji\Documents\DataSet\Concepts\Pedro_Calungsod.pdf
C:\Users\harji\Documents\DataSet\Concepts\Oscillation.pdf
C:\Users\harji\Documents\DataSet\Concepts\Piwi-interacting_RNA.pdf
C:\Users\harji\Documents\DataSet\Concepts\Cairo.pdf
C:\Users\harji\Documents\DataSet\Concepts\Kabayan_Mummies.pdf
C:\Users\harji\Documents\DataSet\Concepts\Invercargill_Passenger_Transport.pdf
C:\Users\harji\Documents\DataSet\Concepts\Isaiah_Berlin.pdf
C:\Users\harji\Documents\DataSet\Concepts\Freidank.pdf
C:\Users\harji\Documents\DataSet\Concepts\Eternalism_(philosophy_of_time).pdf
```

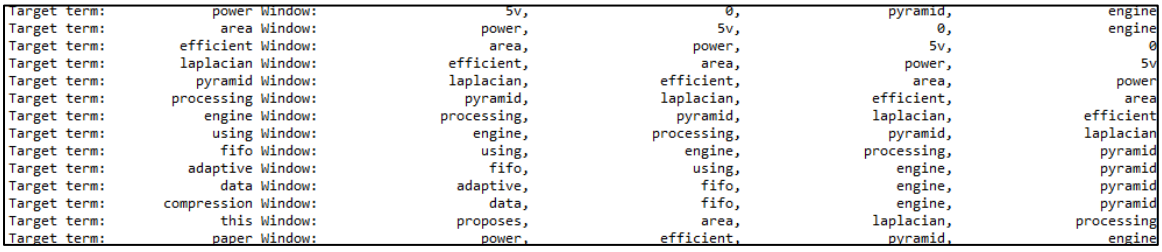
Figure IV-4. Sample List of Concepts Selected from the Wikipedia pages



DOCUMENT# 35	adaptive[0.813]	algorithm[0.208]	awgn[0.726]	based[0.287]
DOCUMENT# 36	achieve[0.336]	adapt[0.460]	adaptive[0.639]	adaptivity[0.522]
DOCUMENT# 37	100[0.403]	20[0.451]	access[0.385]	accuracy[0.273]
DOCUMENT# 38	address[0.281]	algorithmic[0.406]	algorithms[0.341]	apache[0.520]
DOCUMENT# 39	accounted[0.551]	after[0.374]	algorithm[0.548]	also[0.227]
DOCUMENT# 40	alternative[0.478]	based[0.279]	can[0.336]	communication[0.607]
DOCUMENT# 41	after[0.363]	all[0.283]	allow[0.434]	annotated[0.434]
DOCUMENT# 42	79[0.506]	85[0.484]	access[0.563]	also[0.334]
DOCUMENT# 43	accessible[1.505]	affect[0.587]	automatic[0.560]	can[0.218]
DOCUMENT# 44	0[0.438]	100[0.508]	1000[0.702]	97[0.653]
DOCUMENT# 45	all[0.251]	appearance[0.473]	association[0.399]	athletics[0.669]
DOCUMENT# 46	above[0.551]	acquisition[0.508]	affect[0.570]	aligning[0.702]
DOCUMENT# 47	advance[0.564]	afterall[0.473]	algorithm[0.271]	amount[0.268]
DOCUMENT# 48	according[0.771]	algorithm[0.306]	analyzing[0.591]	applications[0.354]
DOCUMENT# 49	1[0.320]	90[0.433]	about[0.329]	accuracy[0.270]
DOCUMENT# 50	abstracts[0.812]	aims[0.384]	annotated[0.708]	annotates[0.444]
DOCUMENT# 51	30[0.337]	accuracy[0.187]	achieve[0.246]	acoustic[0.583]
DOCUMENT# 52	2020[0.470]	abilities[0.437]	above[0.369]	according[0.340]
DOCUMENT# 53	accidents[0.438]	android[0.520]	app[0.684]	assistant[0.520]
DOCUMENT# 54	100[0.317]	about[0.261]	access[0.525]	accustomed[0.437]
DOCUMENT# 55	activities[0.392]	aggregate[0.449]	aggregation[0.483]	algorithms[0.267]
DOCUMENT# 56	active[0.527]	aim[0.420]	approach[0.371]	approaches[0.348]
DOCUMENT# 57	administrators[0.454]	algorithm[0.184]	approach[0.243]	arrive[0.422]
DOCUMENT# 58	11[0.496]	118[0.496]	6[0.403]	75[0.496]
DOCUMENT# 59	abnormalities[0.618]	accurate[0.435]	accurately[0.522]	almost[0.560]
DOCUMENT# 60	according[0.424]	analyze[0.447]	application[0.470]	band[0.476]
DOCUMENT# 61	acquisition[0.475]	agricultural[0.611]	algorithm[0.266]	also[0.270]
DOCUMENT# 62	22[0.537]	all[0.306]	already[0.440]	also[0.336]
DOCUMENT# 63	000[0.463]	36[0.463]	about[0.314]	aircraft[0.848]
DOCUMENT# 64	00[0.463]	05[0.463]	100[0.335]	15[0.408]
DOCUMENT# 65	1[0.313]	2[0.322]	450[0.539]	about[0.322]
DOCUMENT# 66	accuracy[0.314]	adaptive[0.641]	algorithm[0.367]	algorithms[0.354]
DOCUMENT# 67	41[0.439]	43[0.439]	62[0.439]	ability[0.334]

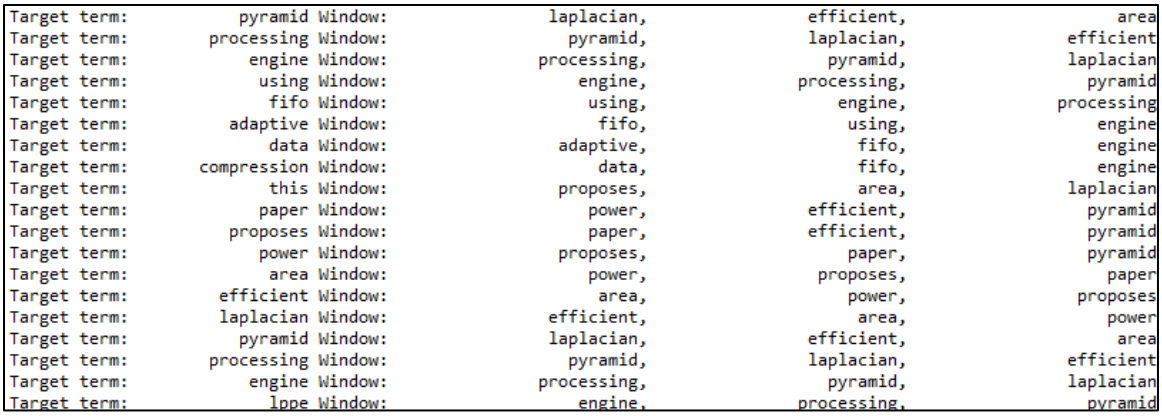
Figure IV-5. Sample Portion of the Document Matrix

After the “Generate Concept Space” module, the document used in the module are then processed in the “Generate Concept Window” module to produce the list of concept space. Figure IV-6 shows a portion of the list of non-unique Concept Windows and Figure IV.7 shows the list of unique Concept Windows.



Target term:	power Window:	5v,	0,	pyramid,	engine
Target term:	area Window:	power,	5v,	0,	engine
Target term:	efficient Window:	area,	power,	5v,	0
Target term:	laplacian Window:	efficient,	area,	power,	5v
Target term:	pyramid Window:	laplacian,	efficient,	area,	power
Target term:	processing Window:	pyramid,	laplacian,	efficient,	area
Target term:	engine Window:	processing,	pyramid,	laplacian,	efficient
Target term:	using Window:	engine,	processing,	pyramid,	laplacian
Target term:	fifo Window:	using,	engine,	processing,	pyramid
Target term:	adaptive Window:	fifo,	using,	engine,	pyramid
Target term:	data Window:	adaptive,	fifo,	engine,	pyramid
Target term:	compression Window:	data,	fifo,	engine,	pyramid
Target term:	this Window:	proposes,	area,	laplacian,	processing
Target term:	paper Window:	power,	efficient,	pyramid,	engine

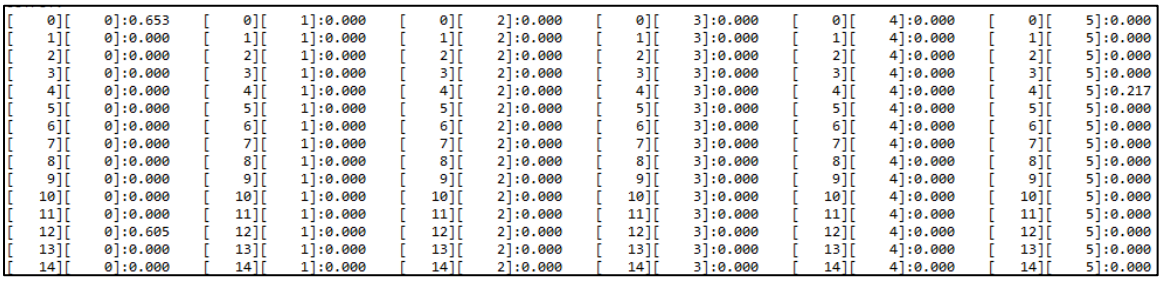
Figure IV-6. Sample Portion of the List of Non-unique Concept Windows



```
Target term:          pyramid Window:          laplacian,          efficient,              area
Target term:       processing Window:            pyramid,          laplacian,         efficient
Target term:           engine Window:         processing,            pyramid,         laplacian
Target term:            using Window:             engine,         processing,           pyramid
Target term:             fifo Window:              using,             engine,        processing
Target term:         adaptive Window:               fifo,              using,            engine
Target term:             data Window:           adaptive,               fifo,            engine
Target term:      compression Window:               data,               fifo,            engine
Target term:             this Window:           proposes,               area,         laplacian
Target term:            paper Window:              power,          efficient,           pyramid
Target term:         proposes Window:              paper,          efficient,           pyramid
Target term:            power Window:           proposes,              paper,           pyramid
Target term:             area Window:              power,           proposes,             paper
Target term:        efficient Window:               area,              power,          proposes
Target term:        laplacian Window:          efficient,               area,             power
Target term:          pyramid Window:          laplacian,          efficient,              area
Target term:       processing Window:            pyramid,          laplacian,         efficient
Target term:           engine Window:         processing,            pyramid,         laplacian
Target term:             lppe Window:             engine,         processing,           pyramid
```

Figure IV-7. Sample Portion of the List of Unique Concept Windows

The document matrix produced by the "Generate Concept Space" module is passed to the 'Construct new document matrix' module. The output of this module is a document matrix where its term is the same as the terms produced by the "Generate Concept Window" module and a sample portion of the matrix is shown in Figure IV-8. The values stored in the new document matrix is displayed in the figure by presenting, first, the term number and the document number then the score of each term.



```
[    0][    0]:0.653   [    0][    1]:0.000   [    0][    2]:0.000   [    0][    3]:0.000   [    0][    4]:0.000   [    0][    5]:0.000
[    1][    0]:0.000   [    1][    1]:0.000   [    1][    2]:0.000   [    1][    3]:0.000   [    1][    4]:0.000   [    1][    5]:0.000
[    2][    0]:0.000   [    2][    1]:0.000   [    2][    2]:0.000   [    2][    3]:0.000   [    2][    4]:0.000   [    2][    5]:0.000
[    3][    0]:0.000   [    3][    1]:0.000   [    3][    2]:0.000   [    3][    3]:0.000   [    3][    4]:0.000   [    3][    5]:0.000
[    4][    0]:0.000   [    4][    1]:0.000   [    4][    2]:0.000   [    4][    3]:0.000   [    4][    4]:0.000   [    4][    5]:0.217
[    5][    0]:0.000   [    5][    1]:0.000   [    5][    2]:0.000   [    5][    3]:0.000   [    5][    4]:0.000   [    5][    5]:0.000
[    6][    0]:0.000   [    6][    1]:0.000   [    6][    2]:0.000   [    6][    3]:0.000   [    6][    4]:0.000   [    6][    5]:0.000
[    7][    0]:0.000   [    7][    1]:0.000   [    7][    2]:0.000   [    7][    3]:0.000   [    7][    4]:0.000   [    7][    5]:0.000
[    8][    0]:0.000   [    8][    1]:0.000   [    8][    2]:0.000   [    8][    3]:0.000   [    8][    4]:0.000   [    8][    5]:0.000
[    9][    0]:0.000   [    9][    1]:0.000   [    9][    2]:0.000   [    9][    3]:0.000   [    9][    4]:0.000   [    9][    5]:0.000
[   10][    0]:0.000   [   10][    1]:0.000   [   10][    2]:0.000   [   10][    3]:0.000   [   10][    4]:0.000   [   10][    5]:0.000
[   11][    0]:0.000   [   11][    1]:0.000   [   11][    2]:0.000   [   11][    3]:0.000   [   11][    4]:0.000   [   11][    5]:0.000
[   12][    0]:0.605   [   12][    1]:0.000   [   12][    2]:0.000   [   12][    3]:0.000   [   12][    4]:0.000   [   12][    5]:0.000
[   13][    0]:0.000   [   13][    1]:0.000   [   13][    2]:0.000   [   13][    3]:0.000   [   13][    4]:0.000   [   13][    5]:0.000
[   14][    0]:0.000   [   14][    1]:0.000   [   14][    2]:0.000   [   14][    3]:0.000   [   14][    4]:0.000   [   14][    5]:0.000
```

Figure IV-8. Sample Portion of the Document Matrix

After producing the new document matrix, the concept space, list of non-unique concept windows, and the new document matrix are passed to the "Construct Concept Vectors" to produce the 3rd-order tensor model. A sample portion of the 3rd-order tensor model is presented in Figure IV-9. The scores stored in the 3rd-order tensor model is

presented by, first, showing the document number then the term and concepts. After that, the score is presented.

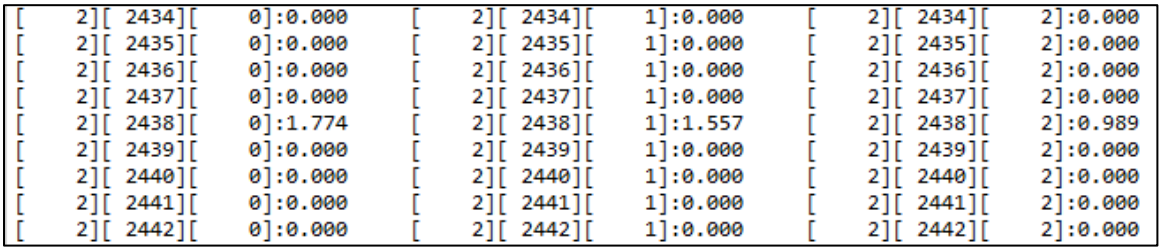


Figure IV-9. Sample Portion of the 3rd-order Tensor Model

After the construction of the 3rd-order tensor model, the model is processed and sorted by the "Sort 3rd - order tensor model" module. A portion of the sorted 3rd - order tensor model is shown in the Figure IV-10.

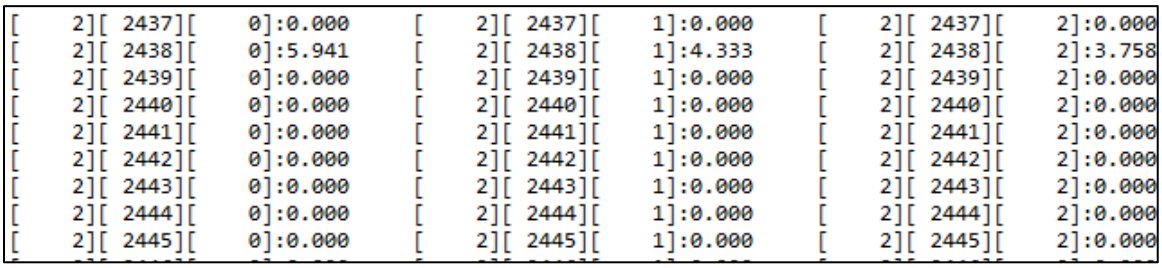


Figure IV-10. Sample Portion of the Sorted 3rd-order Tensor Model

The Sorted 3rd - order tensor model is then passed to the "Generate Super Matrix" module to produce the Super Matrix. A sample portion of the Super Matrix is shown in Figure IV-11.

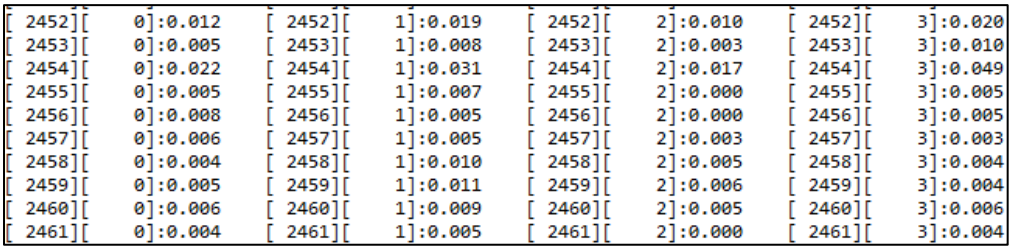


Figure IV-11. Sample Portion of the Super Matrix

The Super Matrix, sorted 3rd - order tensor, list of unique concept windows, and the paths of the list document are passed to the ‘Output to text’ module where these resources are stored in separate text files. The text files are then read by the ‘Conversion of text files into object resources’ where it prepares the resources for the search phase. The search phase begins by accepting inputs from the user and these inputs are retrieved from the system’s user interface show in figure IV.12. The input it requires are the query and the threshold or the alpha value. The queries used in the simulation are listed here:

Q1: determine the proper sense of an ambiguous word in a given context

Q2: the task of determining the correct sense of a word in context

The purpose of Q2 is to demonstrate the result history module retrieving documents from the result set pool. The alpha value used in the simulation is 0.4.

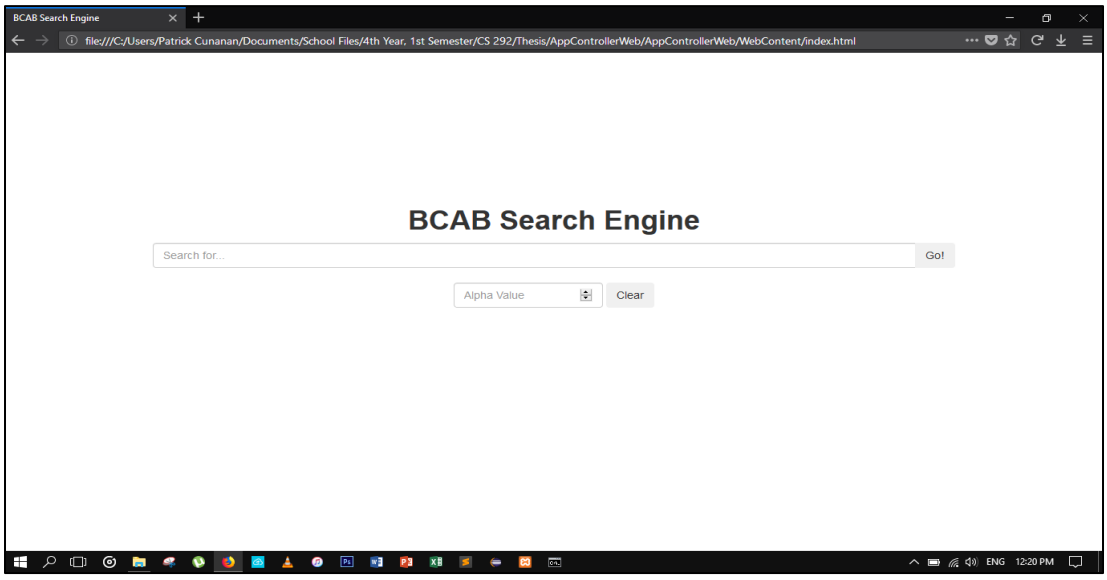


Figure IV-12. User Interface of the Search Phase

Q1 is passed to the preprocessor module and the preprocessed query is shown in Figure IV-13.

```
Preprocessing of the query...
                Processed query: determine, proper, sense, ambiguous, word, given, context
```

Figure IV-13. Tokens of the Preprocessed Q1.

The preprocessed query is then passed to the Lucene search module to produce the user query concept space. Figure IV-14 shows a sample portion of the list of concepts.

```
C:\Users\harji\Documents\DataSet\Concepts\Manananggal.pdf
C:\Users\harji\Documents\DataSet\Concepts\School_of_Isfahan.pdf
C:\Users\harji\Documents\DataSet\Concepts\Database_index.pdf
C:\Users\harji\Documents\DataSet\Concepts\James_Murray_(lexicographer).pdf
C:\Users\harji\Documents\DataSet\Concepts\Intelligence_quotient.pdf
C:\Users\harji\Documents\DataSet\Concepts\International_criminal_law.pdf
C:\Users\harji\Documents\DataSet\Concepts\Domestic_rabbit.pdf
C:\Users\harji\Documents\DataSet\Concepts\Kawit,_Cavite.pdf
```

Figure IV-14. Sample Portion of List of Concepts in User Query Concept Space

The user concept space's size is then feed to the refinement module where it decides whether to limit the 3rd-order tensor model and super matrix or to leave it as is. Figure IV-15 shows the limit size decided by the refinement module.

```
Refinement of third tensor model and super matrix...
                limit size: 68
```

Figure IV-15. Limit Size of the Tensor and Matrix

The limit and the query is passed to the query matrix module where it constructs the query matrix depending on the limit and query specified in the earlier modules. Figure IV-16 shows a sample portion of the query matrix of Q1.

The query matrix is passed to the result history module where it searches for old records and retrieves a similar one to the current query.

```
897][   30]:0.000    [ 4897][   31]:0.000    [ 4897][   32]:0.000    [ 4897][   33]:0.000
898][   30]:0.000    [ 4898][   31]:0.000    [ 4898][   32]:0.000    [ 4898][   33]:0.000
899][   30]:0.000    [ 4899][   31]:0.000    [ 4899][   32]:0.000    [ 4899][   33]:0.000
900][   30]:0.000    [ 4900][   31]:0.000    [ 4900][   32]:0.000    [ 4900][   33]:0.000
901][   30]:0.170    [ 4901][   31]:0.201    [ 4901][   32]:0.193    [ 4901][   33]:0.239
902][   30]:0.000    [ 4902][   31]:0.000    [ 4902][   32]:0.000    [ 4902][   33]:0.000
903][   30]:0.000    [ 4903][   31]:0.000    [ 4903][   32]:0.000    [ 4903][   33]:0.000
904][   30]:0.000    [ 4904][   31]:0.000    [ 4904][   32]:0.000    [ 4904][   33]:0.000
905][   30]:0.000    [ 4905][   31]:0.000    [ 4905][   32]:0.000    [ 4905][   33]:0.000
```

Figure IV-16. Sample Portion of the Query Matrix of Q1

However, since Q1 is the first query there are no records in the result set pool, so the query matrix is passed to the search module to search through the 3rd-order tensor model and retrieve the document set. Figure IV-17 shows the document set of Q1.

Document Path
C:\Users\harji\Documents\DataSet\Documents\A new method for solving context ambiguities using field association knowledge.pdf
C:\Users\harji\Documents\DataSet\Documents\Measuring context-meaning for open class words in Hindi language.pdf
C:\Users\harji\Documents\DataSet\Documents\Disambiguating sentiment ambiguous adjectives.pdf
C:\Users\harji\Documents\DataSet\Documents\A comprehensive analysis of using semantic information in text categorization.pdf
C:\Users\harji\Documents\DataSet\Documents\Semi-supervised Chinese contextual polarity classification with automatic feature selection.pdf
C:\Users\harji\Documents\DataSet\Documents\Sense Annotated Hindi Corpus.pdf
C:\Users\harji\Documents\DataSet\Documents\Using triangulation to identify word senses.pdf
C:\Users\harji\Documents\DataSet\Documents\Framework to extract context vectors from unstructured data using big data analytics.pdf
C:\Users\harji\Documents\DataSet\Documents\A preliminary study on semi-automatic construction of sense tagged corpus with Wordnet senses using semantic vector.pdf

Figure IV-17. Sample Portion of the List of Documents in the Set

Since Q1's resources are stored in the result set pool by the CRUD module, Q2 is used to demonstrate the process of retrieving the document set in the result set pool. The

process of Q2 is the same as the Q2 but instead of using search module, the result history module activates and retrieves a record. Figure IV-18 shows the similarity value between Q1's query matrix and Q2's matrix and the document set which is retrieved in the result set pool is the same list as presented in Figure IV-16.

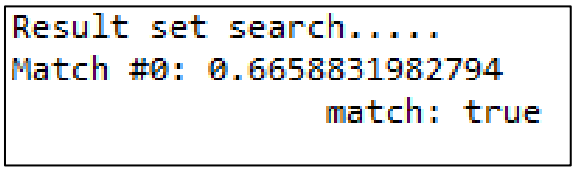


Figure IV-18. Similarity Value between Q1's and Q2's Query Matrix

E. Test Results

In order to test whether the study affects the actual running time in any possible way of the implemented search engine by the researchers, the researchers prepared two sets of queries, which are Set A and Set B respectively, that were defined in Chapter III of the documentation.

The first set, Set A, is used to compare the actual running time of the implemented system with the sorted concepts and pruning feature compared to the original semantic search system by Kim Han-Joon and Hong Ki-Joo.

Set B is used to compare the actual running time between the implemented system by the researchers with result pool and the implemented system also but without the result pool.

Table IV-3 shows the resulting actual running time in seconds for each query per category in Set A after testing it from both the original and the implemented system.

The resulting actual running time yield an average of 11.882 seconds and 6.56 seconds for the original system and the implemented system respectively.

Table IV-4 shows the resulting actual running time in seconds for each query per category in Set B after testing it from both the original and the implemented system.

The resulting actual running time yield an average of 10.136 seconds and 3.547 seconds for the implemented system with result pool and without the result pool respectively.

Table IV-3. Comparison of Actual Running Time (Original and Implemented)

Category	Query	Original (sec)	Implemented (sec)
Data Compression	Q1	12.209	11.244
	Q2	11.485	5.757
	Q3	11.313	8.020
	Q4	11.363	4.204
	Q5	11.485	1.987
Image Processing	Q1	11.915	10.793
	Q2	11.352	3.385
	Q3	11.457	4.977
	Q4	11.243	1.658
	Q5	12.144	12.164
Information Extraction	Q1	12.462	8.657
	Q2	11.827	5.117
	Q3	12.368	11.944
	Q4	12.043	11.384
	Q5	11.675	4.263
Semantic Search	Q1	11.958	11.020
	Q2	12.679	5.899
	Q3	12.687	5.651
	Q4	12.111	1.850
	Q5	12.025	1.163
Word Disambiguation	Q1	12.268	8.843
	Q2	11.764	7.181
	Q3	11.778	5.682
	Q4	11.751	5.511
	Q5	11.700	5.650
Average Actual Running Time		11.882	6.56

In order to test whether the accuracy is maintained, the researchers used the same Set A and Set B used earlier in testing the actual running time. Set A is used to compare the accuracy between the original system and the implemented system, while the Set B is

used to compare the accuracy of using its own result set compared to the accuracy of using the result set of Set $A_{implemented}$.

Table IV-4. Comparison of Actual Running Time (With RP and Without RP)

Category	Query	Without Result Pool (sec)	With Result Pool (sec)
Data Compression	Q1	13.711	3.555
	Q2	14.629	3.490
	Q3	9.910	3.716
	Q4	7.515	3.509
	Q5	6.282	3.503
Image Processing	Q1	13.234	3.568
	Q2	4.678	3.478
	Q3	8.334	3.438
	Q4	5.963	3.632
	Q5	14.969	3.513
Information Extraction	Q1	14.228	3.659
	Q2	8.595	3.587
	Q3	14.176	3.531
	Q4	14.110	3.582
	Q5	7.736	3.769
Semantic Search	Q1	15.082	3.639
	Q2	8.922	3.497
	Q3	8.723	3.518
	Q4	6.339	3.361
	Q5	9.441	3.437
Word Disambiguation	Q1	10.870	3.611
	Q2	10.789	3.482
	Q3	9.640	3.534
	Q4	7.341	3.561
	Q5	8.180	3.513
Average Actual Running Time		10.136	3.547

Tables IV-5 to IV-15 below show the test results and computations obtained in testing for the accuracy using Set A.

Table IV-5. Test and Computation Results for Set $A_{original}$ (D. Compression)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	204	122	142	146	19
Relevant Documents	204	122	142	146	19
True Positive	203	121	141	145	18
False Positive	1	1	1	1	1
False Negative	1	1	1	1	1
Precision	0.995098	0.991803	0.992958	0.993151	0.947368
Recall	0.995098	0.991803	0.992958	0.993151	0.947368
F1-Measure	0.995098	0.991803	0.992958	0.993151	0.947368

Table IV-5 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Data Compression category after testing it to the original system.

Table IV-6. Test and Computation Results for Set $A_{implemented}$ (D. Compression)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	204	122	142	146	19
Relevant Documents	204	122	142	146	19
True Positive	203	121	141	145	18
False Positive	1	1	1	1	1
False Negative	1	1	1	1	1
Precision	0.995098	0.991803	0.992958	0.993151	0.947368
Recall	0.995098	0.991803	0.992958	0.993151	0.947368
F1-Measure	0.995098	0.991803	0.992958	0.993151	0.947368

Table IV-6 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Data Compression category after testing it to the implemented system.

Table IV-7 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Image Processing category after testing it to the original system.

Table IV-7. Test and Computation Results for Set $A_{original}$ (I. Processing)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	153	115	141	67	123
Relevant Documents	170	115	120	67	181
True Positive	151	112	119	67	122
False Positive	2	3	22	0	1
False Negative	19	3	1	0	59
Precision	0.986928	0.973913	0.843972	1	0.99187
Recall	0.888235	0.973913	0.991667	1	0.674033
F1-Measure	0.934985	0.973913	0.911877	1	0.802632

Table IV-8. Test and Computation Results for Set $A_{implemented}$ (I. Processing)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	153	115	141	67	123
Relevant Documents	170	115	120	67	181
True Positive	151	112	119	67	122
False Positive	2	3	22	0	1
False Negative	19	3	1	0	59
Precision	0.986928	0.973913	0.843972	1	0.99187
Recall	0.888235	0.973913	0.991667	1	0.674033
F1-Measure	0.934985	0.973913	0.911877	1	0.802632

Table IV-8 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Image Processing category after testing it to the implemented system.

Table IV-9. Test and Computation Results for Set $A_{original}$ (I. Extraction)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	197	126	78	80	120
Relevant Documents	197	126	164	80	120
True Positive	196	126	76	78	119
False Positive	1	0	2	2	1
False Negative	1	0	88	2	1
Precision	0.994924	1	0.974359	0.975	0.991667
Recall	0.994924	1	0.463415	0.975	0.991667
F1-Measure	0.994924	1	0.628099	0.975	0.991667

Table IV-9 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Information Extraction category after testing it to the original system.

Table IV-10 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Information Extraction category after testing it to the implemented system.

Table IV-10. Test and Computation Results for Set $A_{implemented}$ (I. Extraction)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	197	126	78	80	120
Relevant Documents	197	126	164	80	120
True Positive	196	126	76	78	119
False Positive	1	0	2	2	1
False Negative	1	0	88	2	1
Precision	0.994924	1	0.974359	0.975	0.991667
Recall	0.994924	1	0.463415	0.975	0.991667
F1-Measure	0.994924	1	0.628099	0.975	0.991667

Table IV-11 Test and Computation Results for Set $A_{original}$ (S. Search)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	185	122	134	41	1
Relevant Documents	221	122	134	41	1
True Positive	184	122	133	41	1
False Positive	1	0	1	0	0
False Negative	37	0	1	0	0
Precision	0.994595	1	0.992537	1	1
Recall	0.832579	1	0.992537	1	1
F1-Measure	0.906404	1	0.992537	1	1

Table IV-11 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Semantic Search category after testing it to the original system.

Table IV-12 Test and Computation Results for Set $A_{implemented}$ (S. Search)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	185	122	134	41	1
Relevant Documents	221	122	134	41	1
True Positive	184	122	133	41	1
False Positive	1	0	1	0	0
False Negative	37	0	1	0	0
Precision	0.994595	1	0.992537	1	1
Recall	0.832579	1	0.992537	1	1
F1-Measure	0.906404	1	0.992537	1	1

Table IV-12 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Semantic Search category after testing it to the implemented system.

Table IV-13 Test and Computation Results for Set $A_{original}$ (W. Disamb.)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	87	79	60	78	40
Relevant Documents	87	79	60	78	54
True Positive	84	76	57	74	37
False Positive	3	3	3	4	3
False Negative	3	3	3	4	17
Precision	0.965517	0.962025	0.95	0.948718	0.925
Recall	0.965517	0.962025	0.95	0.948718	0.685185
F1-Measure	0.965517	0.962025	0.95	0.948718	0.787234

Table IV-13 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Word Disambiguation category after testing it to the original system.

Table IV-14 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set A for Word Disambiguation category after testing it to the implemented system.

Table IV-14 Test and Computation Results for Set $A_{implemented}$ (W. Disamb.)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	87	79	60	78	40
Relevant Documents	87	79	60	78	54
True Positive	84	76	57	74	37
False Positive	3	3	3	4	3
False Negative	3	3	3	4	17
Precision	0.965517	0.962025	0.95	0.948718	0.925
Recall	0.965517	0.962025	0.95	0.948718	0.685185
F1-Measure	0.965517	0.962025	0.95	0.948718	0.787234

Table IV-15. Average Precision, Recall, and F1-Measure of Set A

	Original			Implemented		
Category	**Average Precision**	**Average Recall**	**Average F1-Measure**	**Average Precision**	**Average Recall**	**Average F1-Measure**
Data Compression	0.98408	0.98408	0.98408	0.98408	0.98408	0.98408
Image Processing	0.95934	0.90557	0.92468	0.95934	0.90557	0.92468
Information Extraction	0.98719	0.88500	0.91794	0.98719	0.88500	0.91794
Semantic Search	0.99743	0.96502	0.97979	0.99743	0.96502	0.97979
Word Disambiguation	0.95025	0.90229	0.92270	0.95025	0.90229	0.92270
Overall Average	0.97566	0.92839	0.94584	0.97566	0.92839	0.94584

Table IV-15 shows the average precision, recall, and F1-measure for each category of Set A for both the original system and the implemented system. The table also shows the overall average precision, recall, and F1-measure for both the original system and the implemented. The computed results yield an overall average of 0.97566, 0.92839, and 0.94584 of average precision, recall, and F1-measure for the original system, and 0.97566, 0.92839, and 0.94584 of average precision, recall, and F1-measure for the implemented system.

The tables IV-16 to IV-28 below show the test results and computations obtained in testing for the accuracy using Set B.

Table IV-16 shows the similarity scores and their respective alpha (α) Intervals for each query in Set B.

Table IV-16. Similarity Scores and Alpha Intervals for Set B

Category	Query	Similarity Scores	Alpha (α) Intervals
Data Compression	Q1	0.95556	[0.9, 1.0)
	Q2	0.00391	[0.0, 0.1)
	Q3	0.14396	[0.1, 0.2)
	Q4	0.21710	[0.2, 0.3)
	Q5	0.02662	[0.0, 0.1)
Image Processing	Q1	0.00000	[0.0, 0.1)
	Q2	0.00778	[0.0, 0.1)
	Q3	0.16102	[0.1, 0.2)
	Q4	0.00000	[0.0, 0.1)
	Q5	0.00360	[0.0, 0.1)
Information Extraction	Q1	0.91893	[0.9, 1.0)
	Q2	0.97753	[0.9, 1.0)
	Q3	0.31206	[0.3, 0.4)
	Q4	0.45952	[0.4, 0.5)
	Q5	0.00025	[0.0, 0.1)
Semantic Search	Q1	0.14045	[0.1, 0.2)
	Q2	0.13522	[0.1, 0.2)
	Q3	0.99463	[0.9, 1.0)
	Q4	0.00300	[0.0, 0.1)
	Q5	0.00000	[0.0, 0.1)
Word Disambiguation	Q1	0.66588	[0.6, 0.7)
	Q2	0.36358	[0.3, 0.4)
	Q3	0.91890	[0.9, 1.0)
	Q4	0.73170	[0.7, 0.8)
	Q5	0.00064	[0.0, 0.1)

Table IV-17 shows the same similarity scores in Table IV-17 but group by their alpha (α) intervals.

Table IV-18 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each

query in Set B Data Compression category after testing it to the implemented system using its own result set.

Table IV-17. Similarity Scores of Set B Group by Alpha Intervals

Alpha (α) Intervals	Query	Category	Similarity Score
[0.0, 0.1)	Q2	Data Compression	0.00391
	Q5	Data Compression	0.02662
	Q1	Image Processing	0.00000
	Q2	Image Processing	0.00778
	Q4	Image Processing	0.00000
	Q5	Image Processing	0.00360
	Q5	Information Extraction	0.00025
	Q4	Semantic Search	0.00300
	Q5	Semantic Search	0.00000
	Q5	Word Disambiguation	0.00064
[0.1, 0.2)	Q3	Data Compression	0.14396
	Q3	Image Processing	0.16102
	Q1	Semantic Search	0.14045
	Q2	Semantic Search	0.13522
[0.2, 0.3)	Q4	Data Compression	0.21710
[0.3, 0.4)	Q3	Information Extraction	0.31206
	Q2	Word Disambiguation	0.36358
[0.4, 0.5)	Q4	Information Extraction	0.45952
[0.6, 0.7)	Q1	Word Disambiguation	0.66588
[0.7, 0.8)	Q4	Word Disambiguation	0.73170
[0.9, 1.0)	Q1	Data Compression	0.95556
	Q1	Information Extraction	0.91893
	Q2	Information Extraction	0.97753
	Q3	Semantic Search	0.99463
	Q3	Word Disambiguation	0.99463

Table IV-18. Test and Computation Results for Set B (D. Compression)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	188	113	132	114	30
Relevant Documents	195	143	132	114	30
True Positive	185	113	132	114	29
False Positive	3	0	0	0	1
False Negative	10	30	0	0	1
Precision	0.984043	1	1	1	0.966667
Recall	0.948718	0.79021	1	1	0.966667
F1-Measure	0.966057	0.882813	1	1	0.966667

Table IV-19. Test and Computation Results for Set B' (D. Compression)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	204	122	142	146	19
Relevant Documents	195	143	132	114	30
True Positive	184	70	87	81	7
False Positive	20	52	55	65	12
False Negative	11	73	45	33	23
Precision	0.901961	0.57377	0.612676	0.554795	0.368421
Recall	0.94359	0.48951	0.659091	0.710526	0.233333
F1-Measure	0.922306	0.528302	0.635036	0.623077	0.285714

Table IV-19 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained also for each query in Set B Data Compression category after testing it to the implemented system but using the result set of Set $A_{proposed}$ instead of its own result set.

Table IV-20. Test and Computation Results for Set B (I. Processing)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	28	1	178	7	114
Relevant Documents	75	1	117	7	174
True Positive	27	1	117	7	114
False Positive	1	0	61	0	0
False Negative	48	0	0	0	60
Precision	0.964286	1	0.657303	1	1
Recall	0.36	1	1	1	0.655172
F1-Measure	0.524272	1	0.79322	1	0.791667

Table IV-20 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set B Image Processing category after testing it to the implemented system using its own result set.

Table IV-21. Test and Computation Results for Set B' (I. Processing)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	153	115	141	67	123
Relevant Documents	75	4	117	7	174
True Positive	49	1	67	0	92
False Positive	104	114	74	67	31
False Negative	26	3	50	7	82
Precision	0.320261	0.008696	0.475177	0	0.747967
Recall	0.653333	0.25	0.57265	0	0.528736
F1-Measure	0.429825	0.016807	0.51938	0	0.619529

Table IV-21 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained also for each query in Set B Image Processing Category after testing it to the implemented system but using the result set of Set $A_{proposed}$ instead of its own result set.

Table IV-22. Test and Computation Results for Set B (I. Extraction)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	174	123	94	54	110
Relevant Documents	178	123	175	54	110
True Positive	171	123	91	52	110
False Positive	3	0	3	2	0
False Negative	7	0	84	2	0
Precision	0.982759	1	0.968085	0.962963	1
Recall	0.960674	1	0.52	0.962963	1
F1-Measure	0.971591	1	0.67658	0.962963	1

Table IV-22 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set B Information Extraction category after testing it to the implemented system using its own result set.

Table IV-23. Test and Computation Results for Set B' (I. Extraction)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	197	126	78	80	120
Relevant Documents	178	123	175	54	110
True Positive	165	123	66	46	53
False Positive	32	3	12	34	67
False Negative	13	0	109	8	57
Precision	0.837563	0.97619	0.846154	0.575	0.441667
Recall	0.926966	1	0.377143	0.851852	0.481818
F1-Measure	0.88	0.987952	0.521739	0.686567	0.46087

Table IV-23 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained also for each query in Set B Information Extraction Category after testing it to the implemented system but using the result set of Set $A_{proposed}$ instead of its own result set.

Table IV-24. Test and Computation Results for Set B (S. Search)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	185	133	133	89	1
Relevant Documents	213	133	133	89	1
True Positive	182	132	132	88	1
False Positive	3	1	1	1	0
False Negative	31	1	1	1	0
Precision	0.983784	0.992481	0.992481	0.988764	1
Recall	0.85446	0.992481	0.992481	0.988764	1
F1-Measure	0.914573	0.992481	0.992481	0.988764	1

Table IV-24 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set B Semantic Search category after testing it to the implemented system using its own result set.

Table IV-25. Test and Computation Results for Set B' (S. Search)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	185	122	134	41	1
Relevant Documents	213	133	133	89	1
True Positive	162	71	130	19	1
False Positive	23	51	4	22	0
False Negative	51	62	3	70	0
Precision	0.875676	0.581967	0.970149	0.463415	1
Recall	0.760563	0.533835	0.977444	0.213483	1
F1-Measure	0.81407	0.556863	0.973783	0.292308	1

Table IV-25 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained also for each query in Set B Semantic Search Category after testing it to the implemented system but using the result set of Set $A_{proposed}$ instead of its own result set.

Table IV-26. Test and Computation Results for Set B (W. Disamb.)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	88	84	57	56	1
Relevant Documents	88	84	57	56	1
True Positive	85	84	54	52	1
False Positive	3	0	3	4	0
False Negative	3	0	3	4	0
Precision	0.965909	1	0.947368	0.928571	1
Recall	0.965909	1	0.947368	0.928571	1
F1-Measure	0.965909	1	0.947368	0.928571	1

Table IV-26 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained for each query in Set B Word Disambiguation category after testing it to the implemented system using its own result set.

Table IV-27 shows the number of retrieved documents, relevant documents, true positive, false positive, false negative, precision, recall, and F1-measure obtained also for each query in Set B Word Disambiguation Category after testing it to the implemented system but using the result set of Set $A_{proposed}$ instead of its own result set.

Table IV-27. Test and Computation Results for Set B' (W. Disamb.)

	Q1	Q2	Q3	Q4	Q5
Retrieved Documents	87	79	60	78	1
Relevant Documents	88	84	57	56	1
True Positive	81	46	53	52	1
False Positive	6	33	7	26	0
False Negative	7	38	4	4	0
Precision	0.931034	0.582278	0.883333	0.666667	1
Recall	0.920455	0.547619	0.929825	0.928571	1
F1-Measure	0.925714	0.564417	0.905983	0.776119	1

Table IV-28. Average Precision, Recall, and F1-Measure of Set B

	Set B			Set B'		
Category	**Average Precision**	**Average Recall**	**Average F1-Measure**	**Average Precision**	**Average Recall**	**Average F1-Measure**
Data Compression	0.99014	0.94112	0.96311	0.60233	0.60721	0.59889
Image Processing	0.92432	0.80303	0.82183	0.31042	0.40094	0.31711
Information Extraction	0.98276	0.88873	0.92223	0.73532	0.72756	0.70743
Semantic Search	0.99150	0.96564	0.97766	0.77824	0.69707	0.72741
Word Disambiguation	0.95766	0.95766	0.95766	0.77766	0.78315	0.77195
Total Average	0.96928	0.91124	0.92850	0.64079	0.64319	0.62456

Table IV-28 shows the average precision, recall, and F1-measure for each category of Set B. The table also shows the overall average precision, recall, and F1-measure for both Set B and Set B'. The computed results yield an overall average of 0.96928, 0.91124,

and 0.92850 of average precision, recall, and F1-measure for Set B, and 0.64079, 0.64319, and 0.062456 of average precision, recall, and F1-measure for Set B'.

In the significance testing, the values for the Mean Difference, Standard Deviation of Mean Difference, Standard Error of Mean Difference, Alpha Level T 95% CI, and the Upper and Lower Confidence, that were computed using the equations provided in Chapter III, for Set A (Unsorted and Sorted) are shown in Table IV-29.

Values for the Mean Difference, Standard Deviation of Mean Difference, Standard Error of Mean Difference, Alpha Level T 95% CI, and the Upper and Lower Confidence, that were computed using the equations provided in Chapter III, for Set B (With and Without result set pool) are shown in Table IV-30.

Table IV-29. Set A T-Test Values

Mean Difference	**5.322267582**
Standard Deviation of Mean Difference	**3.269630546**
Standard Error of Mean Difference	**0.653926109**
T alpha half of 95% CI	**2.390949315**
Upper Confidence Level	**6.885771765**
Lower Confidence Level	**3.758763398**

Table IV-30. Set B T-Test Values

Mean Difference	**6.5884477**
Standard Deviation of Mean Difference	**3.215360846**
Standard Error of Mean Difference	**0.643072169**
T alpha half of 95% CI	**2.390949315**
Upper Confidence Level	**8.126000662**
Lower Confidence Level	**5.050894737**

Values garnered from the Data Analysis option, which include the Mean, number of observations, the hypothesized mean difference, degrees of freedom, the t-statistic, p-value for one-tailed test, and the t critical one-tailed value for Set A (Unsorted and Sorted) and Set B (With and Without result set pool) from Excel are shown below in Tables IV-31 and IV-32, respectively.

Table IV-31 Paired T-Test Results for Set A

SET A		
	Unsorted (Current)	***Sorted (Implemented)***
Mean	11.88251	6.560241
Observations	25	25
Hypothesized Mean Difference	0	
df	24	
t Stat	**8.138943385**	
P(T<=t) one-tailed	2.32374E-08	
t Critical one-tailed	**2.390949315**	

Table IV-32 Paired T-Test Results for Set B

SET B		
	Without result set pool	***With result set pool***
Mean	10.13578	3.547329
Observations	25	25
Hypothesized Mean Difference	0	
df	24	
t Stat	**10.24526953**	
P(T<=t) one-tailed	**3.06E-10**	
t Critical one-tailed	**2.390949315**	

F. Analysis and Interpretations of the Results

In Table IV.3, It can be seen in that the relation of the query's length is related to the running time done by the system. This is because the number of tokens in the query, without the stop words, are equal to the number of columns filled up with scores in the query matrix. Thus, a query matrix with less values can have a running time that is less than a query matrix with more values.

Table IV-33. T-Statistic & Critical One-tailed Value (Set A)

	Set $A_{Original}$	Set $A_{implemented}$
Mean	11.883	6.560
Observations	25	25
Alpha	**0.05**	
t-Statistic	**8.139**	
Critical One-Tailed Value	**1.711**	

The data presented from Table IV-3 are used to perform the t test introduced in Chapter 3. Using the formula presented in Chapter III, the results of the formula are presented in Table IV-33. The alpha that is going to be used in the t test is 0.05 and the null hypothesis in the t test of set $A_{original}$ and set $A_{implemented}$ is $\mu = 11.883$. The alternative hypothesis in the t test of set $A_{original}$ and set $A_{implemented}$ is $\mu = 6.560$.

The results of the t-test in Set A, wherein the value of the t-statistic is approximately 8.139, indicate that these results are occurring about 8.139 standard deviations from the mean and it also signifies the rejection of the null hypothesis since the conditions to reject the null hypothesis are: the t critical one-tailed value should be less than the t-stat value and the P(T<=t) one-tailed value should be less than the alpha value, 0.05. In this data set and results, both conditions are met and the null hypothesis is rejected. A rejection to the null hypothesis indicates that the system that has the implemented feature of refining the 3^{rd}-order tensor model and super matrix is more efficient than the system without the said implemented feature.

The values of the t-test computed using the equations in Chapter III also present remarkable findings. The researchers state that they are 95% confident that the mean difference of Set A, which is approximately 5.322, between the actual running times at Unsorted and Sorted is between approximately 6.886 and 3.759.

Table IV-34. T-Statistic & Critical One-tailed Value (Set B)

	Set $B_{with\ RP}$	Set $B_{without\ RP}$
Mean	10.136	3.547
Observations	25	25
Alpha	**0.05**	
t-Statistic	**10.245**	
Critical One-Tailed Value	**1.711**	

Data from table IV-4 are used to calculate the t-statistic and critical one-tailed value for the parameters needed in performing the t test and these parameters are presented in Table IV-34. The alpha used in the t-test of results from comparing set $B_{with\ RP}$ and set $B_{without\ RP}$ is 0.05. The nnull hypothesis of the t-test is $\mu = 10.136$, a d the alternative hypothesis used is $\mu = 3.547$.

The results of the t-test in Set B, wherein the value of the t-statistic is approximately 10.245, indicate that these results are occurring about 10.245 standard deviations from the mean and it also signifies the rejection of the null hypothesis since the conditions to reject the null hypothesis are: the t critical one-tailed value should be less than the t-stat value and the P(T<=t) one-tailed value should be less than the alpha value, 0.05. In this data set and results, both conditions are met and the null hypothesis is rejected. The rejection of the null hypothesis means that the gathered running time of using Set B in a system, where it has an implemented feature of using a result pool for retrieving stored document set and also has an implemented feature of limiting the 3rd-order tensor model and super matrix, is more efficient compared with the gathered running time of using Set B in a system, where it only has a feature of limiting or refining the 3rd-order tensor model and super matrix and the process of retrieving document set is done by searching through 3rd-order tensor model.

The values of the t-test computed using the equations in Chapter III also present remarkable findings similar to Set A. The researchers state that they are 95% confident that the mean difference of Set B, which is approximately 6.588, between the actual running times at Sets without Result Set Pool and With Result Set Pool is between approximately 8.126 and 5.051.

Table IV-35. Computation of Space Complexity for the Creation Phase

Module	Original	Implemented
Concept Space	O((D·T)+D+C)	O((D·T)+D+C)
Concept Window	O(D·(T·9))	O(D·(T·9))
Concept Vector	O(D·T·C)	O(D·T·C)
Tensor Sort	N/A	O(C)
Super Matrix	O(D·T·C)	O(D·T·C)
Total	O((D·T)+D+C)+O(D·(T·9)) +O(2(D·T·C))	O((D·T)+D+C)+O(D·(T·9)) +O(C)+O(2(D·T·C))

Table IV-35 shows the space complexity for each module of the creation phase for both the original and implemented system, where D is the number of documents, T is the number of terms in the entire document, and C is the number of concepts in the concept space. The only changes made by the researchers in the creation phase is the sorting of the 3rd-order tensor model, so other than the tensor sort module there are no changes in terms of space complexity from other modules. Since the researchers used merge sort algorithm to sort the 3rd-order tensor model, there is an additional O(C) space because a temporary storage is required in order to sort using merge sort algorithm.

Table IV-36. Computation of Space Complexity for the Search Phase

Module	Original	Implemented
User Query Concept Space	O(D·T·C)	O(D·T·C')
Processor	O(Q)	O(Q)
Query Matrix	O(D·T·C)	O(D·T·C')
Result Pool	N/A	O(R)
Search Module	O(R)	O(R)
Total	O(2(D·T·C))+ O(Q)+ O(R)	O(2·(D·T·C'))+ O(Q)+ O(R)

Table IV-36 shows the space complexity for each module of the search phase for both the original and implemented system, where D is the number of documents, T is the number of terms in the entire documents, C is the number of concepts, C' is the number of pruned concepts, Q is the number of terms in the query not including stop words, and R is the number of documents retrieved from the search module or the result pool. The user query concept space module is the one responsible for pruning unrelated concepts to the concept space, so the difference between the original and the implemented system is the original used all the concepts in the concept space while the implemented system only used the related concepts from the concept space.

The 3rd-order tensor model used in the Set $A_{implemented}$ is processed in the "Sort 3rd-order tensor model", so each document has different order of concept lists. However, the order in each of these list follows the descending order so high valued concepts are moved to the front. Because of these, the result of true positive, false negative, and false positive from Set $A_{implemented}$, presented in tables IV-6, IV-8, IV-10, IV-12 and IV-14, compared with the results from the Set $A_{original}$,, presented in tables IV-5, IV-7, IV-9, IV-11, and IV-13, are equal. Thus, using the mean squared error to the queries in Set $A_{implemented}$ and the queries in Set $A_{original}$ for the comparison of their precision, recall, and F1 scores, it shows that the error scores are equal to zero and these error scores are presented in Table IV-37, Table IV-38, and Table IV-39. The error scores presented in the said tables indicate that the implementation of the refinement process of the 3rd-order tensor model and super matrix does not change the accuracy of the system.

Table IV-37. MSE Scores for Precision scores (Set $A_{original}$ & Set $A_{implemented}$)

Category	Q_1	Q_2	Q_3	Q_4	Q_5
Data compression	0	0	0	0	0
Word disambiguation	0	0	0	0	0
Image processing	0	0	0	0	0
Information extraction	0	0	0	0	0
Semantic search	0	0	0	0	0

Table IV-38. MSE Scores for Recall scores (Set $A_{original}$ & Set $A_{implemented}$)

Category	Q_1	Q_2	Q_3	Q_4	Q_5
Data compression	0	0	0	0	0
Word disambiguation	0	0	0	0	0
Image processing	0	0	0	0	0
Information extraction	0	0	0	0	0
Semantic search	0	0	0	0	0

Table IV-39. MSE Scores for F1 scores (Set $A_{original}$ & Set $A_{implemented}$)

Category	Q_1	Q_2	Q_3	Q_4	Q_5
Data compression	0	0	0	0	0
Word disambiguation	0	0	0	0	0
Image processing	0	0	0	0	0
Information extraction	0	0	0	0	0
Semantic search	0	0	0	0	0

Table IV-16 shows the similarity scores of pairing made from set B queries and set A queries; the queries and pairings made are listed in Appendix B. The table also shows that some of the similarity scores are low and high despite the fact that their scores should be higher and lower respectively. This is because the metric used in comparing queries in the result set pool is based on query matrices. Query matrices are built based on the tokens in the query using the super matrix so if a token in the query is synonym to a term in the

super matrix then the term from the super matrix is neglected. The term can be added in the query matrix only if the term from the super matrix matches the term from the query using lexical matching. Thus, the reason why some scores are high and some are very low is because the query matrices used in searching the result set pool does not detect synonyms words. Another reason can be the lack of documents used in the dataset because each term depends on their concept window and if there are not enough concept window to describe a term then the meaning of the term is limited or uncertain.

Using Tables IV-18 and IV-19, the data are used to calculated mean square errors for precision score, recall score, and F1 score of queries under data compression. The mean square error scores are shown in Table IV-40. The tables indicate that there is a 17.90% precision score error, 16.56% recall score error, and 17.33% score error from comparing the document set of set B from the search the 3rd-order tensor model and the document set of Set B from obtaining the document set from the result set pool. This means that the system with the implemented feature of searching through the result set pool for efficiency gave a document set with an error score of 17.9%, 16.56%, and 17.44% in the precision, recall, and F1 score as compared with the document set in searching the query through the 3rd-order tensor model. The error scores that is presented in the table can be explained by the limitation of comparing query matrices in the result set history module. It was discussed that the limitation or problem of the metric used in the result history could not handle synonym words and it highly depends on the size of data set used.

Based from the queries used under data compression between Set B and Set A, from Table IV-16, from Table IV-18 and IV-19, it can be observed that Q1 of Set B and Q1 of Set A have scores on precision, recall, and F1 that are close to each other. This means that

a detailed query can have a high similarity value with its paraphrase sentence and have a high accuracy score from the document set retrieved from the other. It can also be observed that precision, recall, and F1 score that are low from using the result set of Set A are caused by having low similarity value with the queries in the Set B.

In Table IV-19 and IV-16, it can be observed that the relative increase in the similarity score does not reflect the increase of accuracy like Q3 and Q4 under Set A and set B. There is an increase of similarity value from Q3 to Q4, but there is a decrease in the accuracy from Q3 to Q4. The accuracy and similarity score of Q1 from set A and set B, however, seems to be dependent to each other.

Table IV-20 and IV-21 were used to calculate the mean square errors for precision, recall, and F1 score of queries under image processing and the results of the calculations are stored in table IV-41.

Table IV-40. MSEs of Eval Scores from Queries (Data Compression)

Scores	Mean square error (MSE)
Precision	.1790
Recall	.1656
F1	.1733

The MSE scores presented in the table shows that the errors are high. The reason for this is the metric used in comparing query matrices in the result history module and based from the discussed problems discovered in the metric. The cause for the high error scores listed in table is the metric in choosing the similar query in the result set pool because of the problems discovered from using it which was discussed in earlier.

Table IV-41. MSEs of Eval Scores from Queries (Image Processing)

Scores	Mean square error (MSE)
Precision	0.498828

Recall	0.369432
F1	0.416042

Data from Table IV-22 and IV-23 are used to compute the MSE of precision, recall, F1 score of queries under Information extraction. The results of the computation made are listed in table IV-42. Based from the results in Table IV-42, it indicates that the score of error presented are reflected by the problems discussed in the metric used for searching queries in the result set pool.

Table IV-43 shows the MSE calculations made from using the data from table IV-24 and IV-25 which used queries under semantic search.

Table IV-42. MSEs of Eval Scores from Queries (Information Extraction)

Scores	Mean square error (MSE)
Precision	0.105199
Recall	0.134242
F1	0.079913

The MSE scores calculated shows that the metric used reflects the accuracy of each document set retrieved from the result set pool.

Table IV-43. MSEs of Eval Scores from Queries (Semantic Search)

Scores	Mean square error (MSE)
Precision	0.09134
Recall	0.164092
F1	0.137053

Data listed in table IV-26 and IV-27 are used to compute the MSE scores for the precision, recall, and F1 scores of queries under word disambiguation. Table IV-44 shows the MSE calculations and it shows that the MSE scores low. This is because the queries from the Set A and their paraphrased sentences have little difference in terms of lexical and based from the discussion made on the problem of the metric, it was discussed that the

metric depends on lexical matching. Thus, if majority of term between two query matrices are the same then they have a high similarity score or value and the accuracy of their document set are the approximately the same.

Table IV-44. MSEs of Eval Scores for Queries (Word Disambiguation)

Scores	Mean square error (MSE)
Precision	0.04968
Recall	0.041404
F1	0.043261

Table IV-17 describes the list of similarities arranged by their alpha intervals. The MSE are calculated from using the queries in each interval and tables from IV-18 to IV-27 and the results of the calculation is presented in Table IV-45. The table indicates that the increasing order of the alpha interval does not reflect the order presented in precision, recall, and F1. This means that the assumption where two similar queries have high similarity score and their document set are approximately the same can be violated. The reason for the order of errors to appear like this is explained by the problems discussed on the metric in comparing the similarities between two query matrices.

Table IV-45. MSEs of Eval Scores for Queries under each Alpha Interval

Alpha Interval	$Precision_{MSE}$	$Recall_{MSE}$	$F1_{MSE}$
[0.0, 0.1)	0.36002	0.31623	0.337031
[0.1, 0.2)	0.102897	0.121399	0.104231
[0.2, 0.3)	0.198207	0.083795	0.142071
[0.3, 0.4)	0.094679	0.112528	0.106854
[0.5, 0.6)	0.150515	0.012346	0.076395
[0.6, 0.7)	0.001216	0.002066	0.001616
[0.7, 0.8)	0.068594	0	0.023242
[0.9, 1.0)	0.006597	0.000339	0.002502

G. Summary of Findings

The summary of findings discovered and analyzed from the previous section are as follows:

- The system with the refinement module offers same accuracy with the original system but it is more efficient in terms of actual running time.
- The system with the result history module offers low accuracy as compared to the original system where the searching through the 3rd-order tensor model is processed.
- The similarity metric and method used in the result history module depends on lexical matching. Thus, it cannot detect synonym words.
- The creation phase with the sorting module needs more space than the original system

Chapter V Summary, Conclusions and Recommendations

A. Summary

In this study, An IR-Based Semantic Search Approach on Portable Document Format (PDF) Files using Similarity Measure, the proponents created an intelligent agent that has semantic search capabilities that can generate a list of relevant documents that results from the user query, taking note of the accuracy and reliability of the retrieved documents. The relevance of all documents included in this list is closely examined and inspected by the proponents. With this in mind, evaluation tools such as the precision and recall, and the F1-measure are being utilized and used in order to check on the acquired results' reliability and correctness.

Lastly, the system is able to efficiently and effectively generate a list of relevant documents that are related to the user query. The system is also able to decrease the actual time for search based on the evaluation tools' results. Results from the significance testing shows that there is significant improvement and difference between the existing system and the implemented system.

B. Conclusions

As shown in the study, the new approach was successful in accepting and handling several user queries using semantic methods and information retrieval algorithms. These algorithms were successful in processing these queries and generating the list of documents based on the user query provided. The first and second features of the implemented system, which are the limiting and the result set pool implementations respectively, were properly implemented and integrated into the new system.

The evaluation results show remarkable results, as shown in tables found in Chapter IV. Results indicate that the accuracy of the proposed stayed the same and the actual time or speed is decreased by 30%, compared to the existing system. The actual running time of implemented system also changes variably in every trial or search and the actual running time of current search engine always stays in 11 seconds. This is because the current search engine always uses the full dimensions of the model while the implemented system uses the size of the user query concept space to limit process of searching in using the whole dimensions.

The implemented system garnered mean square error values shown in Table IV-45, which shows that the accuracy of the output produced from the implemented result set feature has errors relating to the values compared to the existing system. This clearly shows that difference between the accuracy of the existing and the implemented system is minimal and is negligible.

The system was also built to have a familiar interface; one that is usable and efficient in accepting user queries from end users without giving them a hard time. The list of documents in the user interface was also displayed in a way that the users can view which are most relevant to their query. The significance study also shows that the implemented system is significantly different and faster in actual running time than the existing system. The implemented system improved on the actual running time, but stayed the same on the accuracy.

Ultimately, the goal of the researchers in administering this research study was partially but correctly achieved and concluded.

C. Recommendations

If further improvements are to be made on this research study, the researchers would like to recommend the following notes and opinions:

1. Instead of utilizing the metric used by the proponents, use other third party metrics such as Semilar.
2. Provide adjustments on the algorithms, either by simplifying or enhancing, in order to acquire better speed and accuracy in the search process.
3. Provide adjustments on the algorithms being utilized in order to acquire better results on the evaluation tools used such as precision and recall, along with the F1-measure.
4. Utilize a more specific algorithm other than that used by the proponents.
5. Take into consideration other file types other than PDF files.
6. Consider including sentences with special symbols such as hyphens, commas, semicolons, etc.
7. Limit database by finding a metric that chooses database records to remove.

References

Amerland, D. (2014). *Google™ Semantic Search: Search Engine Optimization (SEO) Techniques That Get Your Company More Traffic, Increase Brand Impact, and Amplify Your Online Presence, 1st Ed*. Indiana: Pearson Education. doi:0789751348

Ashby, F. G., & Ennis, D. M. (2007). Similarity measures. Retrieved April 04, 2017, from http://www.scholarpedia.org/article/Similarity_measures,doi:10.4249/scholarpedia.4116

Banarjee, S. (2002). *Adapting the Lesk Algorithm for Word Sense Disambiguation to WordNet*. Duluth, Minnesota. University of Minnesota. DOI: 55812

Chowdhury, G. (2010). Basic concepts of information retrieval system [Abstract]. *Introduction to modern information retrieval*. doi:9781856046947

Dubin, D., & Smith, L. (2000). Major Events in the History of Information Retrieval. Retrieved April 04, 2017, from http://courseweb.ischool.illinois.edu/~kmedina/Lis329/IRHistory.html

Edwards, S. (2015). The Rapid Evolution of Semantic Search. Retrieved March 2, 2017, from http://www.inc.com/samuel-edwards/the-rapid-evolution-of-semantic-search.html

Fatima, A., Luca, C., & Wilson, G. (2014). *New Framework for Semantic Search Engine* (Unpublished master's thesis). Anglia Ruskin University. doi:10.1109/UKSim.2014.114

Feist, K. (2015). Compare Databases and Search Engines. Retrieved March 3, 2017, from http://www.library.illinois.edu/ugl/howdoi/compare1.html

Forsythe, G. (n.d.). History of Information Retrieval. Retrieved April 04, 2017, from https://www.asindexing.org/about-indexing/history-of-information-retrieval/

Herbirch, R., & Graepel, T. (2010). *Handbook of Natural Language Processing* (2nd ed.). Florida: CRC Press.

Hong, K., & Kim, H. (2016). *A semantic search technique with Wikipedia-based text representation model,* 177-182. doi: 10.1109/BIGCOMP.2016.7425818

Libut, R., Llaneta, M., Rey Lara, R., & Valencia, J. (2014). *Text - Based Project Document File (PDF) Search Engine Using XRank Algorithm* (Undergraduate). University of Santo Tomas.

Ma, S., Zhao, W., Zhang, S., & Zhang, H. (2012). *Material Hub: A Semantic Search Engine with Rule Reasoning* (Unpublished doctoral dissertation). Peking University. doi: 10.1109/COMPSACW.2012.17

Manning, C. D., Raghayan, P., & Schütze, H. (2009). Introduction to information retrieval. doi: 978 0 521 86571 5

Mulles, C., Pasion, G., See, S., & Sison, D. (2011). *What You Search Is What You Get: A Query - based Automatic Document Searching Using Semantic Search* (Undergraduate). University of Santo Tomas.

Orav, H., Fellbaum, C., & Vossen P. (2014). *Proceedings of the 7th International Global WordNet Conference (GWC 2014)*. pp. 78-85. Tartu, Estonia

PennState University. (2014). 1(b).2.1: Measures of Similarity and Dissimilarity. Retrieved April 04, 2017, from https://onlinecourses.science.psu.edu/stat857/node/3

Petkova, T. (2016, May). Semantic Search: The Paradigm Shift from Results to Relationships. Retrieved March 2, 2017, from http://ontotext.com/semantic-search-the-paradigm-shift-from-results-to-relationships/

Petrovic, D. (2010). Semantic Web: Part of Business World. Retrieved March 2, 2017, from http://www.semanticweb.rs/Article.aspx?iddoc=32&id=65&lang=2

Rus, V., Lintean, M. C., Banjade, R., Niraula, N. B., & Stefanescu, D. (2013, August). SEMILAR: The Semantic Similarity Toolkit. In ACL (Conference System Demonstrations) (pp. 163-168).

Sanders, A. (2016). What Is Semantic Search and What Should You Do About It? Retrieved March 1, 2017, from https://moz.com/blog/what-is-semantic-search

Sanderson, M., & Croft, W. (n.d.). *The History of Information Retrieval Research* [Pamphlet]. CIIR Publications.

Tan, C., & Sumanaweera, H. (n.d.). Information Retrieval. Retrieved March 29, 2017, from http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/hks/inf_ret.html

Vickery, B. (1959). The Structure of Information Retrieval Systems. In *Proceedings of the International Conference on Scientific Information: Two Volumes* (Vol. 2, pp. 1275-1277). Washington D.C.: The National Academies Press. doi:https://doi.org/10.17226/10866

Wall, A. (2006). History of Search Engines: From 1945 to Google Today. Retrieved February 28, 2017, from http://www.searchenginehistory.com/

What is WordNet. (2015, March 17). Retrieved from https://wordnet.princeton.edu.

W.S.D. (2015). Retrieved April 04, 2017, from http://www.nltk.org/howto/wsd.html

Ydav, U., Narula, G., & Duhan, N. (2016). *A novel approach for precise search results retrieval based on semantic web technologies* (Unpublished doctoral dissertation). YMCA University of Science and Technology. doi: no doi available.

Yedidia, A. (2016, December). *Against the F-score*. Essay. Retrieved from https://adamyedidia.files.wordpress.com/2014/11/f_score.pdf.

Appendices

A. List of Documents for Each Category

Table A-1. List of Document for Each Category

Category	Titles
Data Compression	• A 0.5V power and area efficient Laplacian Pyramid processing engine using FIFO with adaptive data compression.pdf • A case for Core-Assisted Bottleneck Acceleration in GPUs Enabling flexible data compression with assist warps.pdf • A data compression application for wireless sensor networks using LTC algorithm.pdf • A data compression scheme for reliable data storage in non-volatile memories.pdf • A low-energy ASIP with flexible exponential Golomb codec for lossless data compression toward artificial vision systems.pdf • A multi-stage data compression and transmission method for bulk data in low speed network.pdf • A nonparametric neural signal processor for online data compression and power management.pdf • Acceleration of ultrasonic data compression using OpenCL on GPU.pdf • Adaptive Sensor Data Compression in IoT systems Sensor data analytics based approach.pdf • An alternative voltage and frequency monitoring scheme for SCADA based communication in power system using data compression.pdf • An implementation of energy efficient data compression & security mechanism in clustered wireless sensor network.pdf • Application of a real-time data compression and adapted protocol technique for WAMS.pdf • Comparison of wavelet thresholding methods for industrial data compression.pdf • Data Compression Cost Optimization.pdf • Data Compression of Digital-Ink with Pen-Slips Using Multi-level L1 Smoothing Splines.pdf • Data compression of excitatory postsynaptic potentials.pdf • Distributed DCT based data compression in clustered wireless sensor networks.pdf • ECG data compression for a portable ECG recorder and transmitter.pdf • Electrocardiogram data compression using adaptive bit encoding of the discrete Fourier transforms coefficients.pdf

	• Enhanced LZMA and BZIP2 for improved energy data compression.pdf • Extending IAMCTD using data fusion and lossless data compression for UWSN.pdf • Failure Recovery Cost Reduction of Disk Arrays Using Adaptive Erasure Correction Coding and Data Compression.pdf • FuzzyCAT A lightweight Adaptive Transform for sensor data compression.pdf • Generalized Context Modeling With Multi-Directional Structuring and MDL-Based Model Selection for Heterogeneous Data Compression.pdf • Haptic data compression based on delta-sigma modulator in quantized scaling-bilateral control.pdf • Hardware and software architectures for computationally efficient three-dimensional ultrasonic data compression.pdf • Image watermarking using data compression.pdf • IoT Data Compression Sensor-Agnostic Approach.pdf • Isometric Test Data Compression.pdf • JICE Joint data compression and encryption for wireless energy auditing networks.pdf • K-Means Clustering-Based Data Compression Scheme for Wireless Imaging Sensor Networks.pdf • KDI A wireless ECoG recording platform with impedance spectroscopy, electrical stimulation and real-time, lossless data compression.pdf • Local recovery in data compression for general sources.pdf • Lossless Data Compression via Substring Enumeration for k-th Order Markov Sources with a Finite Alphabet.pdf • Method of data compression for traffic monitoring.pdf • NGS read data compression using parallel computing algorithm.pdf • On EEG lossy data compression for data-intensive neurological mobile health solutions.pdf • On-the-fly range-velocity radar data compression exploring Doppler spectrum redundancy.pdf • Onboard SPIHT-based and textural-dependent data compression of one-look SAR images.pdf • Performance evaluation of mobile front-haul employing ethernet- based TDM-PON with IQ data compression Invited.pdf • Power analysis of a lossless data compression technique for wireless wearable biometric devices.pdf

	• Predictive Principal Component Analysis as a Data Compression Core in a Simulation Data Management System.pdf • Quadtree-based lightweight data compression for large-scale geospatial rasters on multi-core CPUs.pdf • Real-time channel data compression for improved software beamforming using microbeamforming with error compensation.pdf • Secure DNA data compression using algebraic curves.pdf • Thermal-Aware Test Data Compression Using Dictionary Based Coding.pdf • Traffic prediction, data compression, abnormal data detection and missing data imputation An integrated study based on the decomposition of traffic time series.pdf • Variable frame rate and length analysis for data compression in distributed speech recognition.pdf • Visual Hull-Based Geometric Data Compression of a 3-D Object.pdf • Work in progress Data compression of wireless sensor network employing Kalman filter and QC-LDPC codes.pdf
Information Extraction	• A rule-based information extraction system for human-readable semi-structured scientific documents.pdf • A SAR imaging technique for the target of complex azimuth envelope based on information extraction.pdf • A survey of temporal information extraction and language independent features.pdf • A system for semantic information extraction from mixed soundtracks deploying MARSYAS framework.pdf • A systematic approach to automate spiddos information extraction for on-web GP system.pdf • A webpage information extraction method based on game theory.pdf • Acquisition of Generic Problem Solving Knowledge through Information Extraction and Pattern Mining.pdf • Adaptive information extraction of disaster information from Twitter.pdf • Algorithm Analysis and System Implementation of Advanced Facial Information Extraction and Face Recognition under Wireless Networks.pdf • An alternative algorithm for wave information extraction from X-band nautical radar images.pdf • An automated stock recommendation system from stock investment research using domain specific information extraction.pdf

	• An automatic solution of accessible information extraction from CityGMLLoD4 files.pdf • An effective real-time fingertip positioning system based on gradient information extraction from frame image sequences.pdf • An Improved Ontology-Based Web Information Extraction.pdf • An information extraction framework for legal documents A case study of Thai Supreme Court verdicts.pdf • An information extraction system for protein function prediction.pdf • An Information-Extraction Approach to Speech Processing Analysis, Detection, Verification, and Recognition.pdf • An intelligent personalized traffic information extraction system for road traffic safety.pdf • An intelligent system for information extraction from relational database using HMM.pdf • An Ontology-Based Information Extraction System for bridging the configuration gap in hybrid SDN environments.pdf • Application of Image Classification Technology in Mangrove Information Extraction.pdf • Application of Internet Technology and Web Information Extraction Wrapper Based on DOM for Agricultural Data Acquisition.pdf • Automated Extraction Information System from HUDs Images Using ANN.pdf • Automated information extraction from free-text EEG reports.pdf • Automated Road Information Extraction From Mobile Laser Scanning Data.pdf • Automatic and Adaptive Clusters for Information Extraction.pdf • Automatic Bug Assignment Using Information Extraction Methods.pdf • Automatic building information extraction by modified volumetric shadow analysis from high resolution multispectral data.pdf • Automatic information extraction from heatmaps.pdf • BioHCDP A Hybrid Constituency-Dependency Parser for Biological NLP information extraction.pdf • Characteristic information extraction and developing process recognizing method of surface discharge in oil immersed paper insulation.pdf

	• Color space transformation and object oriented based information extraction of aerial images.pdf • Content Information Extraction of Theme Web Pages Based on Tag Information.pdf • Crop information extraction in China based on NDVI characteristic curve.pdf • Decision Guidance for Optimizing Web Data Quality - A Recommendation Model for Completing Information Extraction Results.pdf • Document Information Extraction and Its Evaluation Based on Client's Relevance.pdf • Drug-related crime information extraction and analysis.pdf • Efficient and anti-interference method of synchronising information extraction for cideo leaking signal.pdf • Error Analysis in an Automated Narrative Information Extraction Pipeline.pdf • Evaluation on geospatial information extraction and retrieval Mining thematic maps from web source.pdf • Event information extraction from Indonesian tweets using conditional random field.pdf • Extending Spark Analytics through Tika-Based Information Extraction and Retrieval.pdf • Floating raft aquaculture information automatic extraction based on high resolution SAR images.pdf • From text to XML by structural information extraction.pdf • From tweet to graph Social network analysis for semantic information extraction.pdf • Ge(o)Lo(cator) Geographic Information Extraction from Unstructured Text Data and Web Documents.pdf • Generating Rules with Common Knowledge A Framework for Sentence Information Extraction.pdf • Growing neural gas for information extraction in gesture recognition and reproduction of robot partners.pdf • Handwritten Information Extraction from Historical Census Documents.pdf • Implicit Information Extraction from Clinical Notes.pdf
Image Processing	• A brief study on paddy applications with image processing and proposed architecture.pdf • A GPU-Based Processing Chain for Linearly Unmixing Hyperspectral Images.pdf • A Guided Tour of Selected Image Processing and Analysis Methods for Fluorescence and Electron Microscopy.pdf • A Hierarchical Distributed Processing Framework for Big Image Data.pdf

	• A Natural-Scene Gradient Distribution Prior and its Application in Light-Microscopy Image Processing.pdf • A New Automatic Testing System Based on Image Processing and Microprobes for IC-Testing.pdf • A New Experimental Approach Using Image Processing-Based Tracking for an Efficient Fault Diagnosis in Pantograph-Catenary Systems.pdf • A Novel Fractional-Order Differentiation Model for Low-Dose CT Image Processing.pdf • A Novel Method Based on Digital Image Processing Technique and Finite Element Method for Rapidly Modeling Optical Properties of Actual Microstructured Optical Fibers.pdf • A novel shape and boundary extraction image processing technique for detecting the breast abnormalities using digital mammogram - labview implementation.pdf • A smart phone image processing application for plant disease diagnosis.pdf • A software demonstrator for cognitive image processing using the Associative Memory chip.pdf • A Unified Geometric Model for Virtual Slide Image Processing and Classification.pdf • Adaptive processing of image using DWT and FFT OFDM in AWGN and Rayleigh channel.pdf • Algorithmic Enhancements to Big Data Computing Frameworks for Medical Image Processing.pdf • An image processing technique combined with Real time data acquisition to identify the posture imbalance in normal subjects.pdf • An innovative practice in a graduate course combining startups evaluation and image processing.pdf • Application of discrete wavelet transform in thermal infrared image processing.pdf • Assessment of quality of rice grain using optical and image processing technique.pdf • Autonomous walking stick for the blind using echolocation and image processing.pdf • Cloud Engineering Principles and Technology Enablers for Medical Image Processing-as-a-Service.pdf • Computational complexity of image processing algorithms for an intelligent mobile enabled tongue diagnosis scheme.pdf • Computer-Aided Diagnosis Software for Hypertensive Risk Determination Through Fundus Image Processing.pdf • Detection and counting of pothole using image processing techniques.pdf

	• Development and implementation of image processing technique for laser-based 3D scanner.pdf • Distance dependent Chinese restaurant process for VHR satellite image oversegmentation.pdf • End-View Image Processing Based Angle Alignment Techniques for Specialty Optical Fibers.pdf • Extraction of Energy Information From Analog Meters Using Image Processing.pdf • Fractional-Order Euler-Lagrange Equation for Fractional-Order Variational Method A Necessary Condition for Fractional-Order Fixed Boundary Optimization Problems in Signal Processing and Image Processing.pdf • Image processing algorithm for droplet measurement after impingement.pdf • Image processing based detection - classification of blood group using color images.pdf • Image Processing With Color Compensation Using LCD Display for Color Vision Deficiency.pdf • Investigation on large batches of UAV image data processing technology based on acceleration matching method.pdf • Medical image processing A review.pdf • Monitoring Kidney Microanatomy Changes During Ischemia-Reperfusion Process Using Texture Analysis of OCT Images.pdf • Multi-Zone Digital Crosstalk Reduction by Image Processing in 3D Display.pdf • PCB Reverse Engineering Using Nondestructive X-ray Tomography and Advanced Image Processing.pdf • Plant diseases detection using image processing techniques.pdf • Processing and 3D printing of Gradient Heterogeneous Bio-Model Based on Computer Tomography Images.pdf • Processing Sliding Mosaic Mode Data With Modified Full-Aperture Imaging Algorithm Integrating Scalloping Correction.pdf • Real time two way communication approach for hearing impaired and dumb person based on image processing.pdf • Recognizing matured cinnamon tree using image processing techniques.pdf • Recurrently Decomposable 2-D Convolvers for FPGA-Based Digital Image Processing.pdf • Satellite image processing on parallel computing A technical review.pdf • Spatio-Temporal Bias-Tunable Readout Circuit for On-Chip Intelligent Image Processing.pdf

	• Structural Variance Based Error-Tolerability Test Method for Image Processing Applications.pdf • Teaching Image Processing in Engineering Using Python.pdf • Towards an efficient high-level modeling of heterogeneous image processing systems.pdf • Trigger-Wave Asynchronous Cellular Logic Array for Fast Binary Image Processing.pdf • Wavelet-Based Classification of Hyperspectral Images Using Extended Morphological Profiles on Graphics Processing Units.pdf
Semantic Search	• Query-Based Ontology Approach for Semantic Search.pdf • Real-Time Semantic Search Using Approximate Methodology for Large-Scale Storage Systems.pdf • Recent Advances and Challenges of Semantic Image Video Search.pdf • Relational WordNet model for semantic search in Holy Quran.pdf • Research and Analysis of Semantic Search Technology Based on Knowledge Graph.pdf • Research and Design of E-commerce Semantic Search.pdf • Research of semantic search algorithm based on Cloud computing.pdf • Research on Improving Performance of Semantic Search in UDDI.pdf • S2P2P Semantic Search in Unstructured Peer-to-Peer Networks.pdf • S3C An architecture for space-efficient semantic search over encrypted data in the cloud.pdf • Scalable Semantic Search with Hybrid Concept Index over Structure Peer-to-Peer Network.pdf • Scaling concurrency of personalized Semantic search over Large RDF data.pdf • SDWS Semantic Search for Deep Web Data.pdf • Searching for semantic person queries using channel representations.pdf • Semantic association search and rank method based on spreading activation for the Semantic Web.pdf • Semantic Computing, Cloud Computing, and Semantic Search Engine.pdf • Semantic Entity Search Diversification.pdf • Semantic flooding Search over semantic links.pdf • Semantic form-based guided search system.pdf • Semantic Full-Text Search with ESTER Scalable, Easy, Fast.pdf • Semantic Key for Meaning Based Searching.pdf

	• Semantic overlay network for peer-to-peer hybrid information search and retrieval.pdf • Semantic report search engine — Questor.pdf • Semantic Search and NLP-Based Diagnostics.pdf • Semantic Search and Preferred Search Survey in Cloud Computing Environment.pdf • Semantic Search Based on SVO Constructions in Chinese.pdf • Semantic search engine and its strategies with IAN encoder.pdf • Semantic search for cloud providers with security conformance to cloud controls matrix.pdf • Semantic search for context-aware learning.pdf • Semantic search for software architecture knowledge A proposal for virtual communities environment.pdf • Semantic search for structured Web finder.pdf • Semantic search in digital library semantic technology.pdf • Semantic search integration to climate data.pdf • Semantic Search Meets the Web.pdf • Semantic search of learning services in a grid-based collaborative system.pdf • Semantic Search over DICOM Repositories.pdf • Semantic search over encrypted data.pdf • Semantic search processing in natural language interface.pdf • Semantic Search Technology Based on Cloud Computing Research.pdf • Semantic search using a similarity graph.pdf • Semantic search with rule reasoning for scholarship information search.pdf • Semantic Search-Based Genetic Programming and the Effect of Intron Deletion.pdf • Semantic TagPrint - Tagging and Indexing Content for Semantic Search and Content Management.pdf • Semantic WEB-Search Developing by Problem-Oriented Ontology Means.pdf • Semantic-driven Management and Search for Resources in Research Community.pdf • Semi-automated Ontology Creation for Semantic Search in Business Process Exploration.pdf • Semi-Automatic Annotation System for OWL-Based Semantic Search.pdf • Service-Based Semantic Search in P2P Systems.pdf • SKMT A Semantic Knowledge Management Tool for Content Tagging, Search and Management.pdf • Smart cloud search services verifiable keyword-based semantic search over encrypted cloud data.pdf

Word Disambiguation	• A cognitively motivated word sense induction algorithm.pdf • A comprehensive analysis of using semantic information in text categorization.pdf • A new method for solving context ambiguities using field association knowledge.pdf • A preliminary study on semi-automatic construction of sense tagged corpus with Wordnet senses using semantic vector.pdf • A semantic-based text classification system.pdf • Active Learning With Sampling by Uncertainty and Density for Data Annotations.pdf • An automated semantic annotation based-on Wordnet ontology.pdf • An Enhanced Bag-of-Visual Word Vector Space Model to Represent Visual Content in Athletics Images.pdf • An interpretation of sentiment analysis for enrichment of Business Intelligence.pdf • An IR-Based Approach Utilizing Query Expansion for Plagiarism Detection in MEDLINE.pdf • Calculating Word Sense Probability Distributions for Semantic Web Applications.pdf • Combining Neural Networks and Statistics for Chinese Word Sense Discrimination.pdf • Construction of Myanmar WordNet lexical database.pdf • Construction of Scholarly n-Gram from Huge Text Data.pdf • DAVE Detecting Agitated Vocal Events.pdf • Disambiguating sentiment ambiguous adjectives.pdf • Disambiguation for polyphones of Chinese based on two-pass unified approach.pdf • Exploiting Disambiguation and Discrimination in Information Retrieval Systems.pdf • Exploring multiple features for sense prediction of Chinese unknown words.pdf • Feature level sentiment analysis on movie reviews.pdf • Framework to extract context vectors from unstructured data using big data analytics.pdf • Generating word clusters by graph clustering based on hearst patterns.pdf • Graph-Based Methods for Natural Language Processing and Understanding—A Survey and Analysis.pdf • Grapheme-to-Phoneme Conversion of Arabic Numeral Expressions for Embedded TTS Systems.pdf • Improving Persian POS tagging using the maximum entropy model.pdf

	• Improving the Understanding between Control Tower Operator and Pilot Using Semantic Techniques — A New Approach.pdf • Improving Verb Sense Disambiguation with Automatically Retrieved Semantic Knowledge.pdf • Issues in parsing for Machine Aided Translation from English to Hindi.pdf • Keyword Extraction Based on Lexical Chains and Word Co-occurrence for Chinese News Web Pages.pdf • Linking Documents to Encyclopedic Knowledge.pdf • Measuring context-meaning for open class words in Hindi language.pdf • Myanmar to English verb translation disambiguation approach based on Naïve Bayesian classifier.pdf • One Sense per N-gram.pdf • Ontological based webpage classification.pdf • Ontology clarification by using semantic disambiguation.pdf • Optimal character arrangement for ambiguous keyboards using a PSO-based algorithm.pdf • POC-NLW Template Based Tagging Method for Chinese Word Segmentation.pdf • Query Expansion for UMLS Metathesaurus Disambiguation Based on Automatic Corpus Extraction.pdf • Rewriting Turkish texts written in English alphabet using Turkish alphabet.pdf • Rule Based Part of Speech Tagging of Sindhi Language.pdf • Semantic disambiguation in a social information discovery system.pdf • Semi-supervised Chinese contextual polarity classification with automatic feature selection.pdf • Sense Annotated Hindi Corpus.pdf • Tag Sense Disambiguation for Clarifying the Vocabulary of Social Tags.pdf • TaxoLearn A Semantic Approach to Domain Taxonomy Learning.pdf • Unsupervised Translation Disambiguation Based on Maximum Web Bilingual Relatedness Web as Lexicon.pdf • Using triangulation to identify word senses.pdf • Word Sense Determination using WordNet and Sense Co-occurrence.pdf • Word sentiment polarity disambiguation based on opinion level context.pdf • Word-Meaning Selection in Multiprocess Language Understanding Programs.pdf

B. Sets of Queries

Table B-1 List of Queries for Set A

Category	ID	Queries
Word disambiguation	Q1	determine the proper sense of an ambiguous word in a given context
	Q2	the task of determining the correct sense of a word in context
	Q3	the correct meaning of a word
	Q4	natural classification problem
	Q5	Word-sense induction
Data compression	Q1	the process of reducing the amount of data needed for the storage or transmission of a given piece of information
	Q2	compressing data into smaller files
	Q3	manipulating order of data
	Q4	extraction of data
	Q5	computer programming
Image processing	Q1	the analysis and manipulation of a digitized image, especially in order to improve its quality
	Q2	processing images or pictures
	Q3	changing the picture's data
	Q4	image manipulation
	Q5	getiing data from music
Information extraction	Q1	the process of turning the unstructured information embedded in texts into structured data
	Q2	automatically extract structured information
	Q3	extract fields of interest from free text
	Q4	text simplification technique
	Q5	data gathering
Semantic search	Q1	a data searching technique in which a search query aims to not only find keywords, but to determine the intent and the context of the search
	Q2	information searching with context and intent
	Q3	data searching context
	Q4	searching technique
	Q5	microsoft windows

Table B-2 List of Queries for Set B

Category	ID	Queries
Word disambiguation	Q1	determine the proper sense of an ambiguous word in a given context
	Q2	the task of determining the correct sense of a word in context
	Q3	the correct meaning of a word
	Q4	natural classification problem
	Q5	word meaning induction
Data compression	Q1	the process of reducing the amount of data needed for the storage or transmission of a given piece of information
	Q2	compressing data into smaller files
	Q3	manipulating order of data
	Q4	extraction of data
	Q5	computer programming
Image processing	Q1	the analysis and manipulation of a digitized image, especially in order to improve its quality
	Q2	processing images or pictures
	Q3	changing the picture's data
	Q4	image manipulation
	Q5	getiing data from music
Information extraction	Q1	the process of turning the unstructured information embedded in texts into structured data
	Q2	automatically extract structured information
	Q3	extract fields of interest from free text
	Q4	text simplification technique
	Q5	data gathering
Semantic search	Q1	a data searching technique in which a search query aims to not only find keywords, but to determine the intent and the context of the search
	Q2	information searching with context and intent
	Q3	data searching context
	Q4	searching technique
	Q5	Windows operating system

C. Sample Document (IEEE Online Database)

Sample Document: **Generating Rules with Common Knowledge A Framework for Sentence Information Extraction.pdf**

Generating Rules with Common Knowledge: A Framework for Sentence Information Extraction.

There are many nature language processing applications. A typical example is information extraction whose target is a sentence. Various rules are often used in this kind of applications. However, automated processing is not accurate enough in some cases. This is because it is easy to construct syntax rules of a sentence but difficult to semantic rules. On the other hand, the knowledge representation community paid much attention to common knowledge. It is insightful to use rules based on this sense on common things in nature language processing. Therefore, we propose an approach to combine the common knowledge and the nature language processing rules. It first applied the name entity reorganization technology and then generated rules based on a specific common knowledge database. As a result, this approach can be a framework for many (but not all) nature language processing applications. In our experimental example, this approach performed well.

D. Sample Concept (Wikipedia Database)

Sample Concept: **Pacificair.pdf**

Pacificair

From Wikipedia, the free encyclopedia

Pacific Airways Corporation operating as **Pacificair** is a charter airline based in Manila in the Philippines. It operates scheduled passenger flights, as well as air taxi services, and agricultural work. Its main base is Ninoy Aquino International Airport, Manila.[1]

Pacificair

IATA GX	ICAO -	Callsign -
Founded	1947	
Hubs	Ninoy Aquino International Airport	
Fleet size	4	
Headquarters	Manila, Philippines	
Website	http://www.pacificair.com.ph/home.html	

Contents

History

The airline was established in 1947. However, its franchise to operate air services was approved by Congress of the Philippines on February 23, 1995, through Republic Act No. 7909.

Incidents and accidents

On 2 April 1996, de Havilland Canada DHC-6 Twin Otter 200 RP-C1154 taxied across Manila International Airport runway 13 at the Taxiway F1 intersection. At the same moment Boeing 737 (EI-BZF) was taking off and collided with the Twin Otter. The 737 carried the Twin Otter for 130 metres (430 ft) before coming to a halt. The aircraft was taxiing to the Pacific Airways hangar after passenger disembarkation at the General Aviation ramp.[2]

On 9 June 1999, Britten-Norman BN-2A-21 Islander RP-C471 crashed near Coron Airport and was written off. There was two crew fatality (pilot: Angelito V. Ruiz, SR. & Co-pilot J. Cristobal).[3]

On 16 October 2004, Britten-Norman BN-2A-21 Islander RP-C1325, bound from Coron Airport to Puerto Princesa Airport with 700 kilos of live fish, flew into the side of Mount Tagpao. Weather was poor with heavy rains. There were two crew fatalities. The aircraft was written off.[4]

Services

Flightseeing tours are available to Boracay, Bohol, Calauit, Camiguin, Bicol, Banaue, Palawan, Davao and Corregidor.

Fleet

The Pacificair fleet includes the following aircraft (at March 2007):[1]

- 4 Britten-Norman BN2A Islander

References

1. "Directory: World Airlines". *Flight International*. 2007-04-10. p. 61.
2. *Accident description* (http://aviation-safety.net/database/record.php?id=19960402-1), Aviation Safety network
3. *Accident description* (http://aviation-safety.net/database/record.php?id=19990609-5), Aviation Safety network
4. *Accident description* (http://aviation-safety.net/database/record.php?id=20041016-0), Aviation Safety network

External links

- Pacific Airways Online

E. Source Code of Some Implemented Algorithms and Modules

Creation Phase

A. ConceptSpacesModel.java


```
package model;

import java.io.IOException;
import java.io.PrintWriter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.HashMap;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.store.FSDirectory;

import utilities.TFIDF;
import utilities.TermTFIDF;
import utilities.luceneIndex;
import utilities.luceneSearch;

public class ConceptSpacesModel {
    private HashMap<Integer, ArrayList<TermTFIDF>> docTFIDF;
    private ArrayList<String> docPath, conceptPath;
    private int alpha;
    private int[] conceptFreq;
    private String conceptIndexPath, docsIndexPath, conceptSpaceIndexPath, logMessage;
    private IndexReader docsReader, conceptsReader;
    private StringBuilder str;

    public ConceptSpacesModel(int numConcepts) {
        docTFIDF = new HashMap<Integer, ArrayList<TermTFIDF>>();
        docPath = new ArrayList<String>();
        conceptPath = new ArrayList<String>();
        conceptFreq = new int[numConcepts];
        logMessage = "";
        str = new StringBuilder("");
    }

    public HashMap<Integer, ArrayList<TermTFIDF>> getDocTFIDF() {
        return docTFIDF;
    }

    public void setDocTFIDF(HashMap<Integer, ArrayList<TermTFIDF>> docTFIDF) {
        this.docTFIDF = docTFIDF;
    }

    public int getAlpha() {
        return alpha;
    }

    public void setAlpha(int alpha) {
        this.alpha = alpha;
    }

    public String getConceptIndexPath() {
        return conceptIndexPath;
    }

    public void setConceptIndexPath(String conceptIndexPath) {
        this.conceptIndexPath = conceptIndexPath;
    }

    public String getDocsIndexPath() {
        return docsIndexPath;
    }

    public void setDocsIndexPath(String docsIndexPath) {
```

```
        this.docsIndexPath = docsIndexPath;
    }

    public String getConceptSpaceIndexPath() {
        return conceptSpaceIndexPath;
    }

    public void setConceptSpaceIndexPath(String conceptSpaceIndexPath) {
        this.conceptSpaceIndexPath = conceptSpaceIndexPath;
    }

    public ArrayList<String> getDocPath() {
        return docPath;
    }

    public ArrayList<String> getConceptPath() {
        return conceptPath;
    }

    public int[] getConceptFreq() {
        return conceptFreq;
    }

    public String getLogMessage() {
        return logMessage;
    }

    public void setReaders() throws IOException {
        docsReader = DirectoryReader.open(FSDirectory.open(Paths.get(docsIndexPath)));
        conceptsReader = DirectoryReader.open(FSDirectory.open(Paths.get(conceptIndexPath)));
    }

    public void closeReaders() throws IOException {
        docsReader.close();
        conceptsReader.close();
    }

    public void conceptSpaceConstruction() throws Exception {
        setReaders();
        System.out.println("Constructing Concept Space...");
        getDCF();
        getTopDCF();
        indexConceptSpace();
        int numDocs = docsReader.numDocs();
        for (int i = 0; i < numDocs; i++) {
            Document doc = docsReader.document(i);
            String path = doc.get("path");
            docPath.add(path);
        }
        closeReaders();
        logMessage = str.toString();
    }

    public void getDCF() throws Exception {
        IndexSearcher conceptsSearcher = new IndexSearcher(conceptsReader);
        IndexSearcher docsSearcher = new IndexSearcher(docsReader);
        Analyzer analyzer = new StandardAnalyzer();
        QueryParser parser = new QueryParser("words", analyzer);
        int numDocs = docsReader.numDocs();
        int numConcepts = conceptsReader.numDocs();
        for (int i = 0; i < numDocs; i++) {
            Document doc = docsReader.document(i);
            System.out.println("Processing " + doc.get("path"));
            str.append("\r\n");
            str.append("Processing " + doc.get("path") + "\r\n");

            str.append("computing TFIDF value...\r\n");
            TFIDF tfidf = new TFIDF();
            tfidf.setDocId(i);
            tfidf.setAlpha(alpha);
            tfidf.setReader(docsReader);
```

```
            tfidf.setSearcher(docsSearcher);
            tfidf.setStr(str);
            tfidf.computeTFIDF();
            String line = tfidf.getHeavyTerms();
            str = tfidf.getStr();
            str.append("Heavy Weighted Terms: "+ line + "\r\n");
            docTFIDF.put(i, tfidf.getTermsTFIDF());

            Query query = parser.parse(line);
            luceneSearch ls = new luceneSearch();
            ls.setNumDocs(numConcepts);
            ls.setSearcher(conceptsSearcher);
            ls.setConceptFreq(conceptFreq);
            ls.setQuery(query);
            ls.search();
            conceptFreq = ls.getConceptFreq();
        }
        str.append("Document Concept Frequency (DCF): \r\n");
        for (int i = 0; i < conceptFreq.length; i++){
            Document doc = conceptsReader.document(i);
            Path path = Paths.get(doc.get("path"));
            String fileName = path.getFileName().toString();
            double dcf = conceptFreq[i] / (double) numDocs;
            str.append(fileName + ": " + conceptFreq[i] + "/" + numDocs + " = " + dcf +
            "\r\n");
        }
    }

    public void getTopDCF() throws IOException {
        System.out.println("Selecting top 100 Concepts...");
        str.append("Selecting top 100 Concepts...\r\n");
        double dcf = 0;
        ArrayList<Double> conceptDCF = new ArrayList<Double>();
        for (int i = 0; i < conceptFreq.length; i++) {
            dcf = conceptFreq[i] / (double) conceptFreq.length;
            Document concept = conceptsReader.document(i);
            String path = concept.get("path");
            if (conceptDCF.isEmpty()) {
                conceptDCF.add(dcf);
                conceptPath.add(path);
            } else if (conceptDCF.get(conceptDCF.size() - 1) > dcf) {
                conceptDCF.add(dcf);
                conceptPath.add(path);
            } else {
                for (int j = 0; j < conceptDCF.size(); j++) {
                    if (conceptDCF.get(j) <= dcf) {
                        conceptDCF.add(j, dcf);
                        conceptPath.add(j, path);
                        break;
                    }
                }
            }
        }
        conceptPath.subList(100, conceptPath.size()).clear();
        str.append("Concept Space: \r\n");
        for(int i = 0; i < conceptPath.size(); i++){
            str.append(conceptPath.get(i) + "\r\n");
         }
    }

    public void indexConceptSpace() {
        luceneIndex li = new luceneIndex();
        li.setIndexPath(conceptSpaceIndexPath);
        li.setCreate(true);
        li.setDocIdCounter(0);
        for (int i = 0; i < conceptPath.size(); i++) {
            li.setDocsPath(conceptPath.get(i));
            li.indexFiles();
            li.setCreate(false);
            str.append(li.getLogMessage() + "\r\n");
        }
```

```
    }


    public static void main(String[] args) throws Exception {
         double table[][] = new double[900000][200];
         for(int i = 0; i < table.length; i++) {
             for(int j = 0; j < table[i].length;j++) {
                 table[i][j] = 9;
             }
         }
    }
}
```

B. ConceptVector.java

C:\Users\jmags\Downloads\java\creation phase\ConceptVector.java Tuesday, 5 December 2017 4:35 AM

```
package model;

import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;

import edu.mit.jwi.Dictionary;
import edu.mit.jwi.IDictionary;
import utilities.PDFtoString;
import utilities.Remover;
import utilities.Window;
import utilities.Stemmer;
import utilities.VectorTFIDF;

public class ConceptVector {
    private List<String> ConceptSpace;
    private List<List<String>> cSpace;
    private double[][] TFIDFTable;
    private List<Window> Terms;
    private String wordNetPath, dict, path;
    private IDictionary dictionary;
    private URL url;
    private String logMessage;

    public List<String> getConceptSpace() {
        return ConceptSpace;
    }

    public void setConceptSpace(List<String> conceptSpace) {
        ConceptSpace = conceptSpace;
    }

    public double[][] getTFIDFTable() {
        return TFIDFTable;
    }

    public void setTFIDFTable(double[][] tFIDFTable) {
        TFIDFTable = tFIDFTable;
    }

    public List<Window> getTerms() {
        return Terms;
    }

    public void setTerms(List<Window> terms) {
        Terms = terms;
    }

    public String getLogMessage() {
        return logMessage;
    }

    public void setLogMessage(String logMessage) {
        this.logMessage = logMessage;
    }

    public ConceptVector(String wordNetPath, List<String> ConceptSpace) {
        this.ConceptSpace = ConceptSpace;
        this.wordNetPath = wordNetPath;
        this.dict = "dict";
        this.path = this.wordNetPath + File.separator + this.dict;
        try {
            this.url = new URL("File", null, this.path);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        dictionary = new Dictionary(this.url);
```

```
        try {
            dictionary.open();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            cSpace = convertToPDF();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public List<List<String>> convertToPDF() throws IOException {
        System.out.println("Conversion of PDF to clean tokens");
        List<List<String>> ListHolder = new ArrayList<List<String>>();
        for (String concept : ConceptSpace) {
            String conceptHolder = PDFtoString.extractPdfText(concept);
            String noSymbolConcept = Remover.removeSymbols(conceptHolder);
            String cleanConcept = Stemmer.stemIt(noSymbolConcept);
            StringTokenizer cleanConceptTokens = new StringTokenizer(cleanConcept);
            List<String> tokens = new ArrayList<String>();
            while (cleanConceptTokens.hasMoreElements()) {
                tokens.add((String) cleanConceptTokens.nextElement());
            }

            ListHolder.add(tokens);
        }
        System.out.println("Process done.");
        return ListHolder;
    }

    public List<Window> mergeSortWindow(List<Window> cWindow) {
        List<Window> sortedWindow;
        if (cWindow.size() == 1) {
            sortedWindow = cWindow;
        } else {
            int mid = cWindow.size() / 2;

            List<Window> leftSide = new ArrayList<Window>();
            List<Window> rightSide = new ArrayList<Window>();

            for (int i = 0; i < mid; i++) {
                leftSide.add(cWindow.get(i));

            }
            for (int i = mid; i < cWindow.size(); i++) {
                rightSide.add(cWindow.get(i));
            }

            leftSide = mergeSortWindow(leftSide);
            rightSide = mergeSortWindow(rightSide);
            sortedWindow = mergeLeftRight(leftSide, rightSide);
        }

        return sortedWindow;
    }

    public List<Window> mergeLeftRight(List<Window> left, List<Window> right) {
        List<Window> merged = new ArrayList<Window>();

        int l = 0;
        int r = 0;

        while (l < left.size() && r < right.size()) {
            if
            ((left.get(l).getTargetTerm()).compareToIgnoreCase(right.get(r).getTargetTerm())
            < 0) {
                merged.add(left.get(l));
                l++;
            } else {
                merged.add(right.get(r));
```

```
                r++;
            }
        }

        while (l < left.size()) {
            merged.add(left.get(l));
            l++;
        }
        while (r < right.size()) {
            merged.add(right.get(r));
            r++;
        }

        return merged;
    }
    public static double[] summary(int start, int end, double[][] matrix) {
        double[] newRow = new double[matrix[0].length];
        for (int ci = 0; ci < matrix[0].length; ci++) {
            double max = -9999;
            for (int i = start; i < end; i++) {
                if (max < matrix[i][ci]) {
                    max = matrix[i][ci];
                }
                newRow[ci] = max;
            }
        }

        return newRow;
    }

    public double getAverage(List<Double> weights) {
        double sum = 0;
        double size = weights.size();
        double average;

        for (double weight : weights) {
            sum += weight;
        }

        average = sum / size;

        return average;
    }

    public double[][][] createVector() {
        StringBuilder log = new StringBuilder("");
        VectorTFIDF v = new VectorTFIDF();
        List<Window> sortedTerms = mergeSortWindow(Terms);
        System.out.println("List<Window> Terms sorted.");
        double[][] conceptSpaceMatrix = new double[sortedTerms.size()][cSpace.size()];

        int termInc = 0;
        int conceptInc = 0;
        int remaining = sortedTerms.size() * cSpace.size();
        System.out.println("Starting double[][] ConceptSpaceMatrix Computation. Dimension:["
        + sortedTerms.size() + "]["
                + cSpace.size() + "] number of computations: " + remaining);
        for (List<String> concept : cSpace) {
            for (Window sTerms : sortedTerms) {
                List<Double> scores = new ArrayList<Double>();
                remaining--;
                double aveScore = 0;
                if(sTerms.getSurroundingTerms().length == 0) {
                    conceptSpaceMatrix[termInc][conceptInc] = aveScore;
                }else{
                    for (String term : sTerms.getSurroundingTerms()) {
                        double score = 0;
                        score = v.TFIDFCompute(concept, cSpace, term);
                        scores.add(score);
                    }
                    aveScore = getAverage(scores);
```

```
                    conceptSpaceMatrix[termInc][conceptInc] = aveScore;
                }
                System.out.println("Average TFIDF Score of surrounding terms: " + aveScore);
                System.out.println("remaining computations: " + remaining);
                termInc++;
            }
            termInc = 0;
            conceptInc++;
        }
        System.out.println("double[][] ConceptSpaceMatrix computed.");

        double[][] nConceptSpaceMatrix = new
        double[conceptSpaceMatrix.length][conceptSpaceMatrix[0].length];
        remaining = sortedTerms.size();
        System.out
                .println("Starting double[][] nConceptSpaceMatrix Computation. Dimension:["
                + conceptSpaceMatrix.length
                        + "][" + conceptSpaceMatrix[0].length + "] number of computations: "
                        + remaining);
        int inc = 0;
        for (int wi = 0; wi < sortedTerms.size(); wi++) {
            remaining--;
            int k = wi + 1;
            while (k < sortedTerms.size()
                    &&
                    sortedTerms.get(k).getTargetTerm().equalsIgnoreCase(sortedTerms.get(wi).ge
                    tTargetTerm())) {
                k++;
            }
            if (wi == sortedTerms.size() - 1) {
                nConceptSpaceMatrix[inc] = conceptSpaceMatrix[wi];
            } else if (k == (wi + 1)) {
                if
                (sortedTerms.get(k).getTargetTerm().equalsIgnoreCase(sortedTerms.get(wi).getTa
                rgetTerm())) {
                    double[] newRow = summary(wi, k, conceptSpaceMatrix);
                    nConceptSpaceMatrix[inc] = newRow;
                } else {
                    nConceptSpaceMatrix[inc] = conceptSpaceMatrix[wi];
                }
            } else {
                double[] newRow = summary(wi, k, conceptSpaceMatrix);
                nConceptSpaceMatrix[inc] = newRow;
            }
            inc++;
            System.out.println("remaining computations: " + remaining);
        }
        System.out.println("double[][] nConceptSpaceMatrix computed.");

        double[][][] Vector = new
        double[TFIDFTable[0].length][TFIDFTable.length][cSpace.size()];
        remaining = TFIDFTable[0].length * TFIDFTable.length * cSpace.size();
        System.out.println("Starting double[][][] Vector Computation. Dimension:[" +
        TFIDFTable[0].length + "]["
                + TFIDFTable.length + "][" + cSpace.size() + "] number of computations: " +
                remaining);

        for (int di = 0; di < TFIDFTable[0].length; di++) {
            for (int ci = 0; ci < cSpace.size(); ci++) {
                for (int ti = 0; ti < TFIDFTable.length; ti++) {
                    remaining--;
                    if(TFIDFTable[ti][di] == 0 || nConceptSpaceMatrix[ti][ci] == 0) {
                        Vector[di][ti][ci] = 0;
                    }
                    else {
                        Vector[di][ti][ci] = TFIDFTable[ti][di] + nConceptSpaceMatrix[ti][ci];
                    }
                    System.out.println("Vector[" + di + "][" + ti + "][" + ci + "] score: "
                    + Vector[di][ti][ci]);
                    System.out.println("remaining computations: " + remaining);
                }
```

```
            }
        }
        System.out.println("double[][][] Vector computed.");
        System.out.println("Concept Vector Process Finished.");
        return Vector;
    }
}
```

C. ConceptWindow.java


```
package model;

import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Scanner;

import edu.mit.jwi.Dictionary;
import edu.mit.jwi.IDictionary;
import edu.mit.jwi.item.IIndexWord;
import edu.mit.jwi.item.ISynset;
import edu.mit.jwi.item.IWord;
import edu.mit.jwi.item.IWordID;
import edu.mit.jwi.item.POS;
import utilities.PDFtoString;
import utilities.Remover;
import utilities.Stemmer;
import utilities.Window;

public class ConceptWindow {
    private String wordNetPath;
    private String path;
    private URL url;
    public static IDictionary dictionary;
    private List<Window> targetTerms;
    private List<Window> uniqueTargetTerms;
    private HashSet<String> uniqueTargetTermsIdentifier;
    private HashSet<String> stopWords;
    private StringBuilder logMessage;

    public StringBuilder getLogMessage() {
        return logMessage;
    }
    public void setLogMessage(StringBuilder logMessage) {
        this.logMessage = logMessage;
    }
    public HashSet<String> getStopWords() {
        return stopWords;
    }
    public void setStopWords(HashSet<String> stopWords) {
        this.stopWords = stopWords;
    }
    public List<Window> getTargetTerms() {
        return targetTerms;
    }
    public void setTargetTerms(List<Window> targetTerms) {
        this.targetTerms = targetTerms;
    }
    public List<Window> getUniqueTargetTerms() {
        return uniqueTargetTerms;
    }
    public void setUniqueTargetTerms(List<Window> uniqueTargetTerms) {
        this.uniqueTargetTerms = uniqueTargetTerms;
    }
    public HashSet<String> getUniqueTargetTermsIdentifier() {
        return uniqueTargetTermsIdentifier;
    }
    public void setUniqueTargetTermsIdentifier(HashSet<String> uniqueTargetTermsIdentifier) {
        this.uniqueTargetTermsIdentifier = uniqueTargetTermsIdentifier;
    }
    public ConceptWindow(String wordNetPath) {
        this.wordNetPath = wordNetPath;
```

```
        path = this.wordNetPath + File.separator + "dict";
        targetTerms = new ArrayList<Window>();
        uniqueTargetTerms = new ArrayList<Window>();
        uniqueTargetTermsIdentifier = new HashSet<String>();
        try {
            url = new URL("File", null,path);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        dictionary = new Dictionary(url);
        try {
            dictionary.open();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void create(String document){
        double before, after;
        List<String> Sentence = new ArrayList<String>();
        Scanner scan = new Scanner(document);
        String regx = "(?<!Mr|Mrs|\b[A-Z]|mr|mrs|dr|sr|Sr|Dr|Prof|prof|[0-9])[.][ ]+";
        scan = scan.useDelimiter(regx);
        while(scan.hasNext()) {
            Sentence.add(scan.next());
        }
        scan.close();
        List<Window> Context = new ArrayList<Window>();
        for(int si = 0; si < Sentence.size(); si++) {
            before = System.nanoTime();
            Window context = new Window();
            int i = si - 1;
            int j = si + 1;
            List<String> left = new ArrayList<String>();
            List<String> right = new ArrayList<String>();
            int sum = left.size() + right.size() + 1;
            while (sum != 5 && (!(i < 0 || i < si - 2) || !(j >= Sentence.size() || j > si +
            2))) {
                if(i >= 0) {
                    int diffL = left.size() - right.size();
                    if(diffL <= 0) {
                        left.add(Sentence.get(i));
                    }
                    else {
                        right.add(Sentence.get(i));
                    }
                    i--;
                }
                if(j < Sentence.size()) {
                    int diffR = right.size() - left.size();
                    if(diffR <= 0) {
                        right.add(Sentence.get(j));
                    }
                    else {
                        left.add(Sentence.get(j));
                    }
                    j++;
                }
                sum = left.size() + right.size() + 1;
            }
            String surroundingSentences[] = new String[left.size()+right.size()];
            int k = 0;
            for(String a: left) {
                surroundingSentences[k] = a;
                k++;
            }
            for(String a: right) {
                surroundingSentences[k] = a;
                k++;
            }
            context.setSurroundingTerms(surroundingSentences);
            context.setTargetTerm(Sentence.get(si));
```

```
            Context.add(context);
            after = (System.nanoTime() - before) / 1000000000.0;
        }
        int inc = 0;
        for(Window con: Context) {
            before = System.nanoTime();
            String currentSentence = con.getTargetTerm();
            String surroundSentences[] = con.getSurroundingTerms();
            String cleansedSentence = Remover.removeSymbols(currentSentence);
            String stemmedSentence = Stemmer.stemIt(cleansedSentence);

            String terms[] = stemmedSentence.split(" ");
            for(int i = 0; i < terms.length;  i++) {
                before = System.nanoTime();
                if(!stopWords.contains(terms[i]) && !terms[i].equalsIgnoreCase("")/* &&
                isTargetTerm(terms[i])*/) {
                    List<String> synonyms;
                    synonyms = getSynonyms(terms[i], surroundSentences);
                    Window win;
                    if(synonyms.size() == 0) {
                        win = createWindow(i, terms);
                    }
                    else {
                        win = createWindow(terms[i], synonyms);
                        win = createWindow(i, win, terms);
                    }
                    if(!uniqueTargetTermsIdentifier.contains(terms[i].toLowerCase())) {
                        uniqueTargetTermsIdentifier.add(terms[i].toLowerCase());
                        uniqueTargetTerms.add(win);
                    }
                    targetTerms.add(win);
                }
                else if(!stopWords.contains(terms[i]) && !terms[i].equalsIgnoreCase("")/* &&
                isTargetTerm(terms[i])*/) {
                    String[] surroundingTerms = new String[0];
                    Window win = new Window();
                    win.setSurroundingTerms(surroundingTerms);
                    win.setTargetTerm(terms[i]);
                }
                else {
                }
            }
            inc++;
            after = (System.nanoTime() - before) / 1000000000.0;
        }
    }
    public List<String> getSynonyms (String term, String sen1, String sen2, String sen3){
        return getSynonyms(term, sen1, sen2, sen3, "asdfgh");
    }
    public List<String> getSynonyms (String term, String sen1, String sen2, String sen3,
    String sen4){
        String context = sen1.trim() + "." + sen2 + "." + sen3 + "." + sen4;
        context = Remover.removeSymbols(context);
        context = Stemmer.stemIt(context);

        String contextTokens[] = context.split(" ");
        List<String> synonymWords = new ArrayList<String>();
        HashSet<String> uniqueSynonymWords = new HashSet<String>();
        List<IWordID> ids = new ArrayList<IWordID>();
        IIndexWord indicesN = dictionary.getIndexWord(term, POS.NOUN);
        IIndexWord indicesV = dictionary.getIndexWord(term, POS.VERB);
        IIndexWord indicesAdj = dictionary.getIndexWord(term, POS.ADJECTIVE);
        IIndexWord indicesAdv = dictionary.getIndexWord(term, POS.ADVERB);
        if(indicesN != null) {
            List<IWordID> idN = indicesN.getWordIDs();
            for(IWordID a: idN) {
                ids.add(a);
            }
        }
        if(indicesV != null) {
            List<IWordID> idV = indicesV.getWordIDs();
```

```
            for(IWordID a: idV) {
                ids.add(a);
            }
        }
        if(indicesAdj != null) {
            List<IWordID> idAdj = indicesAdj.getWordIDs();
            for(IWordID a: idAdj) {
                ids.add(a);
            }
        }
        if(indicesAdv != null) {
            List<IWordID> idAdv = indicesAdv.getWordIDs();
            for(IWordID a: idAdv) {
                ids.add(a);
            }
        }
        if(ids.size() != 0) {
            for(String token: contextTokens) {
                for(int i = 0; i < ids.size(); i++) {
                    if(token.equalsIgnoreCase("asdfgh")) {
                        break;
                    }
                    ISynset synset = dictionary.getWord(ids.get(i)).getSynset();
                    List<IWord> synsetWords = synset.getWords();
                    boolean Break = false;
                    for(int j = 0; j < synsetWords.size(); j++) {
                        if(!uniqueSynonymWords.contains(token.toLowerCase()) &&
                        !token.equalsIgnoreCase(term) &&
                        synsetWords.get(j).getLemma().toUpperCase().equals(token.toUpperCase()
                        )) {
                            uniqueSynonymWords.add(token.toLowerCase());
                            synonymWords.add(token.toLowerCase());
                            Break = true;
                            break;
                        }
                    }
                    if(Break)
                        break;
                }
            }
        }
        return synonymWords;
    }
    public List<String> getSynonyms(String term, String sen1, String sen2){
        return getSynonyms(term, sen1, sen2, "asdfgh", "asdfgh");
    }
    public Window createWindow(String term, List<String> synonyms) {
        Window win = new Window();
        String[] surroundWords;
        List<String> leftWords = new ArrayList<String>();
        List<String> rightWords = new ArrayList<String>();
        int sum = leftWords.size() + rightWords.size() + 1;
        int i = 0;
        while(/*!(sum >= 9 || i > synonyms.size() - 1)*/ !(sum >= 9 || i > synonyms.size() -
        1)) {
            String synonymWord = synonyms.get(i);
            int difference = leftWords.size() - rightWords.size();
            if(difference < 0 || difference == 0) {
                leftWords.add(synonymWord.toLowerCase());
            }
            else {
                rightWords.add(synonymWord.toLowerCase());
            }
            sum = leftWords.size() + rightWords.size() + 1;
            i++;
        }
        /*
        sum = leftWords.size() + rightWords.size() + 1;
        if(sum % 2 != 1) {
            int difference = leftWords.size() - rightWords.size();
            if(difference < 0) {
```

```
                leftWords.remove(leftWords.size() - 1);
            }
            else {
                rightWords.remove(rightWords.size() - 1);
            }
        }
        */
        sum = leftWords.size() + rightWords.size();
        surroundWords = new String[sum];
        int k = 0;
        for(int j = 0; j < leftWords.size(); j++,k++) {
            surroundWords[k] = leftWords.get(j);
        }
        for(int j = 0; j < rightWords.size(); j++,k++) {
            surroundWords[k] = rightWords.get(j);
        }
        win.setSurroundingTerms(surroundWords);
        win.setTargetTerm(term.toLowerCase());
        win.setLeftSize(leftWords.size());
        win.setRightSize(rightWords.size());
        return win;
    }
    public Window createWindow(int termIndex, Window win, String terms[]) {
        String surroundWords[] = win.getSurroundingTerms();
        List<String> leftSide = new ArrayList<String>();
        List<String> rightSide = new ArrayList<String>();
        int k = 0;
        for(int i = 0;i < win.getLeftSize(); i++) {
            leftSide.add(surroundWords[k].toLowerCase());
            k++;
        }
        for(int i = 0; i < win.getRightSize(); i++) {
            rightSide.add(surroundWords[k].toLowerCase());
            k++;
        }
        return createWindow(termIndex, leftSide, rightSide, terms);
    }
    public Window createWindow(int termIndex, List<String> leftSide, List<String> rightSide,
    String terms[]) {
        HashSet<String> leftHash = new HashSet<String>();
        HashSet<String> rightHash = new HashSet<String>();
        Window win = new Window();
        String[] surroundWords;
        int sum = leftSide.size() + rightSide.size() + 1;
        int l = termIndex - 1;
        int j = termIndex + 1;
        while(/*!(sum >= 9) && !( l < 0 && j > terms.length - 1)*/ !((sum >= 9) || (l < 0 &&
        j > terms.length -1))){
            for(int i = l; i > -1; i--) {
                if(!stopWords.contains(terms[i].toLowerCase())/* && isTargetTerm(terms[i])*/){
                    break;
                }
                l--;
            }
            for(int i = j; i < terms.length; i++) {
                if(!stopWords.contains(terms[i].toLowerCase())/* && isTargetTerm(terms[i])*/){
                    break;
                }
                j++;
            }
            if(l >= 0) {
                int diffL = leftSide.size() - rightSide.size();
                if(diffL <= 0) {
                    if(leftHash.contains(terms[l].toLowerCase()) ||
                    rightHash.contains(terms[l].toLowerCase())) {
                    }
                    else {
                        leftSide.add(terms[l].toLowerCase());
                        leftHash.add(terms[l].toLowerCase());
                    }
                }
```

```
                else {
                    if(leftHash.contains(terms[l].toLowerCase()) ||
                    rightHash.contains(terms[l].toLowerCase())) {
                    }
                    else {
                        rightSide.add(terms[l].toLowerCase());
                        rightHash.add(terms[l].toLowerCase());
                    }
                }
                l--;
            }
            if(j <= terms.length - 1) {
                int diffR = rightSide.size() - leftSide.size();
                if(diffR <= 0) {
                    if(leftHash.contains(terms[j].toLowerCase()) ||
                    rightHash.contains(terms[j].toLowerCase())) {
                    }
                    else {
                        rightSide.add(terms[j].toLowerCase());
                        rightHash.add(terms[j].toLowerCase());
                    }
                }
                else {
                    if(leftHash.contains(terms[j].toLowerCase()) ||
                    rightHash.contains(terms[j].toLowerCase())) {
                    }
                    else {
                        leftSide.add(terms[j].toLowerCase());
                        leftHash.add(terms[j].toLowerCase());
                    }
                }
                j++;
            }
            sum = leftSide.size() + rightSide.size() + 1;
        }
        sum = leftSide.size() + rightSide.size() + 1;
        if(sum % 2 != 1) {
            int difference = leftSide.size() - rightSide.size();
            if(difference > 0) {
                leftSide.remove(leftSide.size() - 1);
            }
            else {
                rightSide.remove(rightSide.size() - 1);
            }
        }
        sum = leftSide.size() + rightSide.size();
        surroundWords = new String[sum];
        int k = 0;
        for(int p = 0; p < leftSide.size(); p++,k++) {
            surroundWords[k] = leftSide.get(p);
        }
        for(int p = 0; p < rightSide.size(); p++,k++) {
            surroundWords[k] = rightSide.get(p);
        }
        win.setSurroundingTerms(surroundWords);
        win.setTargetTerm(terms[termIndex].toLowerCase());
        win.setLeftSize(leftSide.size());
        win.setRightSize(rightSide.size());
        return win;
    }
    public Window createWindow(int termIndex,String terms[]) {
        List<String> left = new ArrayList<String>();
        List<String> right = new ArrayList<String>();
        return createWindow(termIndex, left, right, terms);
    }
    public boolean isTargetTerm(String token) {
        IIndexWord indicesN = dictionary.getIndexWord(token, POS.NOUN);
        IIndexWord indicesV = dictionary.getIndexWord(token, POS.VERB);
        IIndexWord indicesAdj = dictionary.getIndexWord(token, POS.ADJECTIVE);
        IIndexWord indicesAdv = dictionary.getIndexWord(token, POS.ADVERB);
        return (indicesN != null) || (indicesV != null) || (indicesAdj != null) ||
```

```
        (indicesAdv != null);
    }
    public List<String> getSynonyms (String term, String [] sens){
        String context = "";
        for(int i = 0; i < sens.length; i ++) {
            if(i == 0) {
                context += sens[i].trim() + ".";
            }
            else {
                context += sens[i] + ".";
            }
        }
        context = Remover.removeSymbols(context);
        context = Stemmer.stemIt(context);

        String contextTokens[] = context.split(" ");
        List<String> synonymWords = new ArrayList<String>();
        HashSet<String> uniqueSynonymWords = new HashSet<String>();
        List<IWordID> ids = new ArrayList<IWordID>();
        IIndexWord indicesN = dictionary.getIndexWord(term, POS.NOUN);
        IIndexWord indicesV = dictionary.getIndexWord(term, POS.VERB);
        IIndexWord indicesAdj = dictionary.getIndexWord(term, POS.ADJECTIVE);
        IIndexWord indicesAdv = dictionary.getIndexWord(term, POS.ADVERB);
        if(indicesN != null) {
            List<IWordID> idN = indicesN.getWordIDs();
            for(IWordID a: idN) {
                ids.add(a);
            }
        }
        if(indicesV != null) {
            List<IWordID> idV = indicesV.getWordIDs();
            for(IWordID a: idV) {
                ids.add(a);
            }
        }
        if(indicesAdj != null) {
            List<IWordID> idAdj = indicesAdj.getWordIDs();
            for(IWordID a: idAdj) {
                ids.add(a);
            }
        }
        if(indicesAdv != null) {
            List<IWordID> idAdv = indicesAdv.getWordIDs();
            for(IWordID a: idAdv) {
                ids.add(a);
            }
        }
        if(ids.size() != 0) {
            for(String token: contextTokens) {
                for(int i = 0; i < ids.size(); i++) {
                    if(token.equalsIgnoreCase("asdfgh")) {
                        break;
                    }
                    ISynset synset = dictionary.getWord(ids.get(i)).getSynset();
                    List<IWord> synsetWords = synset.getWords();
                    boolean Break = false;
                    for(int j = 0; j < synsetWords.size(); j++) {
                        if(!uniqueSynonymWords.contains(token.toLowerCase()) &&
                        !token.equalsIgnoreCase(term) &&
                        synsetWords.get(j).getLemma().toUpperCase().equals(token.toUpperCase()
                        )) {
                            uniqueSynonymWords.add(token.toLowerCase());
                            synonymWords.add(token.toLowerCase());
                            Break = true;
                            break;
                        }
                    }
                    if(Break)
                        break;
                }
            }
```

```
        }
        return synonymWords;
    }

    public static void main(String args[]) {
        HashSet<String> stop = stopWordsReader();
        Stemmer.initialize("C://Program Files (x86)/WordNet/2.1");
        String docuPath = "C://Users/harji/Documents/DataSet/Documents/20 Years of Tourism
        Research in Asia Pacific 1996 to 2015.pdf";
        String docuPath2 = "C://Users/harji/Documents/DataSet/Documents/A Community Based
        Tourism Model Its Conception and Use.pdf";
        String document = "";
        String document2 ="";
        try {
            document = PDFtoString.extractPdfText(docuPath);
            document2 = PDFtoString.extractPdfText(docuPath2);
        }
        catch(IOException e) {
            e.printStackTrace();
        }
        ConceptWindow windowCreator = new ConceptWindow("C://Program Files
        (x86)/WordNet/2.1");
        windowCreator.setStopWords(stop);
        double before = System.nanoTime();
        System.out.println(document);
        windowCreator.create(document);
        double after = (System.nanoTime() - before) / 1000000000.0;
        System.out.println("Time processed: "+after+"s");
        List<Window> duplicateTerms = windowCreator.getTargetTerms();
        List<Window> uniqueTerms = windowCreator.getUniqueTargetTerms();
        for(Window win: uniqueTerms) {
            System.out.println(win.getTargetTerm()+" : "+win);
        }
    }
    public static HashSet<String> stopWordsReader(){
        HashSet<String> Space = new HashSet<String>();
        File file;
        BufferedReader bw = null;
        try {
            file = new
            File("C://Users/harji/Documents/DataSet/Table/StopWords/stopwords.txt");
            bw = new BufferedReader(new InputStreamReader(new FileInputStream(file),"UTF-8"));
        }
        catch(IOException e) {
            System.out.println(e.getMessage());
        }
        String line = "";
        try {
            while((line = bw.readLine())!=null) {
                Space.add(line.toLowerCase());
            }
        }
        catch(IOException e) {
            System.out.println(e.getMessage());
        }
        return Space;
    }

}
```

D. modelController.java


```
package controller;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;

import model.ConceptSpacesModel;
import model.ConceptVector;
import model.ConceptWindow;
import model.SuperMatrix;
import model.TFIDFTable;
import utilities.Item;
import utilities.MergeSort;
import utilities.Reader;
import utilities.Stemmer;
import utilities.TermTFIDF;
import utilities.Window;
import utilities.extractPdfText;
public class modelController {
    private String conceptIPath, documentIPath, wordNetPath, stopWordPath, conceptSpaceIPath;
    private int alpha;
    private ArrayList<String> documentPaths;
    private ArrayList<String> conceptPaths;
    private HashMap<Integer, ArrayList<TermTFIDF>> tf_idfTable;
    private HashSet<String> stopWords;
    private List<Window> uniqueTermsPerDoc, originalTermsPerDoc;
    private double[][] documentMatrix;
    private double[][][] ThirdTensor;
    private double[][] super_Matrix;
    private ConceptSpacesModel conceptSpace;
    private ConceptWindow conceptWindow;
    private ConceptVector conceptVector;
    private SuperMatrix superMatrix;
    private extractPdfText extractor;
    private MergeSort<Window> mergeSort;
    private TFIDFTable tfidfTable;

    public String getConceptIPath() {
        return conceptIPath;
    }
    public void setConceptIPath(String conceptIPath) {
        this.conceptIPath = conceptIPath;
    }
    public String getDocumentIPath() {
        return documentIPath;
    }
    public void setDocumentIPath(String documentIPath) {
        this.documentIPath = documentIPath;
    }
    public String getWordNetPath() {
        return wordNetPath;
    }
    public void setWordNetPath(String wordNetPath) {
        this.wordNetPath = wordNetPath;
    }
    public String getStopWordPath() {
        return stopWordPath;
    }
    public void setStopWordPath(String stopWordPath) {
        this.stopWordPath = stopWordPath;
    }
    public int getAlpha() {
        return alpha;
```

```
    }
    public void setAlpha(int alpha) {
        this.alpha = alpha;
    }
    public String getConceptSpaceIPath() {
        return conceptSpaceIPath;
    }
    public void setConceptSpaceIPath(String conceptSpaceIPath) {
        this.conceptSpaceIPath = conceptSpaceIPath;
    }

    public double[][] getDocumentMatrix() {
        return documentMatrix;
    }
    public void setDocumentMatrix(double[][] documentMatrix) {
        this.documentMatrix = documentMatrix;
    }
    public double[][][] getThirdTensor() {
        return ThirdTensor;
    }
    public void setThirdTensor(double[][][] thirdTensor) {
        ThirdTensor = thirdTensor;
    }

    public void display3DMatrix(double[][][] tensor, int maxd, int maxt, int maxc, String
    label) {
        System.out.println(label.toUpperCase());
        System.out.println("OUTPUT: ");
        for(int i = 0; i <= maxd; i++) {
            System.out.println("DOCUMENT# "+i+", 2D MATRIX");
            for(int j = 0; j <= maxt; j++) {
                for(int k = 0; k <= maxc ; k++) {
                    System.out.printf("[%5d][%5d][%5d]:%5.3f\t", i,j,k,tensor[i][j][k]);
                }
                System.out.println();
            }
            System.out.println();
        }
        System.out.println("END OF OUTPUT");
        System.out.println();
    }
    public void display2DMatrix(double [][] matrix, int maxt,int maxD ,String label) {
        System.out.println(label.toUpperCase());
        System.out.println("OUTPUT: ");
        for(int i = 0; i <= maxt; i++) {
            for(int j = 0; j <= maxD; j++) {
                System.out.printf("[%5d][%5d]:%5.3f\t", i,j,matrix[i][j]);
            }
            System.out.println();
        }
        System.out.println("END OF OUTPUT");
        System.out.println();
    }
    public void displayWindow(List<Window> a, String label) {
        System.out.println(label.toUpperCase());
        System.out.println("OUTPUT: ");
        int i = 0;
        for(Window b: a) {
            String[] surround = b.getSurroundingTerms();
            System.out.printf("Target term: %25s", b.getTargetTerm()+" Window:");
            for(int j = 0; j < surround.length; j++) {
                if(j == surround.length - 1) {
                    System.out.printf("%25s",surround[j]);
                }
                else {
                    System.out.printf("%25s,",surround[j]);
                }
            }
            System.out.println();
        }
        System.out.println("END OF OUTPUT");
```

```
        System.out.println();
    }
    public void displayHashWithArray(HashMap<Integer, ArrayList<TermTFIDF>> a, String label) {
        System.out.println(label.toUpperCase());
        System.out.println("OUTPUT: ");
        int i = 0;
        for(ArrayList<TermTFIDF> b: a.values()) {
            System.out.print("DOCUMENT# "+i+"    ");
            for(TermTFIDF c: b) {
                System.out.printf("%16s[%.3f]", c.term, c.tfidfValue);
            }
            System.out.println();
            i++;
        }
        System.out.println("END OF OUTPUT");
        System.out.println();
    }
    public void displayListString(ArrayList<String> a, String label) {
        System.out.println(label.toUpperCase());
        System.out.println("OUTPUT: ");
        for(String b : a) {
            System.out.println("\t"+b);
        }
        System.out.println("END OF OUTPUT");
        System.out.println("");
    }
    public void rewrite() {
        System.out.println("Rewriting Model.....................");
        System.out.println("\tSorting Model.....................");
        ThirdTensor = sortTensor(ThirdTensor);
        System.out.println("\tSorting Model Done.................");
            display3DMatrix(ThirdTensor,2,ThirdTensor[0].length / 2,
            ThirdTensor[0][0].length / 2, "SORTED THIRD ORDER TENSOR MODEL");
        System.out.println("\tConstructing super matrix.........");
        superMatrix = new SuperMatrix();
        superMatrix.setThirdOrderTensor(ThirdTensor);
        superMatrix.buildSuperMatrix();
        super_Matrix = superMatrix.getSuperMatrix();
        System.out.println("\tConstructing super matrix done....");
            display2DMatrix(super_Matrix,super_Matrix.length / 2, super_Matrix[0].length /
            2,"SORTED SUPER MATRIX");
        System.out.println("\tCaching out the models............");
        outputThirdTensor();
        outputSuperMatrix();
        System.out.println("\tCaching done......................");
        System.out.println("Rewriting Model done................");
    }
    public void create() {
        double b, a;
        b = System.nanoTime();
        Stemmer.initialize(wordNetPath);
        System.out.println("Concept space construction...................");
        conceptSpace = new ConceptSpacesModel(16570);
        conceptSpace.setConceptIndexPath(conceptIPath);
        conceptSpace.setDocsIndexPath(documentIPath);
        conceptSpace.setAlpha(alpha);
        conceptSpace.setConceptSpaceIndexPath(conceptSpaceIPath);
        try {
            conceptSpace.conceptSpaceConstruction();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        conceptPaths = conceptSpace.getConceptPath();
        documentPaths = conceptSpace.getDocPath();
        tf_idfTable = conceptSpace.getDocTFIDF();
            displayListString(conceptPaths,"CONCEPT SPACE");
            displayListString(documentPaths,"LIST OF DOCUMENTS");
            displayHashWithArray(tf_idfTable, "DOCUMENT MATRIX");
        System.out.println("End concept space construction..............");
```

```
        System.out.println("Concept window construction...................");
        stopWords = stopWordsReader();
        extractor = new extractPdfText();
        List<String> documents = new ArrayList<String>(documentPaths.size());
        System.out.println("Preparing Documents...");
        int j = 0;
        for(String docPath: documentPaths) {
            System.out.println("Extracting Document #: "+j+" total: "+documents.size());
            extractor.setPath(docPath);
            String document = "";
            try {
                document = extractor.extractPdfText();
            }
            catch(IOException e) {
                e.printStackTrace();
            }
            documents.add(document);
            j++;
        }
        conceptWindow = new ConceptWindow(wordNetPath);
        conceptWindow.setStopWords(stopWords);
        for( int i = 0; i < documents.size(); i++) {
            double before = System.nanoTime();
            System.out.println("Document #: "+i+" total: "+documents.size());
            conceptWindow.create(documents.get(i));
            double after = (System.nanoTime() - before) / 1000000000.0;
            System.out.println("Time processed: "+after+"s");
        }
        uniqueTermsPerDoc = conceptWindow.getUniqueTargetTerms();
        originalTermsPerDoc = conceptWindow.getTargetTerms();
            displayWindow(uniqueTermsPerDoc, "UNIQUE LIST OF CONCEPT WINDOWS");
            displayWindow(originalTermsPerDoc, "NON-UNIQUE LIST OF CONCEPT WINDOWS");
        System.out.println("End concept window construction.............");
        mergeSort = new MergeSort<Window>();
        mergeSort.setOrder(mergeSort.ASCENDING);
        uniqueTermsPerDoc = mergeSort.mergesort(uniqueTermsPerDoc);

        tfidfTable = new TFIDFTable();
        tfidfTable.setDocTFIDF(tf_idfTable);
        tfidfTable.setUniqueTargetTerms(uniqueTermsPerDoc);
        tfidfTable.refine();
        documentMatrix = tfidfTable.getDocumentSpaceMatrix();
        display2DMatrix(documentMatrix,documentMatrix.length /2,documentMatrix[0].length / 2
        ,"DOCUMENT MATRIX NEW");

        conceptVector = new ConceptVector(wordNetPath, conceptPaths);
        conceptVector.setTFIDFTable(documentMatrix);
        conceptVector.setTerms(originalTermsPerDoc);

        System.out.println("concept vector construction.................");
        ThirdTensor = conceptVector.createVector();
        System.out.println("End concept vector construction.............");
        display3DMatrix(ThirdTensor,2,ThirdTensor[0].length / 2, ThirdTensor[0][0].length /
        2, "THIRD ORDER TENSOR MODEL");


        System.out.println("Sorting Third tensor model.................");
        ThirdTensor = sortTensor(ThirdTensor);
        System.out.println("Sorting Third tensor model done............");
        display3DMatrix(ThirdTensor,2,ThirdTensor[0].length / 2, ThirdTensor[0][0].length /
        2, "SORTED THIRD ORDER TENSOR MODEL");


        System.out.println("Super matrix construction..................");
        superMatrix = new SuperMatrix();
        superMatrix.setThirdOrderTensor(ThirdTensor);
        superMatrix.buildSuperMatrix();
        super_Matrix = superMatrix.getSuperMatrix();
        display2DMatrix(super_Matrix,super_Matrix.length / 2, super_Matrix[0].length /
        2,"SUPER MATRIX");
        System.out.println("End super matrix construction..............");
```

```
        System.out.println("Creating txt files for resources...........");
        outputThirdTensor();
        outputSuperMatrix();
        outputDocumentPath();
        outputTerms();
        System.out.println("Creating txt files for resources done......");
        a = (System.nanoTime() - b) / 1000000000.0;
        System.out.println("Time duration :"+a);
    }
    public HashSet<String> stopWordsReader(){
        HashSet<String> Space = new HashSet<String>();
        File file;
        BufferedReader bw = null;
        try {
            file = new File(stopWordPath);
            bw = new BufferedReader(new InputStreamReader(new FileInputStream(file),"UTF-8"));
        }
        catch(IOException e) {
            System.out.println(e.getMessage());
        }
        String line = "";
        try {
            while((line = bw.readLine())!=null) {
                Space.add(line.toLowerCase());
            }
        }
        catch(IOException e) {
            System.out.println(e.getMessage());
        }
        return Space;
    }
    public void outputThirdTensor() {
        BufferedWriter bw = null;
        FileWriter fw = null;
        try {
            fw = new FileWriter("thirdtensor.txt");
            bw = new BufferedWriter(fw);

            bw.write(ThirdTensor.length+"###"+ThirdTensor[0].length+"###"+ThirdTensor[0][0].le
            ngth);
            bw.newLine();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
        for(int i = 0; i < ThirdTensor.length; i++) {
            for(int j = 0; j < ThirdTensor[i].length; j++) {
                for(int k = 0; k < ThirdTensor[i][j].length; k++) {
                    try {
                        bw.write(ThirdTensor[i][j][k]+"");
                        bw.newLine();
                    }
                    catch(IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
        try {
            bw.close();
            fw.close();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
    public void outputSuperMatrix() {
        BufferedWriter bw = null;
        FileWriter fw = null;
        try {
```

```
            fw = new FileWriter("supermatrix.txt");
            bw = new BufferedWriter(fw);
            bw.write(super_Matrix.length+"###"+super_Matrix[0].length);
            bw.newLine();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
        for(int i = 0; i < super_Matrix.length; i++) {
            for(int j = 0; j < super_Matrix[i].length; j++) {
                try {
                    bw.write(super_Matrix[i][j]+"");
                    bw.newLine();
                }
                catch(IOException e) {
                    e.printStackTrace();
                }
            }
        }
        try {
            bw.close();
            fw.close();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
    public void outputDocumentPath() {
        BufferedWriter bw = null;
        FileWriter fw = null;
        try {
            fw = new FileWriter("docupath.txt");
            bw = new BufferedWriter(fw);
        }
        catch(IOException e) {
            e.printStackTrace();
        }
        for(String a: documentPaths) {
            try {
                bw.write(a);
                bw.newLine();
            }
            catch(IOException e) {
                e.printStackTrace();
            }
        }
        try {
            bw.close();
            fw.close();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
    public void outputTerms() {
        BufferedWriter bw = null;
        FileWriter fw = null;
        try {
            fw = new FileWriter("terms.txt");
            bw = new BufferedWriter(fw);
        }
        catch(IOException e) {
            e.printStackTrace();
        }
        int i = 0;
        for(Window a: uniqueTermsPerDoc) {
            try {
                bw.write(a.getTargetTerm().toLowerCase()+"###"+i);
                bw.newLine();
                i++;
            }
```

```
            catch(IOException e) {
                e.printStackTrace();
            }
        }
        try {
            bw.close();
            fw.close();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
    public static void main(String args[]) {
        modelController creator = new modelController();
        /*
        creator.setConceptIPath("C://Users/harji/Documents/DataSet/ConceptIndices");
        creator.setAlpha(20);
        creator.setDocumentIPath("C://Users/harji/Documents/DataSet/DocumentIndices");
        creator.setConceptSpaceIPath("C://Users/harji/Documents/DataSet/ConceptSpaceIndices");
        creator.setWordNetPath("C://Program Files (x86)/WordNet/2.1");

        creator.setStopWordPath("C://Users/harji/Documents/DataSet/Table/StopWords/stopwords.t
        xt");
        creator.create();
        */
        creator.setThirdTensor(Reader.readTensor("thirdtensor.txt"));
        creator.rewrite();

    }
    public double[][][] sortTensor(double[][][] dummy){
        MergeSort<Item> merger = new MergeSort<Item>();
        merger.setOrder(merger.DESCENDING);
        List<List<Item>> sortedDocu = new ArrayList<List<Item>>();
        for(int i = 0; i < dummy.length; i++) {
            List<Item> sumPerConcept = new ArrayList<Item>();
            for(int j = 0; j < dummy[i][0].length; j++) {
                double sum = 0;
                for(int k = 0; k < dummy[i].length; k++) {
                    sum += dummy[i][k][j];
                }
                Item con = new Item();
                con.setIndex(j);
                con.setSum(sum);
                sumPerConcept.add(con);
            }
            sumPerConcept = merger.mergesort(sumPerConcept);
            sortedDocu.add(sumPerConcept);
        }
        //double[][][] newThirdTensor = new
        double[dummy.length][dummy[0].length][dummy[0][0].length];
        for(int i = 0; i < dummy.length; i++) {
            List<Item> sortedConcepts = sortedDocu.get(i);
            for(int j = 0; j < dummy[i].length; j++) {
                double[] newOrder = new double[sortedConcepts.size()];
                for(int k = 0; k < dummy[i][j].length; k++) {
                    newOrder[k] = dummy[i][j][sortedConcepts.get(k).getIndex()];
                }
                dummy[i][j] = newOrder;
            }
        }
        return dummy;
    }
    public void outputDocumentMatrix() {
        BufferedWriter bw = null;
        FileWriter fw = null;
        try {
            fw = new FileWriter("documentmatrix.txt");
            bw = new BufferedWriter(fw);
            bw.write(documentMatrix.length+"###"+documentMatrix[0].length);
            bw.newLine();
        }
```

```
        catch(IOException e) {
            e.printStackTrace();
        }
        for(int i = 0; i < documentMatrix.length; i++) {
            for(int j = 0; j <documentMatrix[i].length; j++) {
                try {
                    bw.write(documentMatrix[i][j]+"");
                    bw.newLine();
                }
                catch(IOException e) {
                    e.printStackTrace();
                }
            }
        }
        try {
            bw.close();
            fw.close();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```

E. SuperMatrix.java


```
package model;

public class SuperMatrix {
    private double[][][] thirdOrderTensor;
    private double[][] superMatrix;
    private String logMessage;
    private StringBuilder str;

    public double[][][] getThirdOrderTensor() {
        return thirdOrderTensor;
    }

    public void setThirdOrderTensor(double[][][] thirdOrderTensor) {
        this.thirdOrderTensor = thirdOrderTensor;
    }

    public double[][] getSuperMatrix() {
        return superMatrix;
    }

    public void setSuperMatrix(double[][] superMatrix) {
        this.superMatrix = superMatrix;
    }

    public void buildSuperMatrix() {
        str = new StringBuilder("");
        System.out.println("Building Super Matrix...");
        superMatrix = new double[thirdOrderTensor[0].length][thirdOrderTensor[0][0].length];
        int numDocs = thirdOrderTensor.length;
        for (int i = 0; i < numDocs; i++) {
            System.out.println("Adding Document "+i+" matrix to Super Matrix");
            System.out.println("Remaining Document: "+(numDocs-(i+1))+"/"+numDocs);
            double[][] matrix = thirdOrderTensor[i];
            if (i == (numDocs - 1)) {
                System.out.println("Super Matrix: ");
                int p = matrix.length * matrix[0].length;
                for (int j = 0; j < matrix.length; j++) {
                    for (int k = 0; k < matrix[j].length; k++) {
                        superMatrix[j][k] += matrix[j][k];
                        superMatrix[j][k] = superMatrix[j][k] / numDocs;
                        System.out.println("Remaining compuations for SuperMatrix, "+p);
                        p--;
                    }
                }
            } else {
                for (int j = 0; j < matrix.length; j++) {
                    for (int k = 0; k < matrix[j].length; k++) {
                        superMatrix[j][k] += matrix[j][k];
                    }
                }
            }
        }
    }
    public static void main(String[] args) throws Exception {
        double[][][] thirdOrderTensor = { { { 1, 2, 3 }, { 4, 5, 6 } }, { { 1, 2, 3 }, { 13,
        14, 15 } },
                { { 1, 2, 3 }, { 22, 23, 24 } } };
        SuperMatrix sm = new SuperMatrix();
        sm.setThirdOrderTensor(thirdOrderTensor);
        sm.buildSuperMatrix();
    }
}
```

F. TFIDFTable.java


```
package model;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import utilities.TermTFIDF;
import utilities.Window;

public class TFIDFTable {
    private HashMap<Integer, ArrayList<TermTFIDF>> docTFIDF;
    private List<Window> uniqueTargetTerms;
    private double [][] DocumentSpaceMatrix;
    public HashMap<Integer, ArrayList<TermTFIDF>> getDocTFIDF() {
        return docTFIDF;
    }
    public void setDocTFIDF(HashMap<Integer, ArrayList<TermTFIDF>> docTFIDF) {
        this.docTFIDF = docTFIDF;
    }
    public List<Window> getUniqueTargetTerms() {
        return uniqueTargetTerms;
    }
    public void setUniqueTargetTerms(List<Window> uniqueTargetTerms) {
        this.uniqueTargetTerms = uniqueTargetTerms;
    }
    public double[][] getDocumentSpaceMatrix() {
        return DocumentSpaceMatrix;
    }
    public void setDocumentSpaceMatrix(double[][] documentSpaceMatrix) {
        DocumentSpaceMatrix = documentSpaceMatrix;
    }

    public void refine() {
        DocumentSpaceMatrix = new double[uniqueTargetTerms.size()][docTFIDF.size()];
        for(int i = 0; i < DocumentSpaceMatrix[0].length; i++) {
            HashMap<String, Double> keys = new HashMap<String, Double>();
            ArrayList<TermTFIDF> terms = docTFIDF.get(i);
            for(int p = 0; p < terms.size(); p++) {
                TermTFIDF t = terms.get(p);
                keys.put(t.term.toLowerCase(), t.tfidfValue);
            }
            for( int j = 0; j < uniqueTargetTerms.size(); j++) {
                Window term2 = uniqueTargetTerms.get(j);
                Double value = keys.get(term2.getTargetTerm().toLowerCase());
                if(value == null) {
                    DocumentSpaceMatrix[j][i] = 0;
                }
                else {
                    DocumentSpaceMatrix[j][i] = value.doubleValue();
                }
            }
        }
    }
    public static void main(String args[]) {
        HashMap<Integer, ArrayList<TermTFIDF>> a = new HashMap<Integer,
        ArrayList<TermTFIDF>>();
        ArrayList<TermTFIDF> b = new ArrayList<TermTFIDF>();
        ArrayList<TermTFIDF> c = new ArrayList<TermTFIDF>();
        TermTFIDF term1 = new TermTFIDF("apple", 5);
        TermTFIDF term2 = new TermTFIDF("apple", 2);
        TermTFIDF term3 = new TermTFIDF("apple", 1);
        TermTFIDF term4 = new TermTFIDF("apple", 3);
        b.add(term1);
        b.add(term2);
        c.add(term3);
        c.add(term4);
        a.put(0, b);
        a.put(1, c);

        List<Window> unique = new ArrayList<Window>();
        Window win = new Window();
        win.setTargetTerm("apple");
```

```
        unique.add(win);
        TFIDFTable tf = new TFIDFTable();
        tf.setDocTFIDF(a);
        tf.setUniqueTargetTerms(unique);
        tf.refine();
        double[][] matrix = tf.getDocumentSpaceMatrix();
        for(int i = 0; i < matrix.length; i++) {
            for(int j = 0; j < matrix[0].length; j++) {
                System.out.print(matrix[i][j]+" ");
            }
        }

    }
}
```

Search Phase

A. AppController.java

C:\Users\jmags\Downloads\java\search phase\AppController.java Tuesday, 5 December 2017 4:39 AM

```
package controller;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Paths;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.store.FSDirectory;

import model.CRUD;
import model.Processor;
import model.Refinement;
import model.ResultHistory;
import model.SearchModule;
import model.luceneSearch;
import utilities.DBConUtility;
import utilities.Reader;
import utilities.Stemmer;

public class AppController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private double[][][] model;
    private double[][] matrix;
    private ArrayList<String> documents;
    private HashMap<String, Integer> terms;
    private HashSet<String> stopWords;
    private String modelPath, matrixPath, docuPath, conceptSpaceIPath, termPath,
    wordNetPath, stopWordPath;
    private IndexReader reader;
    private IndexSearcher searcher;
    private Connection con;
    public void display2DMatrix(double [][] matrix, int maxt,int maxD ,String label) {
        System.out.println(label.toUpperCase());
        System.out.println("OUTPUT: ");
        for(int i = 0; i <= maxt; i++) {
            for(int j = 0; j <= maxD; j++) {
                System.out.printf("[%5d][%5d]:%5.3f\t", i,j,matrix[i][j]);
            }
            System.out.println();
        }
        System.out.println("END OF OUTPUT");
        System.out.println();
    }
    public void init(ServletConfig config)throws ServletException{
        super.init(config);
        modelPath = config.getInitParameter("tensorPath");
        matrixPath = config.getInitParameter("matrixPath");
        docuPath = config.getInitParameter("docuPath");
        conceptSpaceIPath = config.getInitParameter("csIPath");
        termPath = config.getInitParameter("termPath");
        wordNetPath = config.getInitParameter("wordNetPath");
        stopWordPath = config.getInitParameter("stopPath");
        terms = Reader.readTerms(termPath);
        model  = Reader.readTensor(modelPath);
        matrix = Reader.readMatrix(matrixPath);
        documents = Reader.readDocuPath(docuPath);
```

```
        stopWords = Reader.readStopWords(stopWordPath);
        Stemmer.initialize(wordNetPath);
        con = DBConUtility.getConnection();

    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        double before = System.nanoTime();
        System.out.println("Actual running time: "+before);
        String query = request.getParameter("query");
        double alpha = Double.parseDouble(request.getParameter("alpha"));

        System.out.println("Preprocessing of the query...");
        Processor p = new Processor();
        p.setStopWords(stopWords);
        p.setQuery(query);
        String[] tokens = p.processIt();
        System.out.print("\t\tProcessed query: ");
        for(int i = 0; i < tokens.length; i++) {
            if(i == tokens.length - 1) {
                System.out.print(tokens[i]);
            }
            else {
                System.out.print(tokens[i] +", ");
            }
        }
        System.out.println();

        System.out.println("User query concept Space construction......");
        reader = DirectoryReader.open(FSDirectory.open(Paths.get(conceptSpaceIPath)));
        searcher = new IndexSearcher(reader);
        luceneSearch ls = new luceneSearch();
        ls.setSearcher(searcher);
        ls.setNumDocs(100);
        try {
            ls.search(query);
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        for(String s: ls.getResults()) {
            System.out.println("\t\t"+s);
        }
        System.out.print("\t\tOutput of user concept space: ");
        for(String s: ls.getResults()) {
            System.out.println("\t\t"+s);
        }
        System.out.println("Refinement of third tensor model and super matrix...");
        int size =  ls.getResults().size();
        Refinement refiner  = new Refinement();
        refiner.setMatrix(matrix);
        refiner.setTensor(model);
        refiner.setSize(size);
        refiner.refine();
        int limit = refiner.getLimit();
        System.out.println("\t\tlimit size: "+size);
        System.out.println("User query matrix construction.....");
        SearchModule searcher = new SearchModule();
        searcher.setLimit(limit);
        searcher.setDocPath(documents);
        searcher.setQuery(tokens);
        searcher.setSuperMatrix(matrix);
        searcher.setTensorRefined(model);
        searcher.setTerms(terms);
        searcher.buildQuerySuperMatrix();
        double[][] queryMatrix = searcher.getQuerySuperMatrix();
        display2DMatrix(queryMatrix, queryMatrix.length - 1, queryMatrix[0].length - 1,
        "QUERY MATRIX");
        System.out.println("Result set search.....");
        ResultHistory pool = new ResultHistory();
        pool.setAlpha(alpha);
```

```
        pool.setMatrix(queryMatrix);
        pool.search();
        System.out.println("\t\tmatch: "+pool.isMatch());
        if(pool.isMatch()) {
            System.out.println("\t\tRetrieved documents: ");
            request.setAttribute("documents", pool.getResultSet());
        }
        else {
            System.out.println("Searching documents.....");
            searcher.search();
            System.out.println("\t\tResult set:");
            List<String> results = searcher.getResults();
            try {
                CRUD.insertToDB(con, results, queryMatrix, query);
            }
            catch(SQLException e) {
                e.printStackTrace();
            }
            request.setAttribute("documents", results);
        }
        double after = System.nanoTime();
        after = (after - before) / 1000000000.0;
        System.out.println("Time elapsed:\t\t"+after);
        request.getRequestDispatcher("searchResults.jsp").forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        doGet(request, response);
    }

}
```

B. CRUD.java


```
package model;

import java.util.ArrayList;
import java.util.List;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import utilities.History;

public class CRUD {
    public static void main(String args[]) {
        double array[][] = new double[900000][1];
        for(int i = 0; i < array.length; i++) {
            for(int j = 0; j < array[0].length; j++) {
                array[i][j] = 9;
            }
        }
    }
    public static List<Double> toList(double[][] array) {
        List<Double> listHolder = new ArrayList<Double>();
        int arrayRowLength = array.length;
        int arrayColumnLength = array[0].length;

        for (int i = 0; i < arrayRowLength; i++) {
            for (int j = 0; j < arrayColumnLength; j++) {
                listHolder.add(array[i][j]);
            }
        }
        return listHolder;
    }

    public static double[][] toArray(List<Double> list, int row, int column) {
        double[][] arrayHolder = new double[row][column];
        int counter = 0;
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < column; j++) {
                arrayHolder[i][j] = list.get(counter);
                counter++;
            }
        }
        return arrayHolder;
    }

    public static long insertToDB(Connection con, List<String> concepts, double[][] matrix,
    String query)
            throws SQLException {
        List<Object> object = new ArrayList<Object>();
        int row = matrix.length;
        int column = matrix[0].length;
        List<Double> matrixList = toList(matrix);
        object.add(query);
        object.add(concepts);
        object.add(matrixList);
        object.add(row);
        object.add(column);

        PreparedStatement pstmt = con.prepareStatement("insert into concepttable(object)
        values (?)",
                Statement.RETURN_GENERATED_KEYS);
        pstmt.setObject(1, object);
        pstmt.executeUpdate();
        ResultSet rs = pstmt.getGeneratedKeys();
        int serialid = -1;
        if (rs.next()) {
            serialid = rs.getInt(1);
```

```
        }
        rs.close();
        pstmt.close();
        return serialid;
    }

    public static History[] getResult(Connection con) throws SQLException, IOException,
    ClassNotFoundException {
        PreparedStatement pstmt = con.prepareStatement("select object from concepttable");
        ResultSet rs = pstmt.executeQuery();
        List<History> history = new ArrayList<History>();
        while (rs.next()) {
            byte[] buffer = rs.getBytes(1);
            ObjectInputStream objectIn = null;
            if (buffer != null) {
                objectIn = new ObjectInputStream(new ByteArrayInputStream(buffer));
            }

            Object object = objectIn.readObject();
            List<Object> objectHolder = (List<Object>) object;
            String queryHolder = (String) objectHolder.get(0);
            List<String> conceptHolder = (List<String>) objectHolder.get(1);
            List<Double> matrixListHolder = (List<Double>) objectHolder.get(2);
            int rowHolder = (Integer) objectHolder.get(3);
            int columnHolder = (Integer) objectHolder.get(4);
            double[][] matrixHolder = toArray(matrixListHolder, rowHolder, columnHolder);

            History result = new History();
            result.setQuery(queryHolder);
            result.setResultSet(conceptHolder);
            result.setMatrix(matrixHolder);

            history.add(result);


        }
        rs.close();
        pstmt.close();

        History[] historyHolder = new History[history.size()];
        historyHolder = history.toArray(historyHolder);

        return historyHolder;
    }

    public static void deleteFromDB(Connection con, long serialid) throws SQLException {
        PreparedStatement pstmt = con.prepareStatement("delete from concepttable where
        serialid = ?");
        pstmt.setLong(1, serialid);
        pstmt.executeQuery();
        pstmt.close();
    }


}
```

C. luceneSearch.java


```
package model;

import java.nio.file.Paths;
import java.util.ArrayList;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.BooleanQuery;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.FSDirectory;

public class luceneSearch {
    private IndexSearcher searcher;
    private Query query;
    private int numDocs;
    private ArrayList<String> results;
    private int[] conceptFreq;

    public luceneSearch() {
        results = new ArrayList<String>();
    }

    public Query getQuery() {
        return query;
    }

    public void setQuery(Query query) {
        this.query = query;
    }

    public int getNumDocs() {
        return numDocs;
    }

    public void setNumDocs(int numDocs) {
        this.numDocs = numDocs;
    }

    public ArrayList<String> getResults() {
        return results;
    }

    public int[] getConceptFreq() {
        return conceptFreq;
    }

    public void setConceptFreq(int[] conceptFreq) {
        this.conceptFreq = conceptFreq;
    }

    public IndexSearcher getSearcher() {
        return searcher;
    }

    public void setSearcher(IndexSearcher searcher) {
        this.searcher = searcher;
    }

    public void search() throws Exception {
        BooleanQuery.setMaxClauseCount(Integer.MAX_VALUE);
        TopDocs results = searcher.search(query, numDocs);
        ScoreDoc[] hits = results.scoreDocs;
        int numTotalHits = results.totalHits;
        for (int j = 0; j < numTotalHits; j++) {
```

```
            Document doc = searcher.doc(hits[j].doc);
            getResults().add(doc.get("path"));
            conceptFreq[Integer.parseInt(doc.get("docID"))]++;
        }
    }

    public void search(String query) throws Exception {
        Analyzer analyzer = new StandardAnalyzer();
        QueryParser parser = new QueryParser("words", analyzer);
        Query q = parser.parse(query);
        TopDocs results = searcher.search(q, numDocs);
        ScoreDoc[] hits = results.scoreDocs;
        BooleanQuery.setMaxClauseCount(Integer.MAX_VALUE);
        int numTotalHits = results.totalHits;
        for (int j = 0; j < numTotalHits; j++) {
            Document doc = searcher.doc(hits[j].doc);
            getResults().add(doc.get("path"));
        }
    }

    public static void main(String[] args) throws Exception {
        IndexReader reader =
        DirectoryReader.open(FSDirectory.open(Paths.get("conceptsIndex")));
        IndexSearcher searcher = new IndexSearcher(reader);
        luceneSearch ls = new luceneSearch();
        ls.setSearcher(searcher);
        ls.setNumDocs(101);
        ls.search("philippines");
        ArrayList<String> results = ls.getResults();
        for (int i = 0; i < results.size(); i++) {
            System.out.println(results.get(i));
        }
    }
}
```

D. Processor.java


```
package model;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

import utilities.Remover;
import utilities.Stemmer;

public class Processor {
    private String query;
    private HashSet<String> stopWords;

    public HashSet<String> getStopWords() {
        return stopWords;
    }

    public void setStopWords(HashSet<String> stopWords) {
        this.stopWords = stopWords;
    }

    public String getQuery() {
        return query;
    }

    public void setQuery(String query) {
        this.query = query;
    }

    public String[] processIt() {
        query = Remover.removeSymbols(query);
        query = Stemmer.stemIt(query);
        String[] tokens = query.split(" ");
        List<String> refinedTokens = new ArrayList<String>();
        for(String a: tokens) {
            if(stopWords.contains(a)) {
            }
            else {
                refinedTokens.add(a);
            }
        }
        return refinedTokens.toArray(new String[0]);
    }
}
```

E. Refinement.java


```
package model;

public class Refinement {
    private double[][][] tensor;
    private double[][] matrix;
    private int size;
    private int limit;

    public int getLimit() {
        return limit;
    }
    public void setLimit(int limit) {
        this.limit = limit;
    }
    public double[][][] getTensor() {
        return tensor;
    }
    public void setTensor(double[][][] tensor) {
        this.tensor = tensor;
    }
    public double[][] getMatrix() {
        return matrix;
    }
    public void setMatrix(double[][] matrix) {
        this.matrix = matrix;
    }
    public int getSize() {
        return size;
    }
    public void setSize(int size) {
        this.size = size;
    }
    public void refine() {
        if(size >= tensor[0][0].length) {
            limit = tensor[0][0].length;
        }
        else {
            limit = size;
        }
    }

}
```

F. ResultHistory.java


```
package model;


import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import utilities.DBConUtility;
import utilities.History;
import utilities.Matrices;
import utilities.MergeSort;
import utilities.simMeasure;

public class ResultHistory {
    private double alpha;
    private Connection con;
    private double[][] matrix;
    private List<String> resultSet;
    private boolean match;
    public double getAlpha() {
        return alpha;
    }

    public double[][] getMatrix() {
        return matrix;
    }

    public void setMatrix(double[][] matrix) {
        this.matrix = matrix;
    }

    public List<String> getResultSet() {
        return resultSet;
    }

    public void setResultSet(List<String> resultSet) {
        this.resultSet = resultSet;
    }

    public boolean isMatch() {
        return match;
    }

    public void setMatch(boolean match) {
        this.match = match;
    }

    public void setAlpha(double alpha) {
        this.alpha = alpha;
    }
    public double[][] equalize(double[][] matrix, int size){
        double[][] newMatrix = new double[matrix.length][size];
        for(int i = 0; i < matrix.length; i++) {
            for(int j = 0; j < size; j++) {
                if(j < matrix[0].length) {
                    newMatrix[i][j] = matrix[i][j];
                }
                else {
                    newMatrix[i][j] = 0;
                }
            }
        }
        return newMatrix;
    }
    public ResultHistory() {
        try {
            con = DBConUtility.getConnection();
        }
        catch(Exception e) {
```

```
            System.out.println(e.getMessage());
        }
    }

    public void search() {

        History[] histories = null;
        try {
            histories = CRUD.getResult(con);
        }
        catch(SQLException e) {
            System.out.println(e.getMessage());
        }
        catch(IOException e) {
            System.out.println(e.getMessage());
        }
        catch(ClassNotFoundException e) {
            System.out.println(e.getMessage());
        }
        finally{
            try{ if(con != null) con.close(); } catch(Exception e) {}
        }
        if(histories.length != 0) {
            List<Matrices> matrices = new ArrayList<Matrices>();
            for(int i = 0; i < histories.length; i++) {
                Matrices mainMatrix = new Matrices();
                simMeasure calculator = new simMeasure();
                double [][] matrix2 = histories[i].getMatrix();
                if(matrix[0].length > matrix2[0].length) {
                    matrix2 = equalize(matrix2, matrix[0].length);
                }
                else if(matrix[0].length < matrix2[0].length) {
                    matrix = equalize(matrix, matrix2[0].length);
                }
                else {
                }
                double score = 0;
                calculator.setMatrix1(matrix);
                calculator.setMatrix2(matrix2);
                calculator.setLimit(matrix[0].length);
                score = calculator.measure();
                System.out.println("Match #"+(i+1)+": "+score);
                mainMatrix.setMatrix1(matrix);
                mainMatrix.setMatrix2(matrix2);
                mainMatrix.setIndex(i);
                mainMatrix.setScore(score);;
                matrices.add(mainMatrix);
            }
            MergeSort<Matrices> merger = new MergeSort<Matrices>();
            merger.setOrder(merger.DESCENDING);
            matrices = merger.mergesort(matrices);

            if(matrices.get(0).getScore() >= alpha) {
                match = true;
                resultSet = histories[matrices.get(0).getIndex()].getResultSet();
            }
            else {
                match = false;
                resultSet = null;
            }
        }
        else {
            match = false;
            resultSet = null;
        }
    }
}
```

G. SearchModule.java

C:\Users\jmags\Downloads\java\search phase\SearchModule.java — Tuesday, 5 December 2017 4:40 AM

```
package model;

import java.util.ArrayList;
import java.util.HashMap;
import utilities.*;

public class SearchModule {
    private HashMap<String, Integer> terms;
    private ArrayList<String> docPath, results;
    private double[][][] tensorRefined;
    private double[][] superMatrix, querySuperMatrix;
    private String[] query;
    private int limit;

    public int getLimit() {
        return limit;
    }
    public void setLimit(int limit) {
        this.limit = limit;
    }

    public SearchModule() {
        results = new ArrayList<String>();
    }

    public ArrayList<String> getResults() {
        return results;
    }

    public double[][] getQuerySuperMatrix() {
        return querySuperMatrix;
    }

    public void setTerms(HashMap<String, Integer> terms) {
        this.terms = terms;
    }

    public void setDocPath(ArrayList<String> docPath) {
        this.docPath = docPath;
    }

    public void setTensorRefined(double[][][] tensorRefined) {
        this.tensorRefined = tensorRefined;
    }

    public void setSuperMatrix(double[][] superMatrix) {
        this.superMatrix = superMatrix;
    }

    public void setQuery(String[] query) {
        this.query = query;
    }

    public void search() {
        ArrayList<Double> similarityList = new ArrayList<Double>();
        for (int i = 0; i < tensorRefined.length; i++) {
            double[][] matrix = tensorRefined[i];
            simMeasure sim = new simMeasure();
            sim.setMatrix1(matrix);
            sim.setMatrix2(querySuperMatrix);
            sim.setLimit(limit);
            double similarity = sim.measure();
            if (!(similarity > 0)) {
                continue;
            } else {
                if (similarityList.isEmpty()) {
                    similarityList.add(similarity);
                    results.add(docPath.get(i));
                } else if (similarityList.get(similarityList.size() - 1) > similarity) {
                    similarityList.add(similarity);
                    results.add(docPath.get(i));
```

```
                } else {
                    for (int j = 0; j < similarityList.size(); j++) {
                        if (similarityList.get(j) <= similarity) {
                            similarityList.add(j, similarity);
                            results.add(j, docPath.get(i));
                            break;
                        }
                    }
                }
            }
        }
    }

    public void buildQuerySuperMatrix() {
        int numTerms = terms.size();
        int numConcepts = limit;
        querySuperMatrix = new double[numTerms][numConcepts];
        for (int i = 0; i < numTerms; i++) {
            for (int j = 0; j < numConcepts; j++) {
                querySuperMatrix[i][j] = 0;
            }
        }
        for (int i = 0; i < query.length; i++) {
            int queryIndex = (terms.get(query[i]) != null) ? terms.get(query[i]) : -1;
            if (queryIndex == -1) {
                continue;
            } else {
                for (int j = 0; j < numConcepts; j++) {
                    querySuperMatrix[queryIndex][j] = superMatrix[queryIndex][j];
                }
            }
        }
    }
    /*
     * public void search() { double[][] simMeasureMatrix = new
     * double[numDocs][numDocs]; int[] similarityFreq = new int[numDocs];
     * ArrayList<Integer> similarityFreqList = new ArrayList<Integer>(); for
     * (int i = 0; i < numDocs; i++) { double[][] matrix1 = tensorRefined[i];
     * for (int j = 0; j < numDocs; j++) { if (j < i) { simMeasureMatrix[i][j] =
     * simMeasureMatrix[j][i]; } else { double[][] matrix2 = tensorRefined[j];
     * simMeasure sim = new simMeasure(); sim.setMatrix1(matrix1);
     * sim.setMatrix2(matrix2); simMeasureMatrix[i][j] = sim.measure(); }
     *
     * if (simMeasureMatrix[i][j] >= alpha) { similarityFreq[i]++; } }
     *
     * if (similarityFreqList.isEmpty()) {
     * similarityFreqList.add(similarityFreq[i]);
     * results.add(conceptSpace.get(i).docPath); } else if
     * (similarityFreqList.get(similarityFreqList.size() - 1) >
     * similarityFreq[i]) { similarityFreqList.add(similarityFreq[i]);
     * results.add(conceptSpace.get(i).docPath); } else { for (int j = 0; j <
     * similarityFreqList.size(); j++) { if (similarityFreqList.get(j) <=
     * similarityFreq[i]) { similarityFreqList.add(j, similarityFreq[i]);
     * results.add(j, conceptSpace.get(i).docPath); break; } } } }
     */

    public static void main(String[] args) {
        ArrayList<String> docPath = new ArrayList<String>();
        HashMap<String, Integer> terms = new HashMap<String, Integer>();
        docPath.add("document1");
        docPath.add("document2");
        docPath.add("document3");
        docPath.add("document4");
        docPath.add("document5");
        terms.put("apple", 0);
        terms.put("banana", 1);
        double[][][] tensorRefined = { { { 1, 2, 3 }, { 4, 5, 6 } }, { { 1, 2, 3 }, { 13,
        14, 15 } },
                { { 1, 2, 3 }, { 22, 23, 24 } } };
        double[][] superMatrix = {{1,2,3},{13,14,15}};
        String[] query = {"apple", "banana"};
```

```
        SearchModule sm = new SearchModule();
        sm.setDocPath(docPath);
        sm.setTensorRefined(tensorRefined);
        sm.setQuery(query);
        sm.setSuperMatrix(superMatrix);
        sm.setTerms(terms);
        sm.buildQuerySuperMatrix();
        sm.search();
        for (int i = 0; i < sm.getQuerySuperMatrix().length; i++){
            for (int j = 0; j < sm.getQuerySuperMatrix()[i].length; j++){
                System.out.print(sm.getQuerySuperMatrix()[i][j]+" ");
            }
            System.out.println();;
        }
        for (int i = 0; i < sm.getResults().size(); i++) {
            System.out.println(sm.getResults().get(i));
        }

    }

}
```

F. Table of Stop Words and Symbols

	-	are	during	i	once	theirs	we've	yourself	theyre
--	(	aren't	each	i'd	only	them	were	yourselves	theyve
!!	)	as	few	i'll	or	themselves	weren't	###	wasnt
?!	&	at	for	i'm	other	then	what	return	were
??	%	be	from	i've	ought	there	what's	arent	werent
!?	$	because	further	if	our	there's	when	cant	whats
`	@	been	had	in	ours	these	when's	couldnt	whens
``	!	before	hadn't	into	ourselves	they	where	didnt	wheres
"	^	being	has	is	out	they'd	where's	doesnt	whos
-lrb-	#	below	hasn't	isn't	over	they'll	which	dont	whys
-rrb-	*	between	have	it	own	they're	while	hadnt	wont
-lsb-	..	both	haven't	it's	same	they've	who	hasnt	wouldnt
-rsb-	...	but	having	its	shan't	this	who's	havent	youd
,	'll	by	he	itself	she	those	whom	hes	youll
.	's	can	he'd	let's	she'd	through	why	heres	youre
:	'm	can't	he'll	me	she'll	to	why's	hows	youve
;	a	cannot	he's	more	she's	too	with	im	thou
"	about	could	her	most	should	under	won't	isnt	us
'	above	couldn't	here	mustn't	shouldn't	until	would	its	theyre
?	after	did	here's	my	so	up	wouldn't	lets	theyve
<	again	didn't	hers	myself	some	very	you	mustnt	wasnt
>	against	do	herself	no	such	was	you'd	shant	were
{	all	does	him	nor	than	wasn't	you'll	shes	werent
}	am	doesn't	himself	not	that	we	you're	shouldnt	whats
[	an	doing	his	of	that's	we'd	you've	thats	whens
]	and	don't	how	off	the	we'll	your	theres	wheres
+	any	down	how's	on	their	we're	yours	theyll	whos

Curriculum Vitae

JOHN MARK ROBERT M. AGSUNOD

#11-C Saleng Street, Veterans Village, Project 7, Q.C.
(+632) 414-4143, 0915-320-2435
jmagsunod.main@gmail.com, jmagsunod53@gmail.com
www.linkedin.com/in/jm-agsunod-972915108

EDUCATION

Bachelor of Science in Computer Science
Institute of Information and Computing Sciences
University of Santo Tomas, España, Manila, Philippines
July 2014 - May 2018

High School Graduate
Quezon City Science High School
Bago-Bantay, Project 6, Quezon City, Philippines
June 2010 - March 2014

Elementary School Graduate
Our Lady of Montichiari School
Lanete Street, Project 7, Quezon City, Philippines
June 2004 - March 2010

CO-CURRICULAR ACTIVITIES

Staff, Academics Committee
Institute of Information and Computing Sciences Student Council, University of Santo Tomas, España, Manila
Member | August 28, 2017 – Present

Staff, Publicity Committee
Computer Science Society, University of Santo Tomas, España, Manila
Member | August 2016 – May 2017

Staff, Logistics Committee
Computer Science Society, University of Santo Tomas, España, Manila
Member | August 2015 – May 2016

Member, Microsoft Student Community, University of Santo Tomas, España, Manila | 2016 - 2017

Career Ambassador, IICS Counseling & Career Center, University of Santo Tomas, España, Manila | 2015 - Present

EXTRA-CURRICULAR ACTIVITIES

Junior Officer, Community Achievers Association Engineering Division, University of Santo Tomas, España, Manila | 2014 – 2015

ACHIEVEMENTS RECEIVED

Class Secretary (January 2017 – June 2017), Class Public Relations Officer (2015 – 2016), Class President (2013 – 2014), Best in Zoology Award (2014), 2nd Honorable Mention (2010), Best in Science & Filipino (2010), Honor Student (2004 – 2010)

TRAININGS/SEMINARS ATTENDED

UI/UX Seminar: Design to Break Barriers, University of Santo Tomas, España, Manila, October 27, 2017

C# Seminar, University of Santo Tomas, España, Manila, 2015

Rotaract Leadership Seminar, University of Santo Tomas, España, Manila, 2015

Game Etiquette Seminar, University of Santo Tomas, España, Manila, 2015

ThomThinks: Think to Create, University of Santo Tomas, España, Manila, January 7, 2015

.NET Seminar, University of Santo Tomas, España, Manila, April 5, 2016

Microsoft Network Labs Seminar, University of Santo Tomas, España, Manila, April 5,2016

Start-Up Seminar, University of Santo Tomas, España, Manila, April 6, 2016

SKILLS & ABILITIES

Languages: Filipino (Tagalog), English (American and British)
Operating Systems: Microsoft Windows, Apple/Macintosh, Android
Programming & Web Page Development (knowledge rating shown out of 10):

Java – 5/10
C & C++ - 5/10
MySQL – 4.5/10
Prolog – 3/10
HTML – 8/10
CSS – 6/10
Javascript – 3/10
Assembly Language – 5/10

Other Technical Skills: Microsoft Office 2016, Prezi 3.0.3, Gmail, Yahoo Mail, Sony Vegas Pro, Adobe Photoshop, Adobe Illustrator

CARL JAYSON M. BANTING

Kamias Alley, Balingasa, Balintawak, Quezon City
448-59-94 ,0995-728-5720
carl.jayson397@gmail.com
https:/ph.linkedin.com/in/carl-jayson-banting-562339140/

EDUCATIONAL BACKGROUND

Tertiary

Bachelor of Science in Computer Science *(2014-2018)* Institute of Information and Computing Sciences University of Santo Tomas, España, Manila, Philippines May 29, 2018

Secondary

Siena College, Quezon City *(2010-2014)*
Del Monte Ave, Quezon City, Metro Manila

Mater Carmeli School, Quezon City *(2004-2010)*
D. Tuazon, Quezon City

CO-CURRICULAR ACTIVITIES

Computer Science Society, Institute of Information and Computing Sciences, University of Santo Tomas Member, August 2014- present

AWARDS RECEIVED

IBM DB2 Academic Associate, IBM, Manila, November 2015

Dean's List *(A.Y. 2014-2015, 1st Semester)*, UST, Manila, January 2015

TRAININGS/SEMINARS ATTENDED

Think Before You Click, Siena College Student Council, Siena College, Siena Hall, 2014

Microsoft .Net Seminar 2016, Institute of Information and Computing Sciences, Room 49, 4th floor of Roque Ruano Bldg., UST, April 5, 2016

Microsoft Network Labs Seminar, Institute of Information and Computing Sciences, Medicine Auditorium, UST, April 5, 2016

Start-up Seminar, Institute of Information and Computing Sciences, Room 45, 4th floor of Roque Ruano Bldg., UST, April 6, 2016

Adobe Photoshop Bootcamp, Technokids, Lab 1, 3rd floor of of Roque Ruano Bldg., UST, April 4, 2017

SKILLS & ABILITIES

- Programming: Skilled in programming in Java and C. Familiar with PROLOG and MySQL. Also skilled in MVC and Struts 2 architecture. Familiar with Hibernate architecture.
- Computer: Experienced in MS Office specifically MS Word, MS Powerpoint, and MS Excel.
- Language: Has good communication skills using both English and Filipino language.

Harjit S. Brar

117 C k-4th Kamuning street, Q.C
0919-809-53-11 / 0977-461-49-86
Harjitbrar07@gmail.com
https://www.linkedin.com/in/harjit-brar-917ab0131/

EDUCATION

Bachelor of Science in Computer Science
Institute of Information and Computing sciences
University of Santo Tomas, España, Manila, Philippines
July 2014 - May 2018

High School Graduate
Immaculate Conception Cathedral School
39 Lantana Street, Cubao, Quezon City
June 2010 - March 2014

Elementary School Graduate
Immaculate Conception Cathedral School
39 Lantana Street, Cubao, Quezon City
June 2004 - March 2010

CO-CURRICULAR ACTIVITIES

Computer Science Society
Member, August 2014 – present

AWARDS RECEIVED

IBM DB2 Academic Associate, University of Santo Tomas, Manila, 2015

Dean's List awardee, University of Santo Tomas, Manila, 2014

Dean's List awardee, University of Santo Tomas, Manila, 2015

Honor student, Immaculate Conception Cathedral school, Q.C,2010 and 2012 – 2013

TRAININGS/SEMINARS ATTENDED

C# Seminar, 2015, University of Santo Tomas

Rotaract Leadership Seminar, 2015, University of Santo Tomas

Game Etiquette Seminar, 2015, University of Santo Tomas

.NET seminar, 2016, University of Santo Tomas

Start up, 2016, University of Santo Tomas

Microsoft Network Labs Seminar, 2016, University of Santo Tomas

SKILLS & ABILITIES

- Programming: Skilled in programming in C, java, MVC architecture, and Struts2 architecture; familiar with XML, SQL, and Hibernate architecture.
- Computer: Experienced in using MS office.
- Analytics: Skilled in algorithm design, implementation, and Optimization.
- Leadership: Led and worked with groups in research projects and software development projects.
- Communication: Fluent speaking skills obtained through professional environments, and organizations.
- Language: Fluent in English, basic reading, writing, and speaking skills in Tagalog; familiar with Hindi.

PATRICK BRYAN F. CUNANAN

#55 Sampaguita St., Sta. Lucia Village,
Veinte Reales, Valenzuela City
09332123116
patrick.cunanan15@gmail.com, 2014069410@ust.edu.ph
https://www.linkedin.com/in/pcunanan/

EDUCATION

Bachelor of Science in Computer Science
Institute of Information and Computing Sciences
University of Santo Tomas, España, Manila, Philippines
August 2014 - May 2018

High School Graduate
Meycauayan College, Meycauayan City, Bulacan
June 2010 - March 2014

Elementary School Graduate
Meycauayan College, Meycauayan City, Bulacan
June 2005 - March 2010

WORK EXPERIENCE

Staff, Creatives and Technical Committee, Computer Science Society
University of Santo Tomas, España, Manila, 2014 - 2016

Member, Junior Philippine Computer Society
University of Santo Tomas, España, Manila, 2015 - 2017

Poster Designer, #ICSPERIENCE IICS Freshmen Week Event
University of Santo Tomas, España, Manila, August 2014

Poster Designer, Picture Perfect: Basic Photography and Photo Editing Seminar
University of Santo Tomas, España, Manila, January 23, 2016

Freelance Designer
2015

TRAININGS/SEMINARS ATTENDED

C# Seminar, University of Santo Tomas, España, Manila, 2015

ThomThinks: Think to Create, University of Santo Tomas, España, Manila, January 7, 2015

.NET Seminar, University of Santo Tomas, España, Manila, April 5, 2016

Networklabs Seminar, University of Santo Tomas, España, Manila, April 5, 2016

Start-Up Seminar, University of Santo Tomas, España, Manila, April 6, 2016

Specialization Tracks Seminar, University of Santo Tomas, España, Manila, April 6, 2016

CERTIFICATIONS AND ACHIEVEMENTS

IBM DB2 Academic Associate, November 2015

Overall Champion, Human-Computer Interaction Project Defense, University of Santo Tomas, España, Manila, December 2015

Round 2 Finalist (Team Member), Microsoft Imagine Cup Philippines 2016, University of Santo Tomas, España, Manila, February 2016

SKILLS & ABILITIES:

Programming: Java, C, MySQL, Python

Web Designing & Development: HTML, CSS, JavaScript, JSP, Basic PHP

Photo Editing: Adobe Photoshop CS6

Video Editing: AVS Video Editor

Audio Editing: Audacity

Other Technical Skills: Microsoft Office, WPS Kingsoft Suite, LibreOffice