Spring 2012

Knowledge Engineering in Search Engines

Yun-Chieh Lin
San Jose State University

Knowledge Engineering in Search Engines

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment
of Requirements for the Degree
Master of Science

by
Yun-Chieh Lin

Fall 2011

ABSTRACT

With large amounts of information being exchanged on the Internet, search engines have become the most popular tools for helping users to search and filter this information. However, keyword-based search engines sometimes obtain information, which does not meet user' needs. Some of them are even irrelevant to what the user queries. When the users get query results, they have to read and organize them by themselves. It is not easy for users to handle information when a search engine returns several million results. This project uses a granular computing approach to find knowledge structures of a search engine. The project focuses on knowledge engineering components of a search engine. Based on the earlier work of Dr. Lin and his former student [1], it represents concepts in the Web by simplicial complexes. We found that to represent simplicial complexes adequately, we only need the maximal simplexes. Therefore, this project focuses on building maximal simplexes. Since it is too costly to analyze all Web pages or documents, the project uses the sampling method to get sampling documents. The project constructs simplexes of documents and uses the simplexes to find maximal simplexes. These maximal simplexes are regarded as primitive concepts that can represent Web pages or documents. The maximal simplexes can be used to build an index of a search engine in the future.

Table of Contents

List of Figures

1.0 INTRODUCTION

The Internet has grown so fast, and it feels as if it contains almost everything. If people want to travel, they search online to find the most recommended place. If people want to research a field of study, they can get a lot of information available on the internet from all over the world. People are able to search almost any information from the Internet. Since the information is available at one's fingertips and searching information on the internet is a lot more convenience than it used to be, this introduces an issue of information overload. In addition, there is another issue of getting the right information that people are looking for since the specific information can be buried amongst irrelevant information.

Search engine tools help users to filter and extract the desired information. They can provide Web pages or documents that are relevant to user queries in a fraction of a second. Although the results have been filtered, some of them still do not match what users are looking for. The problem is that machines cannot understand the meaning implied in content. These keyword-based search engines return results that match keywords of user inputs instead of latent semantics. This causes poor efficiency in keyword searches.

Each document consists of terms or tokens. A high-frequency token in a document is called a keyword and a high-frequency co-occurring set of keywords is called a keyword set. The keywords and keyword sets represent concepts in documents. These concepts can form a simplicial complex of concepts. They can be used as an index of a knowledge-based search engine.

This project uses a granular computing approach to analyze documents and Web pages. The purpose is to find primitive concepts. Granular Computing is an emerging technology. It is a subset of granular mathematics and makes use of granularity in problem solving. A simplicial complex is the second granular computing model. This model can visualize text processing in a geometric way. In text processing, a high frequency keyword sequence is regarded as an ordered simplex.

2.0 THEORY

2.1 Information Retrieval

Information retrieval has two processes, storage, and retrieval. In the overall process, one first collects the vast and disordered information on the Internet or elsewhere. This is the storage phrase. Then one organizes and classifies it in a certain way to make the information ordered and systematic. This ordered data can then be used to build a database. Retrieval is the process that utilizes retrieval systems or tools to search information that can be potential queries. In the information retrieval field, there are three types of techniques, text retrieval, data retrieval, and knowledge retrieval. The following briefly describes the three types:

1. Text Retrieval: In this system, one parses whole statements and directly compares word by word.

   Advantage: High recall

   Disadvantages: Easily misleading semantics, low precision, and non-semantic

2. Data Retrieval: In this system, one has a structured data store and specific schema query.

   Advantage: High-Speed

   Disadvantages: Low semantics

3. Knowledge Retrieval: In this system, one returns information based on knowledge matches.

Advantage: High recall and high precision

Disadvantage: The constructive process is more complicated

Knowledge retrieval is better than the others are for information retrieval since it has high recall and high precision. This project uses new algorithms to build a knowledge base for this information retrieval purpose.

2.2 TEXT PROCESSING TECHNOLOGY

2.2.1 Tokenization Process

In order to analyze each document to get its main idea, the document has to be converted to a list of terms. Tokenization is used to extract words from documents and gets rid of spaces and punctuations such as period, commas, or semicolons. The reason is that those symbols are irrelevant since a document turns into a set of terms instead of sentences.

2.2.2 Keywords Extraction

Term Frequency – Inverse Document Frequency, TFIDF, is an often-used statistical method in the vector space model. The term "Frequency" means the total number of times a term appears in a document. Document Frequency means the total

number of times a term appears in all documents. The idea is to use term frequency to weigh a keyword in a document. If the frequency is higher than for other keywords, the term could represent more important concepts in the document. However, some terms have high term frequency but may not be relevant. Those terms, such as the subject or a conjunction are too common to represent a concept. Although the terms have high frequency, they do not make documents distinguishable. Therefore, if a term has high term frequency and low document frequency, it is more representative of the document. The following is the format of TFIDF:

1. Term Frequency (TF)

   $tf_{ij} = \frac{Nj}{Nall}$

   Nj: total number of term j in document i

   Nall: total number of terms in document I

2. Inverse Document Frequency (IDF)

   $idf_j = \log_2 \frac{d}{dfj}$

   d: total number of documents

   df_j: total number of documents that has term j

3. TFIDF

   $W_{ij} = TF \times IDF = \frac{Nj}{Nall} \times \log_2 \frac{d}{dfj}$

   W_{ij}: weight of term j in document i

After computing weights of all terms in every document, each document now has TF-IDF values of terms. These values are used to extract keywords.

2.3 Granular Computing

Granular Computing is an emerging technology and computing paradigm of information processing. It is a subset of granular mathematics and makes use of granularity in problem solving. Granules are the ingredients of granular computing; they are subset, clusters, and classes of the universe. In our daily lives, many things are granulated into sub-things. For instance, the human body is granulated into parts such as hand, leg, and head. A faculty in a computer department is a granule. Each faculty member may play different roles.

When people process massive and complicated information, they divide the information into small simple parts. Each part is regarded as a granule. In fact, granules are parts that are formed by individuals such as a point, an element through proximity relation, a similarity relation, or a functional relation. This information processing is called information granulation.

Granular computing turns a complicated problem into several simpler problems. It helps people get better analysis and problem solving. Granular computing has been applied to many other fields, such as cluster analysis, information retrieval, databases, machine learning, and concept formation.

The next few sections will introduce structures of granular computing, (e.g. simplicial complex model, keyword simplicial complex model, and the primitive concept). The last section illustrates the intersection of simplexes.

2.3.1 Structures of a Granular Computing Model

A granular computing model has four structures. These structures are granular structure, quotient structure, knowledge structure, and linguistic structure.

1. **Granular Structure (GrS)**: It is the collection of all granules. In the case of partition, GrS is the collection of the equivalence classes.
2. **Quotient Structure (QS)**: If each granule is abstracted into a point and the intersections of granules are abstract to the interactions of points, then such a collection of points is called the quotient structure. In the case of partition, the quotient structure is a classical set, called quotient set . The process of abstracting granular structure into quotient structure is called information hiding.
3. **Knowledge Structure**: By giving each granule (point) in the quotient structure a meaningful symbol, the named quotient structure is called knowledge structure . The knowledge structure provides an intuitive view of the quotient structure; the symbols and interaction among symbols are in sync with the granules (points) and interactions among granules(points) . In the case of *n* partitions (equivalence relations), the knowledge structure can be arranged into a *n*-column relational table .
4. **Linguistic structure**: By assigning each granule in the granular structure a word that reflects its meaning . The interactions among these words are reflected implicitly in precisiated natural language. (in knowledge structure, the interactions among symbols are explicitly reflected from the quotient structure) . The linguistic structure is the domain of computing with words.

(Lin, Hsu, 2008)

2.3.2 Simplicial Complex

A simplicial complex is the second granular computing model. This model can be interpreted in a geometric way. By using the Cartesian product of n sets, elements can be addressed in an n-dimensional Euclidean space. This is used to denote n-simplex. A simplex that contains (n+1) vertices is called n-simplex.

For instance, if n = 0, a 0-simplex is formed by a vertex. If n = 1, a 1-simplex is formed by two vertices. If n=2, a 2-simplex is formed by three vertices. If n=3, a 3-simplex is formed by four vertices. In other words, an n-simplex can be constructed when an (n-1)-simplex connects to a new vertex. An m-subset of n-simplex is called an m-face. Take a 3-simplex for example; it has four 0-faces, six 1-faces, four 2-faces, and one 3-face. Figure 1 shows the geometric shapes of these four simplexes.

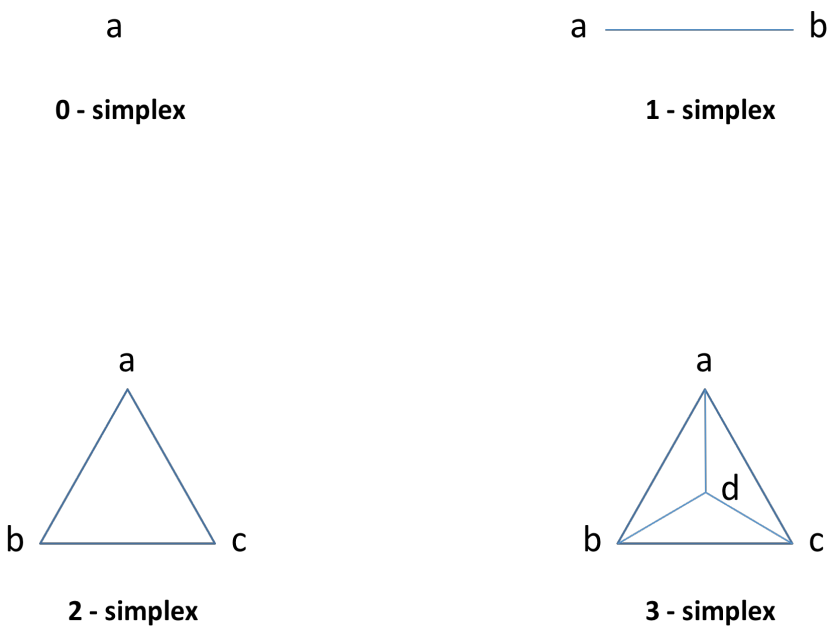

Figure 1. Example of four geometric shapes

A combination of simplexes can form a simplicial complex. A simplicial complex consists of vertices and simplexes. If a simplex is called a maximal simplex, it means that the simplex is not a face of other simplexes. In a simplicial complex, if a k-simplex is the maximal simplex in the complex, the complex is called a k-complex.

Figure 2 is an example of a hyper-graph of the simplicial complex. This simplicial complex contains eleven vertices and the highest dimension of the simplex is three. It is a 3-complex. In this project, the main concern is to find maximal simplexes in complexes. The reason is that these maximal simplexes are applied to obtain primitive concepts of a set of documents.

Maximal simplexes in Figure 2:

1. Two 3-simplex,$\triangle$(a,b,c,d) and $\triangle$(e,g,h,f)
2. Five 2-simplex,$\triangle$(b,c,k),$\triangle$(c,j,k), $\triangle$(j,k,i), $\triangle$(j,i,f) , and $\triangle$(i,f,g)

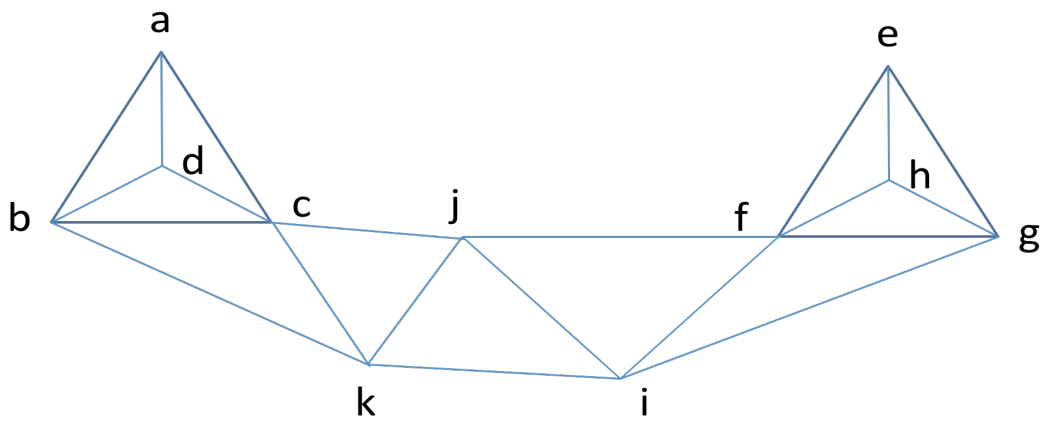

Figure 2. An example of a simplicial complex

2.3.3 Knowledge Complex

According to the last section, a simplicial complex model is applied to linear text in this project. Vertices are keywords and edges are distances between two vertices. However, the main different between a simplicial complex and a knowledge complex is that a simplicial complex has no vertex order. In a text file, term order is important. Different orderings of the terms may present different concepts or no meaning. Therefore, when two keyword sets have exactly the same keywords but in different order, they are considered two simplexes.

2.3.4 Primitive Concept

A 0-simplex as a keyword might contain different concepts. If a term combines to other terms, it would present clearer concepts. For example, "Network" might be present in many fields. "Traffic, Network" or "Neural, Network" are the combinations of a pair of keywords. The 2-simplexes such as "Biological, Neural, Network" or "Artificial, Neural, Network" denote further semantics.

A set of given documents may contain a set of concepts. Each concept consists of a connected component of the simplex. Humans use articles to present their ideas or thoughts. The idea consists of many concepts. Some of them consist of primitive concepts. A primitive concept is regarded as a maximal simplex with the highest dimension. Faces of a primitive concept are called sub-concepts. The maximal simplex is not a part of other simplexes in the complex. There are many concepts in a

document. This project proposes to find primitive concepts of a set of documents and uses them to form a knowledge base.

2.3.5 Simplex Intersection

As previous sections mentioned, primitive concepts are also called maximal simplexes. The characteristic of a maximal simplex is that it is not any other simplex's face in the simplicial complex. A simplicial complex consists of simplexes. Each of them is a maximal simplex.

This section introduces the relation between two maximal simplexes. This project considers the intersection between two simplexes to be the relation between two simplexes. The intersection could be a 1-simplex, 2-simplexes.... The intersection is formed when two keyword sequences are shared with a keyword sequence. For example, a keyword sequence is a simplex, and keyword components of a simplex are regarded as vertices. Figure 7 shows two examples of the intersection of two simplexes.

The following are the steps to find the intersection of every two simplexes.

Step 1: Get all keywords and keyword sequences of the knowledge database from the longest sequences to single keywords.

Step 2: Compare every two of them to find the common keyword components. If they have common keywords, these keywords are the intersection.

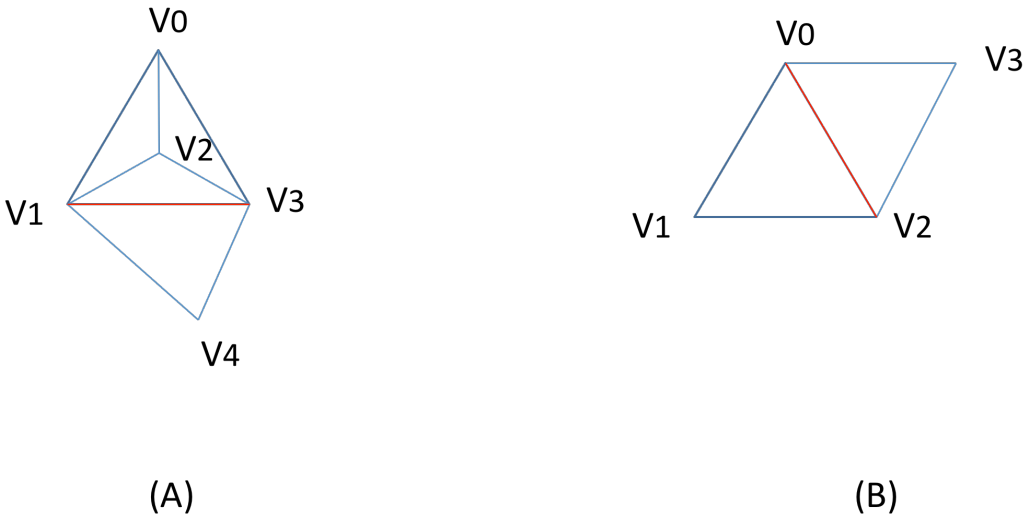

Figure 3. Examples of an intersection of two simplexes (A) 3-simplex and 2simplex (B) 2-simplex and 2-simplex

Figure 3 (A): there is a 3-simplex (V_0, V_1, V_2, V_3) and a 2-simplex (V_1, V_3, V_4). These two simplexes have a common 1-simplex (V_1,V_3) .

Figure 3 (B): there is a 2-simplex (V_0, V_1, V_2) and a 2-simplex (V_0, V_2, V_3). These two simplexes have a common 1-simplex (V_0,V_2) .

2.4 SAMPLING

In social science research, usually there is not enough time to collect the data of a whole population. To do this is to select samples, which are parts of a population, through a sampling method. These property samples can refer to the population property by using inferential statistics. They have the same characteristics as the population. This sampling technique is used to handle large volumes of data. Even if the number of the population cannot be estimated, it is still possible to refer to the parameters of the population in a fixed quantity range.

2.4.1 Sampling Method

The sampling method has two classes. The first one is probability sampling and the second is non-probability sampling. In probability sampling, every individual of the population has a certain probability to be selected to research samples. In non-probability sampling, the relation of probability between samples and the population is unknown. There are four commonly used probability sampling methods, simple random sampling, systematic sampling, stratified sampling, and cluster sampling.

The following illustrates these methods of probability sampling.

1. Simple random sampling, SRS: Every individual of the population has the same probability to be selected to samples. This method is simple and fair and can use statistical theory directly to refer and estimate. To do random sampling, first

   we need to obtain a sampling frame of the population. Second, a random generator uses the list to generate some random numbers. Each of these random numbers has corresponding individuals. The corresponding individuals are regarded as samples.
2. Systematic sampling: It is a simplified random method. The most common way is to select samples according to a fixed interval from a sampling frame of the population. For example, the total population is 500 and the distance of intervals is 10. The method has to choose a sample from every interval. First,

a random number from 1 to 10 is generated in an interval. If the number is 3, it is the first sample. To compute the rest of the samples, the number adds 10 every time. The samples should be 3, 13, 23, 33, 43, 53… untill the last one.

3. Stratified sampling: this method is more accurate than SRS. According to some related conditions, individuals of the population are separated into different strata. Certain individuals can be selected from these separating strata. These individuals from the population are samples.
4. Cluster sampling: According to some characteristic of the parent population, individuals are separated into different clusters. The method randomly selects some of these clusters and uses they to select samples.

2.4.2 Document Sampling

This project is to design a knowledge base system. Selecting a collection of documents becomes important. Those documents should be randomly selected from Web pages or library resources. If documents belong to some specific topics, the knowledge base would not be useful since it only contains primitive related concepts.

A set of documents that are selected for analysis should contain more topics. However, usually there is too much information. If we want to build a knowledge base for a search engine, it is not feasible to go through all of the available documents. This type of data processing is usually too hardware intensive and not very efficient. In addition, collecting all of the available information from the Internet is

currently impossible.

The document sampling method solves the problem and helps to reduce computing loads. It is a way of dealing with massive number of Web pages or documents. Use a sampling method to select Web pages or documents; these samples represent general knowledge. If the number of samples is large enough and chosen correctly, those document samples can obtain most fields and provide full concepts.

3.0 Design

This system is used to capture maximal simplexes of each document and forms a knowledge base. This section introduces the system architecture and discusses each step in detail.

3.1 System Architecture

This system first analyzes each document and uses tokenization to extract terms. After parsing the document into terms, it removes terms that are stop words. The system then computes the weight of the rest of the terms to get document keywords. These keywords can be formed pair-keyword sets by computing their distances and document frequency. Using the algorithm, two paired keyword sets can be combined to a three-keyword set, two three-keyword sets can be combined to a four-keyword sets, and two four-keyword set can be combined to a five-keyword set. Finally, those keyword sets are used to eliminate partial concepts and find maximal simplexes of documents. The following is the flowchart of system architecture:

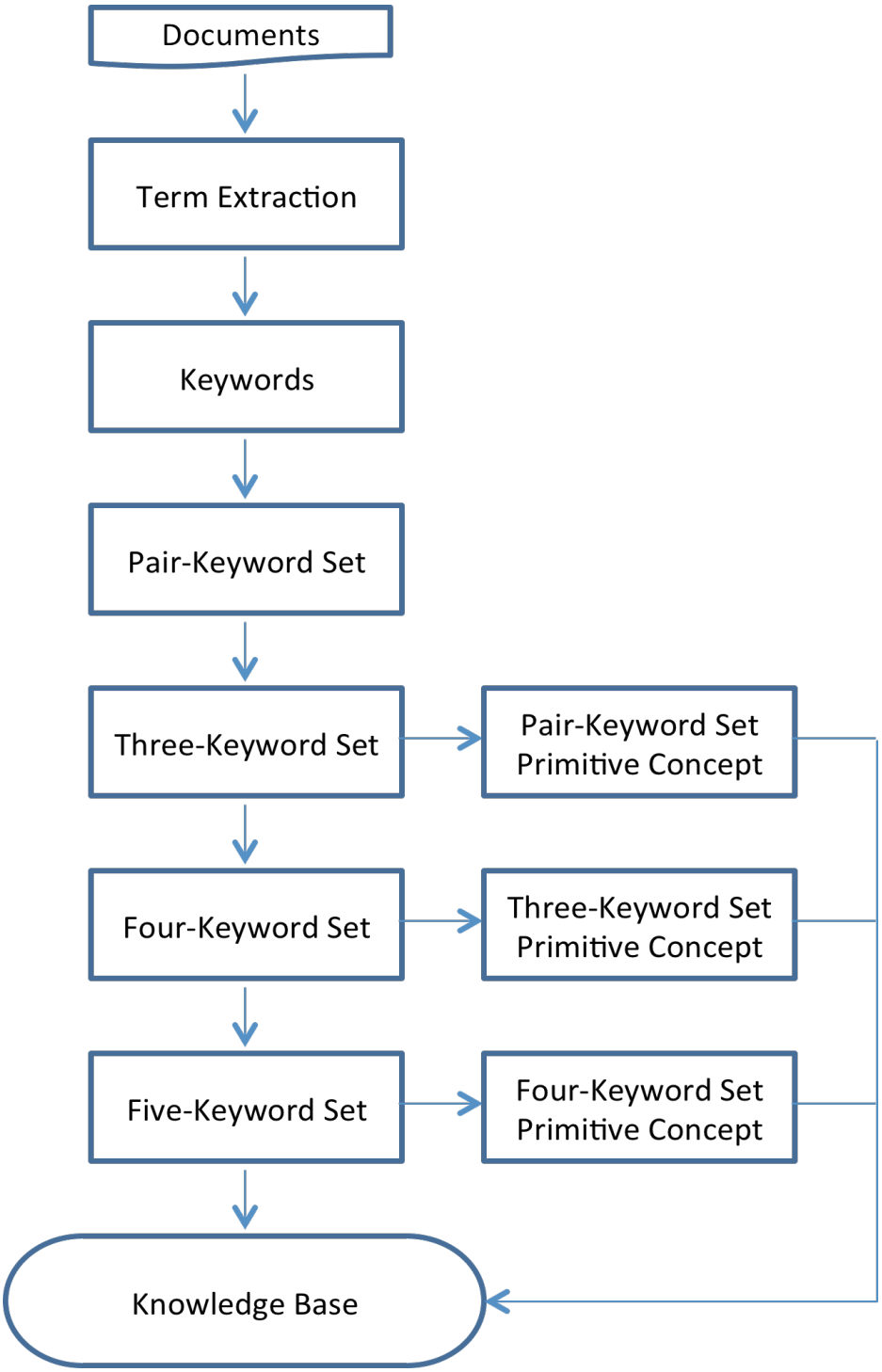

Figure 4. Flowchart of system Architecture

3.2 Term Extraction

To analyze documents, each document needs to be considered a list of ordered terms or tokens. Every character of a document is read during tokenization. If the character is a letter from an alphabet, the system keeps it and goes to the next character. This process will not stop until the next character is non-alphabetic. Once it stops, the previous alphabetic characters are regarded as a token and they go into a token list.

Second, if a line of the document is a blank line or the first character is a paragraph mark, all terms between these lines are selected as a group. They are marked the same paragraph number. These paragraph numbers are regarded as distances between terms that are also used to determine a relationship between two terms. Each document has a table, it contains three attributes, the first one is a Document Identifier, the second is a Token, and the last one is a Paragraph Number. They are used for computing keywords and keyword sets.

3.3 Keyword Computing

Keywords are meaningful words. In a linguistics aspect, keywords can describe an article's main subject. In the information retrieval field, keywords are interpreted terms that can represent an article. Every article has its own keywords to define its subject. An article can be distinguished from others by using keywords.

Stop words are words that do not contain important meanings. They are usually common words or function words such as "are", "on", "is", or "the". After the

tokenization process, every document contains some stop words. These terms should be removed since stop words do not provide important meaning. Figure 2 shows the flowchart of keyword processing.

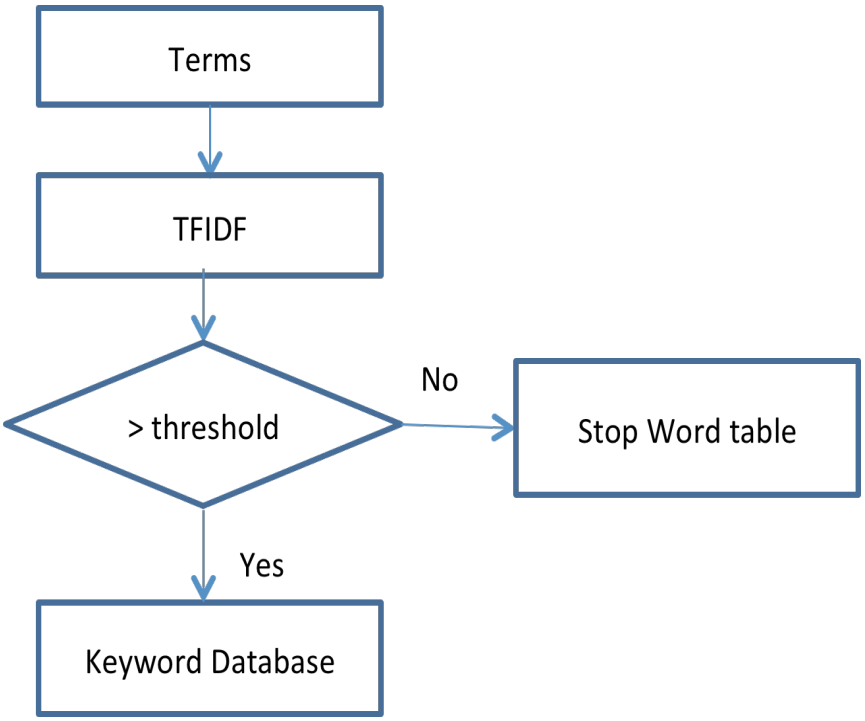

Figure 5. Flowchart of keyword processing

To extract keywords from each document, every document needs to go through this procedure.

Step 1: To calculate weights of terms, use the TFIDF method denoted in section 2.2.2.

Step 2: Decide the keywords according to a threshold that is the default value by

the system. If the weight of a term is greater than the threshold, the system selects the term for the keyword database. Otherwise, it defines the term as a stop word.

These terms are inserted into different tables according to corresponding documents. The table contains four attributes: a document identifier, a token, a position, and a paragraph number of tokens. They are used for the next step to compute keyword sets.

The table of stop words contains terms that are all less than the threshold. These terms are regarded as a list of stop words in this project. They are used to extract keywords from a new document. This will be illustrated in section 3.6.

3.4 Keyword Sequence Computing

Keywords are retrieved from a database to form a keyword set, which can consist of keywords from the same paragraph. In this project, a keyword set is ordered. It is also called a keyword sequence. This section will describe how to form paired keyword sequences, three-keyword sequences, four-keyword sequences, and five-keyword sequences in detail. The section also describes how to use keyword sequences of high-dimension to eliminate sub-keyword sequences.

3.4.1 Pair-Keyword Sequence

From each table of document keywords, every keyword has its own position and paragraph number. These two attributes are the key to assembling pair-keyword sequences. The positions of keywords are used for ordering keyword sequences. As the previous section mentioned, a pair-keyword sequence is regarded as a 1-simplex in a complex. The system defines an edge as a paragraph length. In other words, two keywords are only considered combining only if they are in the same paragraph. Figure 5 is the process of forming pair-keyword sequences.

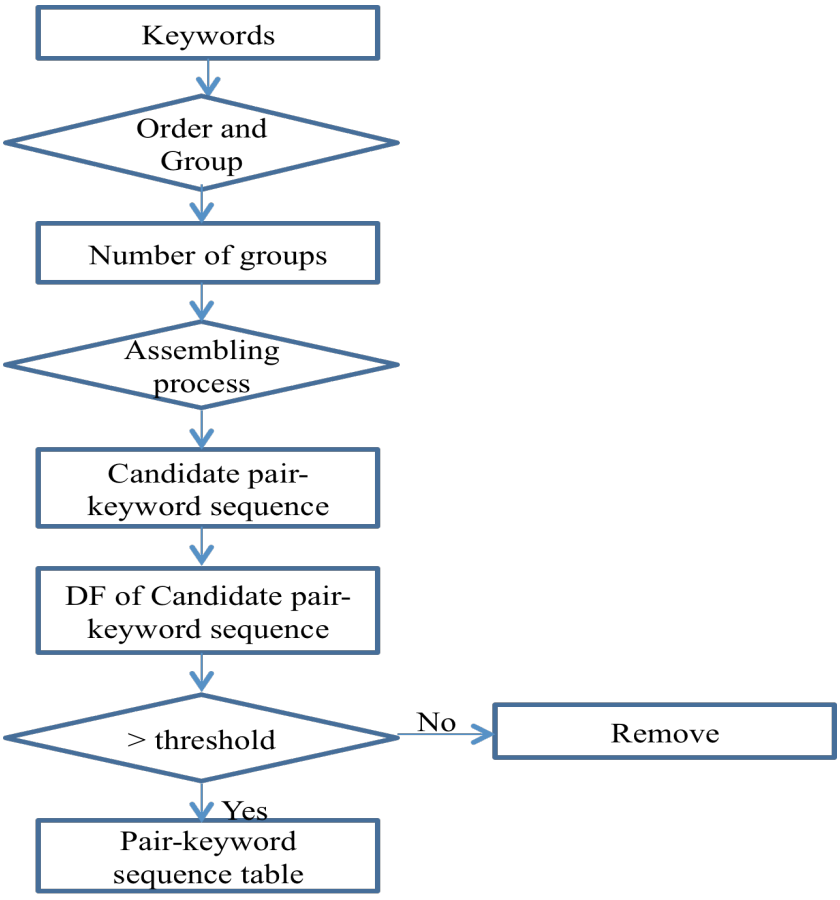

Figure 6. Flowchart of the process of pair-keyword sequencing

The following are steps in the computing process. These steps illustrate the process of computing pair-keyword sequences in detail:

Step 1: Order keywords according to their positions and group them by paragraph number.

Step 2: According to these groups, keywords in the same paragraph are combined with each other by their positions. For example, “social”, “structure”, “network” can be combined to (“social”, “structure”), (“social”, “network”), and (“structure”, “network”). Combinations of two keywords are called paired keyword sequence candidates.

Step 3: Compute document frequency. The pair-keyword sequence needs to be checked in all of the documents. It only counts when the two keywords are in the same paragraph of the document. If these two keywords appear in the same document but in different paragraphs, they do not count as a pair-keyword sequence. The DF value is used to decide whether the sequence meets a threshold.

Step 4: If the value is greater than the threshold, this sequence is selected for the table of pair-keyword sequences. If not, the system removes the paired keyword sequence candidate.

A table of pair-keyword sequences contains three attributes, they are the document identify, pair-keyword1, and pair-keyword2. These tables provide information

for computing three-keyword sequences. Pair-keyword sequences that are in the tables present some concepts. These concepts are used to form three-keyword sequences. The process will be described in the next section.

3.4.2 Three-Keyword Sequence

To form three-keyword sequences of a document, the system checks the document's paired keyword sequences but not keywords. The reason is that existent keywords are subsets of paired keyword sequences. These keywords are already examined during computing paired keyword sequence. In other words, a combination of (n-1)-keyword sequence can form an n-keyword sequence.

3.4.2.1 Three-keyword Sequence Computing

It is more complicated to form a three-keyword sequence than a paired keyword sequence. To generate a real three-keyword sequence, many conditions need to be checked. After generating three-keyword sequence candidates, these candidates then go through the processes to generate real three-keyword sequences. (See the flowchart in Figure 5):

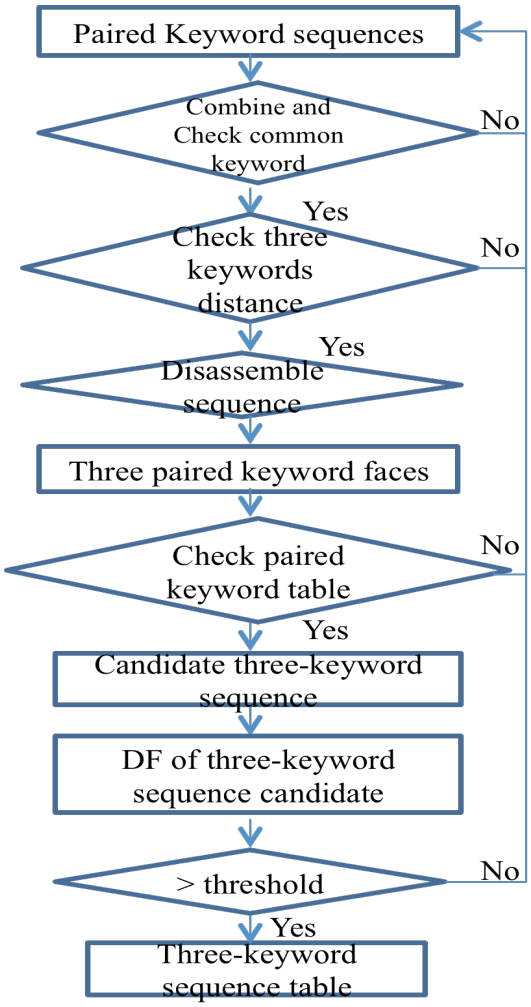

Figure 7. Flowchart for computing three-keyword sequences

The following steps illustrate these procedures in detail:

Step 1: Obtain a list of paired keyword sequences and check every two of these paired keyword sequences. If the combination of the two paired keyword sequences have one matching keyword element, this ordered combination is considered a temporary three-keyword sequence. Otherwise, the system ignores any combination that has more

than three elements

Step 2: The paragraph number of each keyword in a temporary three-keyword sequence needs to be examined. This process is to check if the edges between the temporary three-keyword sequences are outside the distance restriction. If the paragraph is the same, the system keeps the three-keyword sequence in the list. Otherwise, it ignores the sequence and goes back to the procedure of combining paired keyword sequences.

Step 3: A temporary three-keyword sequence is required to be disassembled into three 2-faces. All three 2-faces are contained in paired keyword sequences. This temporary three-keyword sequence is considered in the next step. Three 1-simplexes are required to form one 2-simplex. The temporary three-keyword sequence is formed from two paired keyword sequences does not meet the requirement of 2-simplex. It needs to be examined in this step. If the 2-faces are all contained in the paired keyword sequences, this temporary three-keyword sequence is regarded as a three-keyword sequence candidate.

Step 4: Compute the document frequency of the three-keyword sequences candidates of all of the documents. The condition is the same as the previous computing process. All three keywords should be in the same paragraph of one document.

Step 5: If the DF value is greater than the threshold, this three-keyword sequence candidate is inserted into a corresponding table of three-keyword

sequences.

3.4.2.2 Primitive Paired Keyword Concept

The keyword sequences in tables are regarded as concepts. These concepts include primitive concepts and sub-concepts. The system eliminates those sub-concepts and keeps primitive concepts. Every time a three-keyword sequence is generated from a document, two-faces of that three-keyword sequence are added to a sub-concept list.

After the process of computing three-keyword sequences, the system uses this sub-concept list to remove tuples from the corresponding table of paired keyword sequences. The rest of the paired keyword sequences are maximal simplexes since each of these sequences is not a subset of any three-keyword sequences. These paired keyword sequences are also called primitive concepts.

3.4.3 Four-Keyword Sequence

To form a four-keyword sequence, the mechanism is the same as computing three-keyword sequences with some small changes. This section first illustrates the process of computing four-keyword sequences. Then it describes how to use these four-keyword sequences to define primitive three-keyword sequences.

3.4.3.1 Four-Keyword Sequence Computing

Two three-keyword sequences can assemble to a four-keyword set. A four-keyword set needs to be checked to form a four-keyword sequence. Here is the process

to examine a keyword set. Figure 7 shows the process of computing four-keyword sequences.

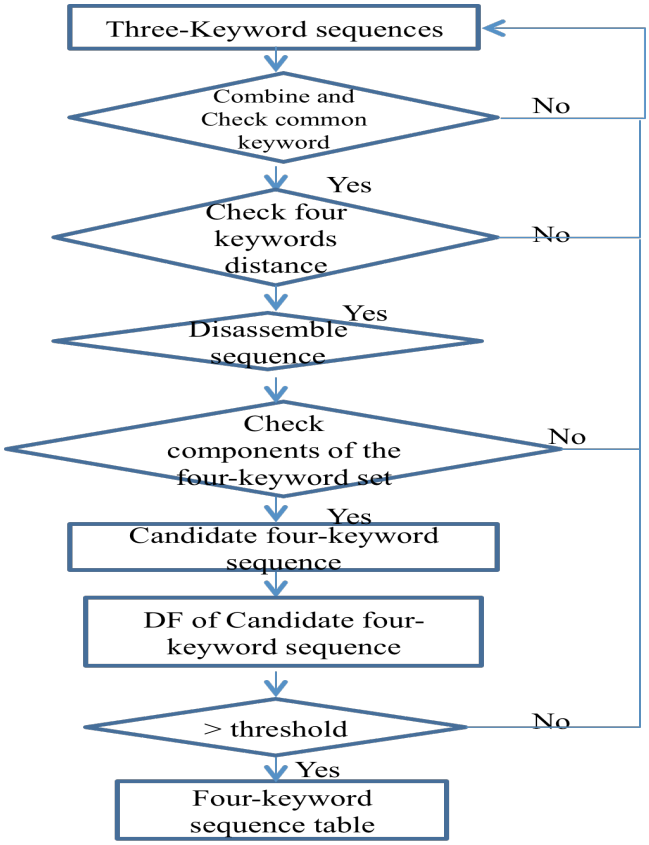

Figure 8. Flowchart of four-keyword sequence computing

The following are steps of the process:

Step 1: Combine two three-keyword sequences. In each table of three-keyword sequences, the system assembles every two sequences and generates a keyword set by their position ordered.

Step 2: Check components of the keyword set. If the number of its components is four, this four-keyword set goes to the next step. Otherwise, the step moves back to assemble another pair of three-keyword sequences.

Step 3: Check the distances between the keywords of the four-keyword set. If they are all in the same paragraph; the four-keyword set goes to the next step. Otherwise, the step moves back to Step1.

Step 4: Disassemble the four-keyword set into four three-keyword components.

Step 5: Check the two components that are not used to assemble in Step 1. If the components exist in the table of three-keyword sequences, the four-keyword set moves to the next step. Otherwise, this four-keyword set is disqualified.

Step 6: Compute the DF of the four-keyword set.

Step 7: Compare the DF value with the user input threshold. If the DF value is greater than the threshold, this sequence is inserted into the table of four-keyword sequences. Otherwise, move back to Step 1.

3.4.3.2 Primitive Three-Keyword Concept

After the process of computing four-keyword sequences, the next procedure is removing sub-concepts from tables of three-keyword sequences. In the disassembling step, every four-keyword sequence has disassembled into four three-keyword sequences. These three-keyword sequences should be removed to leave primitive concepts.

3.4.4 Five-Keyword Sequence

To generate a five-keyword sequence, the algorithm is almost the same as computing four-keyword sequences. The reason to keep extending keywords is that the system is trying to find all primitive concepts from a collection of documents. These primitive concepts are close enough to represent an idea. Although sub-concepts also provide some concepts, they are not close enough to represent the subject of the document.

Most primitive concepts of five-keyword sequences have already narrowed down the concepts to specific fields. On the other hand, if the system can use the algorithm to find five-keyword sequences, this algorithm can also be applied to compute the n-keyword sequence.

3.4.4.1 Five-Keyword Sequence Computing

When combining an n-keyword sequence, the system always selects a pair of (n-1)-keyword sequences to be the elements. The reason is that (n-1)-keyword sequences are currently maximal simplexes. Simplexes of lower dimension can also form five-keyword sequences, which are subsets of maximal simplexes.

The following briefly describes steps of the computing process:

Step 1: Combine two four-keyword sequences. From each table of four-keyword sequences, the system assembles every two sequences by their position ordered.

Step 2: Check components of the keyword set. If the number of its components

is five, then it goes to the next step. Otherwise, the step moves back to assemble another pair of four-keyword sequences.

Step 3: Check the paragraph numbers of five keyword elements of the five-keyword set. If the numbers are the same, then the set goes to the next step. Otherwise, move back to Step1.

Step 4: Disassemble the five-keyword set into four four-keyword components.

Step 5: Check the three components that are not used to assemble in Step 1. If they both exist in the table of three-keyword sequences; the system goes to the next step. Otherwise, this four-keyword set is disqualified.

Step 6: Compute the DF of the five-keyword set.

Step 7: Compare the DF value with the user input threshold. If the value of document frequency is greater than the threshold, the sequence is inserted into the corresponding table.

3.4.4.2 Primitive Concept of Four-Keyword Sequence

After computing the procedure, the system uses the same algorithm in section 3.4.3.2 to find primitive four-keyword concepts. First, the system obtains a list of four-keyword components of five-keyword sequences. Second, it removes all of them from the corresponding four-keyword sequence tables.

3.4.5 N-Keyword Sequence

The longest keyword sequence that this project can compute is a seven-keyword sequence. The algorithms of computing sequences and concepts are similar to section 3.4.3 and section 3.4.4. In other word, this algorithm can apply to N-keyword sequences. The system will not begin the process of computing n-keyword sequences if there is no (n-1)-keyword sequence.

3.5 Knowledge Base

After term extraction, keyword computing, keyword sequence computing, and the primitive concept process, this system can build a knowledge base according to these primitive concepts. Keyword sequences of the knowledge base are a high frequency of co-occurrences of keywords. These keyword sequences can represent latent semantics.

The knowledge database has seven tables. The table headings are Keyword, Paired Keyword Sequence, Three-Keyword Sequence, Four-Keyword Sequence, Five-Keyword Sequence, Six-Keyword Sequence, and Document Name.

The following shows attributes in each table:

1. Table Name: Keyword

   Attributes: doc_id, keyword

2. Table Name: Paired Keyword Sequence

   Attribute: doc_id, keyword1, keyword2

3. Table Name: Three-Keyword Sequence

   Attribute: doc_id, keyword1, keyword2, keyword3

4. Table Name: Four-Keyword Sequence

Attribute: doc_id, keyword1, keyword2, keyword3, keyword4

5. Table Name: Five-Keyword Sequence

   Attribute: doc_id, keyword1, keyword2, keyword3, keyword4, keyword5

6. Table Name: Six-Keyword Sequence

   Attribute: doc_id, keyword1, keyword2, keyword3, keyword4, keyword5
   , keyword6

7. Table Name: Seven-Keyword Sequence

   Attribute: doc_id, keyword1, keyword2, keyword3, keyword4, keyword5
   , keyword6,keyword7

The keyword attributes in these tables are position-ordered. This database can be applyied to a search engine in the future. When users input a keyword set, it compares the index tuples to get the results. A different order of keywords makes different latent semantics. The most important thing is that a search engine uses this mechanism to return the most closely related results that match user' requirements.

3.6 Document Analysis

After the system has built the knowledge base, it can be used to analyze the new documents. Each new document does not need to go through the entire computing process. The system uses the knowledge base and the table of stop words to analyze these new documents. These are some of the advantages of this process. The first one is that the new documents do not need to compute TFIDF. The computation is very time

consuming for keywords. The second is that it also saves the time to compute DF for keyword sequences. The normal way to compute document frequency is to go through all documents to complete the computing. It would be inefficient when the number of documents is large.

The following is the process of analyzing a document:

Step 1: Use the term extraction method to convert document into terms.

Step 2: Remove stop words from these terms to obtain keywords of each document. The system compares terms with a list of stop words from the stop word table. If the term matches the list, the system removes it. In this process, the system uses the removing process instead of keyword computing. The reason is that it does not need to compute TFIDF for keywords. Those stop words are low weights of the terms of sampling documents. Removing these stop words from terms can extract keywords.

Step 3: Use a paired keyword sequence candidate to compare with paired keyword sequences. First, the system orders the keywords according to their positions and groups them by paragraph numbers. Second, according to these groups, keywords are combined to each other by their position. Third, the system compares these paired keyword sequence candidates to the existing paired keyword sequences, which are generated from sampling documents. If the sequence matches, then insert it into the corresponding table.

Step 4: Use a three-keyword sequence candidate to compare with the existing three-keyword sequences. The way to form a three-keyword sequence candidate is to combine two paired keyword sequences and then check the components.

Step 5: Use four-keyword sequence candidates to compare with existing four-keyword sequences. To form a four-keyword sequence candidate first combine two three-keyword sequences to a sequence candidate. Then check the components of the keyword sequence.

Step 6: Use five-keyword sequence candidates to compare with existing five-keyword sequences. The method to generate the sequence candidates is the same as section 3.4.2.1.

The different between the process of analyzing documents and analyzing samples is that the system does not compute TFIDF for keywords and DF for keyword sequences. Definitions of keywords and keyword sequences are based on the knowledge database. This approach reduces the computing time since the system does not need to go through all the documents every time.

4.0 Experimental Result

This chapter illustrates the experimental result based on the system design mentioned on chapter 3. Since the entire computing process is time consuming, we use 10,000 documents to build a knowledge database at the beginning. In this stage, the test

result was used to examine the keywords and keyword sequences. Normally, function words and stop words should have low TFIDF values and the keywords that can represent concepts of a document should appear in the middle of the keyword list. If keyword sequences appear in many documents, it means these keyword sequences contain some latent semantics. Although the number of data sources might be small, the system still can capture some concepts of the document set.

After the first experiment, we made some changes. In the second experiment, we chose 20000 documents for sampling. This allowed us to build a larger knowledge database and our experimental equipment can still handle the computing load. The following are the results of the system setting and computing procedures:

When the system is launched, it requires the user to input some conditions. The first one is the size of samples. From a collection of documents, users can choose how many documents they want to sample. The system chooses a simple random sampling method to obtain sampled documents. It randomly selects a document number and puts it into a list. Every time the system selects a new number, it checks whether the number is different from the previous numbers. If the system chose the same number, it needs to select again until all 20000 document numbers are different. Figure 8 shows the screenshot of user input.

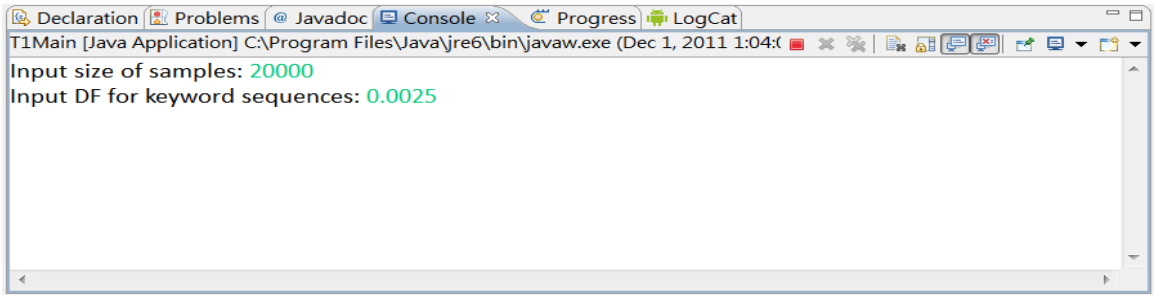

Figure 9. Screenshot of user input

The total number of documents in this experiment is 20000. We chose results from one of the documents as an example in this section. From this example, we can observe the result of keywords and keyword sequences from a text file. Figure 9 is the partial content of an original file. After the process of keyword computing, figure 10 shows the list of keywords of the file. Figure 11 to figure 13 are the result of computing keyword sequences in the file.

IEEE International Conference on Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust

Multi-layered Social Network Creation Based on Bibliographic Data

Przemys?aw Kazienko1,Piotr Brodka1,4, Katarzyna Musial2,3, Jaros?aw Gaworecki4
1 Institute of Informatics, Wroc?aw University of Technology, Wroc?aw, Poland
2 School of Design, Engineering & Computing, Bournemouth University, Poole, United Kingdom
3 Telnet Sp. z o.o., Wroc?aw, 4Research & Engineering Center Sp. z o.o., Wroc?aw, Poland
{kazienko, piotr.brodka, katarzyna.musial}@pwr.wroc.pl, jaroslaw.gaworecki@rec-global.com

Abstract
A method for extraction of the multi-layered social network based on the data about human collaborative achievements, in particular scientific papers, is presented in the paper. The objects linking people form a hierarchy, which is flattened in the pre-processing stage. Only one level of the hierarchy remains together with new activities moved from its other levels. Separate layers of the multi-layered social network are created based on these pre-processed activities.

1. Introduction
There are plenty of possibilities for people to interact and collaborate each other using different IT systems. The data about human interactions and common achievements provides the opportunity to extract social networks existing between people. However, due to their scale, complexity and dynamics, these networks are difficult to be analysed by means of traditional social network analysis methods. Overall, the relations between users can be extracted based on both direct communication and indirect meetings at common activities. The former is e.g. a teleconference via VoIP, whereas to the latter we can count co-authorship of scientific papers. In both situations there exists an object (e.g. a teleconference, an article) that serves as an medium in linking people. Having these objects together with user activities towards them, different types(layers) of relations can be extracted. Additionally, there may exists some relations between objects themselves, e.g. a book co-edited by some scientists contains some chapters worked out by other authors or is cited by another book. The concept of social network has been described by different researchers [11, 12]. The general definition is that a social network is a finite set of individuals, who are the nodes of the network, and activities or relations between them, which represent edges of the network. A social network (SN) commonly represents the mutual communication and activity between users as well as their direction, intensity, and even profile.

Figure 10. Content of the text file

Figure 11. Keywords of partial content of the text file

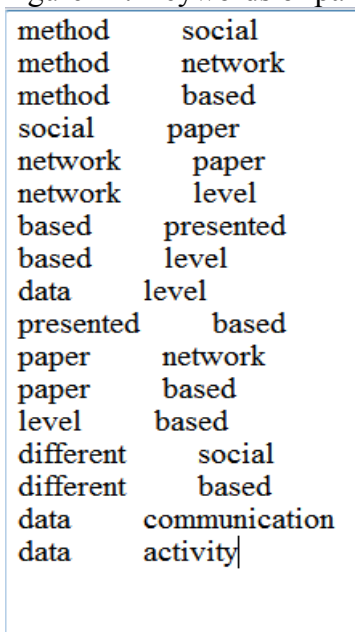

Figure 12. Paired keyword sequences of partial content of the text file

conference social analysis
conference social structure
conference ieee network
social ieee network
social conference network
social conference analysis
social network ieee
social network methods
social network communication
social network nodes
social network work
social network structure
social based data
social based networks
social based research
social data network
social data based
network based data
network based nodes
network data based
based data social
based data network
network social ieee
network social based
network social nodes
network social approach
data networks social
data networks network
data networks research
social networks based

Figure 13. Three-keyword sequences of partial content of the text file

ieee international conference social
ieee international social conference
ieee conference social international
ieee conference social computing
ieee social international conference
ieee social computing conference

Figure 14. Four-keyword sequences of partial content of the text file

In this partial content, the maximal simplexes are 3-simplexes shown as above. These figures show the simplexes of the document. They also show that the longer sequence can present clearer semantics. Therefore, the algorithm of the system captures the concepts of documents. In this case, as long as the document contains maximal simplexes such as n-simplex, the system is able to find these simplexes by using the algorithm.

5.0 CONCLUSION

To provide complete information to users, search engines must contain massive amount of data corresponding to user queries. When a user makes a query, the search engines often returns too many results. Users cannot get the desired information immediately because the results are often unorganized. Although these results are returned fast, users still have to check the results one by one to find out which ones match what they are searching for.

The purpose of the project was to analyze Web pages and documents to obtain knowledge components in them. These knowledge components represent the latent semantics of the documents. If the concepts can be captured, documents can be clustered into meaningful classes. This approach helps to organize Web pages and documents in

search engines. It would also improve the quality of the returned results.

This project proposes to find the maximal simplexes of documents. A 0-simplex is regarded as a keyword and is not enough to identify a latent semantic in a document. The reason is that a single keyword can have too many meanings. The way to identify a document from others is to capture primitive concepts in the document. A maximal simplex can represent a primitive concept. Each maximal simplex consists of many simplexes of low-dimension. In this project, simplexes with low-dimension are used to form maximal simplexes. These simplexes with low-dimension are not used to build a knowledge base.

The algorithms in this project were successful at discovering maximal simplexes in a given set of documents. The maximal simplexes can be used to cluster a document set. These clusters are classified by concepts. These maximal simplexes can also be used to index a knowledge base in the future.

REFERENCE

[1] Knowledge Based Search Engine: Granular Computing on Web, Tsau Young (T. Y.) Lin, Jean-David Hsu, 2008.

[2] A Simplicial complex, A Hyper-graph, Structure in the Latent Semantic Space of Document Clustering, T.Y. Lin, I Chiang, 2005.

[3] Granular Computing: Examples, Intuitions and Modeling, T.Y. Lin. In: the Proceedings of 2005 IEEE International Conference on Granular Computing," July 25-27, 2005.

[4] Concept Analysis and Web Clustering using Combinatorial Topology, T.Y. Lin. In: Workshops Proceedings of the 6th IEEE International Conference on Data Mining, 2006.

[5] Granular Computing: Ancient Practices, Modern Theories and Future Directions, T.Y. Lin, 2008.

[6] Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/ intelligent systems, Soft Computing, Zadeh, L.A, 1998.

[7] Term Weighting Approaches in Automatic Text Retrieval, G. Salton and C. Buckley, 1960.

[8] Granular Computing II: Infrastructure for AI-Engineering, T. Y. Lin. In: the Proceedings of 2006 IEEE International Conference on Granular Computing, 2006.

[9] Extraction of knowledge from databases, Information Processing and Management, P. Willett,1988.

[10] The Key Roles of Information Granulation and Fuzzy logic in Human Reasoning, Zadeh,L. A, In: 1996 IEEE International Conference on Fuzzy Systems, New Orleans, Louisiana, 1996