TOPIC GRAPHS: A CLUSTERING TECHNIQUE TO CATEGORIZE SUBJECT BASED SEARCHES

By

PRATHAMESH R. DIVEKAR

(Under the Direction of Hamid R. Arabnia)

ABSTRACT

Topic based classification and searches have always been a hefty challenge along the corridors of data mining. Reading a large amount of articles and indentifying them of to be the same genre or precisely one subject matter is nearly impossible. With the ever popular need for refinement and quick results we have cropped up a technique to apply graph clustering and probabilistic theory along with known data mining concepts to develop a relationship between words that present high instances of existing together across a majority of documents. These words or topics as we call them form a “Topic Graph”. A Graph is thus a set of words with a high frequent, high probabilistic relationship amongst them. In more technical theory, it is a highly connected graph with words as nodes and relationships between these words as edges. We can apply these concepts of Topic Graphs to refine and categorize search result along with creating new Graphs if the need arises. One of the possible resulting applications should be able to provide precise and specific search answers satisfying user’s requests.

INDEX WORDS: Topic Maps, Graph Theory, Word Relations

TOPIC GRAPHS: A CLUSTERING TECHNIQUE TO CATEGORISE SUBJECT BASED SEARCHES

By

PRATHAMESH R. DIVEKAR

B.E. Information Technology, Pune University, India, 2010

A Thesis Submitted to the Graduate Faculty of

The University of Georgia in Partial Fulfillment

Of the Requirements for the Degree

MASTERS OF SCIENCE

ATHENS, GEORGIA

2014

TOPIC GRAPHS: A CLUSTERING TECHNIQUE TO CATEGORISE SUBJECT BASED SEARCHES

By

PRATHAMESH R. DIVEKAR

Major Professor:	Hamid R. Arabnia
Committee:	Thiab R. Taha
	Ismailcem Budak Arpinar

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2014

DEDICATION

"No one but my parents, are responsible for my success, and I alone am responsible for my failures"

This work has been dedicated to my ever giving, loving and caring parents. I would also like to dedicate my work to my late first cousin "Smita Deo" – who is my inspiration not only for entering into the field of Software and Software development but also for being the true spirit of life in my eyes.

ACKNOWLEDGEMENTS

I would first and foremost, would like to acknowledge the immense support of Dr. Hamid R. Arabnia over the past two years. I wasn't accustomed to the concept of research and research oriented thinking but his tutelage proved to be more than just a guidance but firm hand on my shoulder for motivation. I have cherished the past three years I have spent at UGA's Computer Science department, and the knowledge gained has been priceless to say the least. I would also like to thank my committee members Dr. Taha and Dr. Arpinar for their forever extending support.

I would not forget mention my closest friends with whom I have spent my past three years with. They have been a family away from my family and helped me make a new home away from home. A salute to you people.

Sometimes, it not someone but something, that someone says that proves to be a wakeup call in life. A thank you to that someone for that inadvertent taunt during a homely game of 'Taboo', which indeed woke me up from a deep slumber and motivated me to work harder to achieve my goals.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The invention of World Wide Web (www) has ushered in the era of search engines and information retrieval. Although, the ascension of internet along with its many diversities and interests provided a near unlimited area of storage space for information, it's just too huge to search and thus makes its more and more difficult to find information. Popular web search engines line Google, Yahoo, AltaVista, Infoseek and MSN do exist to help people find information on the web. Most of these systems return a ranked list of web pages in response to a user's search request. Web pages on different topics or different aspects of the same topic are mixed together in the returned list. The user has to sift through a long list to locate pages of interest [18]. Some believe this to be an annoying issue.

Most internet search engines of the present perform a phenomenal task of providing a linear list of sorted or ranked results for a query. For known-item queries, users often find the site they are looking for in the first page of results. However, a list may not suffice for more sophisticated exploratory tasks, such as learning about a new topic or surveying the literature of an unfamiliar field of research, or when information needs are imprecise or evolving [19][20]. Many a times a single word even though a proper noun, may have complete different meanings.

For example: ‘S5’ may refer to ‘Samsung Galaxy S5’ or ‘Audi S5’. The former is the newest phone offered by the Samsung Galaxy series, while the latter refers to a luxury sedan offered by the ever popular German car manufacturer Audi. Queries having ambiguous terms may retrieve documents which are not what users are searching for [22]. No search engine can predict what the user wants to search for at any given moment of time. Although most search engines provide possible related searches or search suggestion, but can never know for sure what the user wants to search.

Another issue is the “why”! The “why” of user search behavior is actually essential to satisfying the user’s information need. After all, users don’t sit down at their computer and say to themselves, “I think I’ll do some searches.” Searching is merely a means to an end – a way to satisfy an underlying goal that the user is trying to achieve. (By “underlying goal,” we mean how the user might answer the question “why are you performing that search?”)[21]. That goal may range from buying the grocery to the newest video game. Or it may range from finding the latest election results to finding what his next door neighbor is up to. Or it even may be to find if some famous celebrity said something controversial to voicing his own opinion about a certain pertaining issue or a plethora of possibilities. In fact, in some cases the same query might be used to convey different goals - For instance, an user searching for ‘Samsung Galaxy S5’ might get results ranging from the technical knowhow, price to possible outlet stores that sell the product. He may only want to know about the technical issues of the phone rather than the best place to purchase it and if the search results produced somehow tend to be more inclined towards ‘possible places to purchase’ type, that itself may annoy him enough to produce a negative impression about it.

Perhaps, no technology of the present or the near future may have the capability to completely solve such issues, but techniques like related searches and suggestion may resolve them to some extent. This thesis presents a possible solution or at least a way to reduce user's annoyance with search engines called as "Topic Graphs and Topic/Sets" - a system to compartmentalize search results based on sets of closely associated words. Here we devise a technique to formulate bands of words together, having highly logical relations in the real world. By real world I mean the human universe as we know it. The common issue of unnecessary, ambiguous and redundant search results can be reduced to some extent.

What we attempt is to provide search engine, users or any possible searching techniques a hierarchy to search for. Instead for directly searching for the user query, a system may use our topic sets for searching a query thus returning results associated directly with these sets and organized categorically based on these sets. This thesis introduces this concept of topic graphs and topic sets, their benefit for searching and a process through which they are forged from any available collection of documents. The final result that is produced is a compilation of groups of words that can be then used as templates for searching as mentioned before. For example: Our previous example of 'Samsung Galaxy S5' could be associated with searching for price, outlet stores, tech specs or comparison with competitors. For that follows topic sets can be available: [Galaxy, S5, Target, At&t], [S5, Verizon, At&t, T-Mobile, ...], [Samsung, S5, Galaxy, PC Magazine, Chips, Amoled, …] or [Galaxy, S5, HTC, One, M8, ….] etc.

1.1 The Concept

The core theory of this document is based on the simple fundamental belief that “no word is ever alone”. Thus we have personified words and phrases to have relationships with other words. Any definition of any topic is a group of words semantically arranged together to make sense. Thus, every single term is any sort of search query can be believed to be associated with a set of words that define them and rather add character to them. We establish relationships between keywords by using a graphical approach. Why Graphs? Because graphs are an efficient data structure to represent hierarchical information. This question gets answered more clearly in later chapters.

Our application is a four stage structure in technical terms but we can introduce it in a three step perception to put forth a foundation for this thesis as follows:

1. **The Corpus:** A plethora of documents exist in every format in the universe. Every known information is represented in a written format so that it can reach every corner of the world to be distributed. The first step is to take a bunch of such documents at random and formulate their base topic based on devising key words from written paragraphs.
2. **The application:** The application will establish associations between available keywords by calculating a probabilistic weight between those words that frequently appear together in a host of documents. The result will be a graph with keywords as vertices and their relationships with other words would be the reason for its edges and their weights. The weights help to establish a relationship score between these words.
3. **Clusterize:** The final step is to group together those words which have the maximum or near maximum relationship score between them. So we eliminate weaker edges and form

several sub graphs. We then establish a hierarchical tree structures for these sub graphs which gives us topic graphs which can be further categorized and organized into topic sets.

The further chapters in the thesis will present in depth the whole process of how an assemblage of random documents result in the formation of topic sets. We will describe the core system architecture - which essentially is four stages, a working on-paper demonstration, an evaluation of our tests that provide an evidence for the theory and possible enhancements to the proposed concept.

CHAPTER 2

BACKGROUND AND MOTIVATION

The need to categorize has always been evident in human nature. Categories present a systematical approach in any organizational approach. Simple examples are evident in our day to day lives starting from our personal bedrooms to kitchens, from our TV guides to restaurant menus, from groceries to books. Categorical and systematic organization of information has always been and will always be the prime need and expectation of any venture.

Three general techniques have been used to organize documents into topical contexts. The first one uses structural information (Meta data) associated with each document. The DynaCat system by Pratt [23] used Meta data from the UMLS medical thesaurus to organize search results. In the SuperBook project [24], paragraphs of texts were organized into an author-created hierarchical table of contents. Others have used the link structure of Web pages to automatically generate structured views of Web sites. Maarek et al.'s WebCutter system [25] displayed a site map tailored to the user's search query. Manually-created systems are quite useful but require a lot of initial effort to create and are difficult to maintain. Automatically derived structures often result in heterogeneous criteria for category membership and can be difficult to understand [18]. A second way to organize documents is by clustering. Documents

are organized into groups based their overall similarity to one another. Zamir et al. [26, 27] grouped Web search results using suffix tree clustering. Hearst et al. [28, 29] used the scatter/gather technique to organize and browse documents. Clusters are usually labeled by common phrases extracted from member documents [18]. A third way to organize documents is by classification. In this approach, statistical techniques are used to learn a model based on a labeled set of training documents (documents with category labels). The model is then applied to new documents (documents without category labels) to determine their categories [18].

Our concept uses a bit of each three methods described above. The first method of organizing in a hierarchical format based on the structural information of the document is the final result of our whole application. The final output is a file containing terms organized as trees and we formulate our topic sets based on these trees. The second method is applied to create the topic trees. We clusterize one major graph obtained by establishing relationships between all the terms and then organize these clusters into a hierarchical tree structure. The whole idea and its introduction are based on the third method. We formulate these topic trees and graphs along with topic sets and then propose an application to categorize searches based on these topic sets we have obtained.

Search Engines like Google, Bing, and Yahoo now-a-days deliver a customized search result. This leads to an effect that has been called a filter bubble. Thus, the user has information retrieval process based on his past experiences and searches rather than the present ongoing. News articles produce new results on a daily basis which can never be found in any user's search history because they are new. Thus it becomes imperative to categorize searches based on inter

topic relationships as well. According to Eli Pariser, who coined the term, users get less exposure to conflicting viewpoints and are isolated intellectually in their own informational bubble. Pariser related an example in which one user searched Google for “BP” and got investment news about British Petroleum while another searcher got information about the Deepwater Horizon oil spill and that the two search results pages were “strikingly different”[1][2][9]. The bubble effect may have negative implications for civic discourse, according to Pariser. Since this problem has been identified, competing search engines have emerged that seek to avoid this problem by not tracking or “bubbling users [10].

A filter bubble is a result state in which a website algorithm selectively guesses what information a user would like to see based on information about the user (such as location, past click behavior and search history) and, as a result, users become separated from information that disagrees with their viewpoints, effectively isolating them in their own cultural or ideological bubbles. Prime examples are Google’s personalized search results and Face book’s personalized news stream [1][2][9].

We argue although these personalized searches present the user with results that adhere to their interests and liking rather than presenting the data which is more complementary to the updated happenings on the planet. We thus take this into account and present a more public knowledge based categorizations rather to help the user negate this filter bubble. Instead we try and filter out those redundant and unnecessary results that prove more of a nuisance.

2.1 Why Graphs?

A month ago a search for ‘Malaysia Airlines’ would have returned the web address of the chief website of the airlines company or a schedule depicting the coming and goings of certain flights operated by the same airline organization. A search for most now would be expected to return results concerning foremost the words “MH370, Flight 370, disappearance, mystery etc.” – all concerned with the latest tragedy that occurred over heaven and earth. Such incidents that happen over time change the complexion of the world as we believe it. Such changes not only affect our mind set but automatically set its tone on the cyber world as well. Among all one thing that remains consistent is the ability or the human nature to associate relationships between what we call words no matter how much situations change. Times change we roll on with it and learn to adapt the fact that words will be associated with each other for an era.

Every search provides with some related search suggestions yet there are some redundant results or those that contain the searched query as a mere formality. Every term – or word – has some associations – related words – which have a certain high probability of occurring together in a host of documents. Such word association can be assumed to have a relation between them and can be assigned certain weights - to signify the strength of their bond – which would ultimately mean that they appear together in most documents and searching for one along with its siblings can return more specifically categorized results. Also classifying documents (books, articles, web pages etc) based on these words and association can help users to have more content specific searches.

So where does the graph theory come in? Consider each term as a node or a vertex of the graph. It's a vast ocean of words out there and the only thing we know about them is that they belong to some document. We then establish some relationships between some of these words and create edges amongst them. Thus we have a huge graph where words are connected to each other. Next we contract this graph. We remove some unneeded edges and with them some isolated vertices. Thus we have particularly important words with a certain calculated weights and each of these has some associations with other words – strong or weak. After we are done with the contractions, we need to have only those word associations that have enough to be together forever. Thus we form clusters from the main graph. Each of these clusters is based on the fact that the terms in them a certain high probability of appearing together in different document. The thing to note here is that not all the words a cluster necessarily appear in the same document. The probability that a document would contain all the terms of the cluster in probably pretty low but the certain group of terms in the cluster do appear together with a high probability. In more technical terms we have highly connected graphs – not complete graphs – which signify the close associations between words and gives us more than one topic sets out of a topic graph.

For example: staying with the Malaysia Airlines disaster. Suppose we have over ten articles concerned with the latest incident that occurred over the Indian Ocean waters. Each of these documents filters out words that occur in them with a certain high frequency. We can have the following words – *Malaysia, Flight, MH, and 370, Kuala Lumpur, Indian, Ocean, India, Australia, Asia, Beijing, Boeing, Thailand, gulf, Malay,* accident and some more. Now the actual incident happened when "**Malaysia Airline Flight MH 370**, travelling from **Kuala Lumpur** International airport to **Beijing** International airport went missing less than an hour after its take

off". All the bold words in the above sentence are the key words discussed or mentioned when people speak about the incident. To be more precise, these words appear together with high probability in a group of documents related to this incident. Thus these words could be clustered together since they have such a strong association. Thus we have a topic graph G (V, E) where *V = {Malaysia, Airline, Flight, MH 370, Kuala Lumpur, Beijing}.*

The above example thus implements the theory of graph clustering to string together terms which have a high certainty of appearing together in a corpus of document.

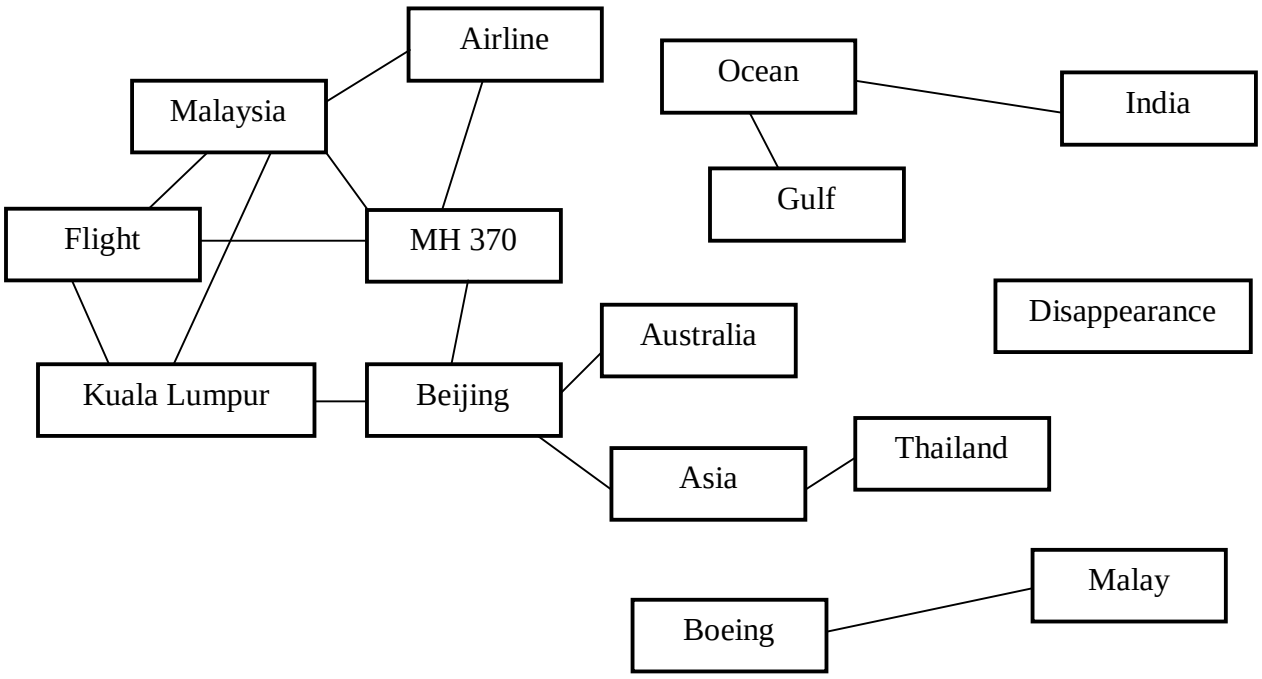

Figure 2.1: *A graph of related terms based on the example mentioned above.*

2.2 Graph Representation

In computer science, a graph is an abstract data type that is meant to implement the graph and hyper graph concepts from mathematics. A graph data structure consists of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices. As in mathematics, an edge (x, y) is said to point or go from x to y. The nodes may be part of the graph structure, or may be external entities represented by integer indices or references. A graph data structure may also associate to each edge some edge value, such as a symbolic label or a numeric attribute (cost, capacity, length, etc.) [11].

We thus exploit this of nodes and edges since our prime objective is to establish a relationship between words. The terms themselves become the nodes and the edges between them become a reason to define the existence of a relationship between them.

Various ways to implement graphs exist in programming terms. The two most basic ways are Adjacency Lists and Adjacency Matrix. Operations with a graph represented by an adjacency matrix are faster. But if a graph is large we can't use such big matrix to represent a graph, so we should use collection of adjacency lists, which is more compact. Using adjacency lists is preferable, when a graph is sparse, i.e. $|E|$ is much less than $|V|^2$, but if $|E|$ is close to $|V|^2$, choose adjacency matrix, because in any case we should use $O(|V|^2)$ memory [12].

Adjacency matrix and adjacency lists can be used for both directed and undirected graphs [12]. We however use a combination of these two basic representations. We use Hash Maps and Linked List. Both these Data structure are used for different phases of the system architecture which has been described in the further chapters.

We use Hash Maps to define the term graph (graphical representation of all key words and their relationships with other key words in the corpus). The key is the term as the node of the graph and the value is a List of terms that and its connections.

Linked list are utilized during the Topic Graph creation phase where we arrange all terms in a hierarchical tree formulate topic sets later. Each node of the tree/graph is linked to its child in the structure thus having a kind of a unidirectional traversal – our graphs are not directed.

	Hash Map
Memory complexity *(optimal – O(|E|)*	***O(|E|)***
Add new term *(optimal – O(1))*	***O(1)***
Remove term *(optimal – O(1))*	***O(1)***
Search for a term *(optimal – O(1))*	***O(1)***
Enumeration of vertices (term) adjacent to 't (term in question)' *(optimal – O(|K|))*	***O(|K|)***

Table 2.1: *Memories and complexities for a HashMap. We consider each term to a vertex in terms of a graph. Thus K is the number of adjacent terms to a term t* **[12]**.

CHAPTER 3

THE SYSTEM ARCHITECTURE

In this chapter we describe the multi stage architecture of our topic-set maker and provide a detail insight into each component of the system. We have discussed in the prior chapters the challenges that we encounter in creating these graphs and categorized topic sets. Having a multi stage system is essential to prevent redundancy along with preserving the precision that does not yield false results.

The journey from a group of documents to creating a graph and a more precise topic graphs to topic sets, is a fourfold. Each stage outputs a distinct set of files with more precise and simplified information than its predecessor stage. Our input files start as text files. Each input file is an article or paragraphed sentence as defined in legible English language. By legible English we do not mean, they are random words just typed for the sake of typing or a computer language program – which though English do not meet the legible criteria – that is they can be successfully parsed by an English language parser. The end result is one single file though called as '.tt' file – tt stands for topic template, which has essentially topic graphs and topic sets associated with each graph. The end result also contains two more files: 1) Set of edges and their weights. 2) A hash map file where the key is a term and value is a list of all its true connection. We will explain further what true connections are.

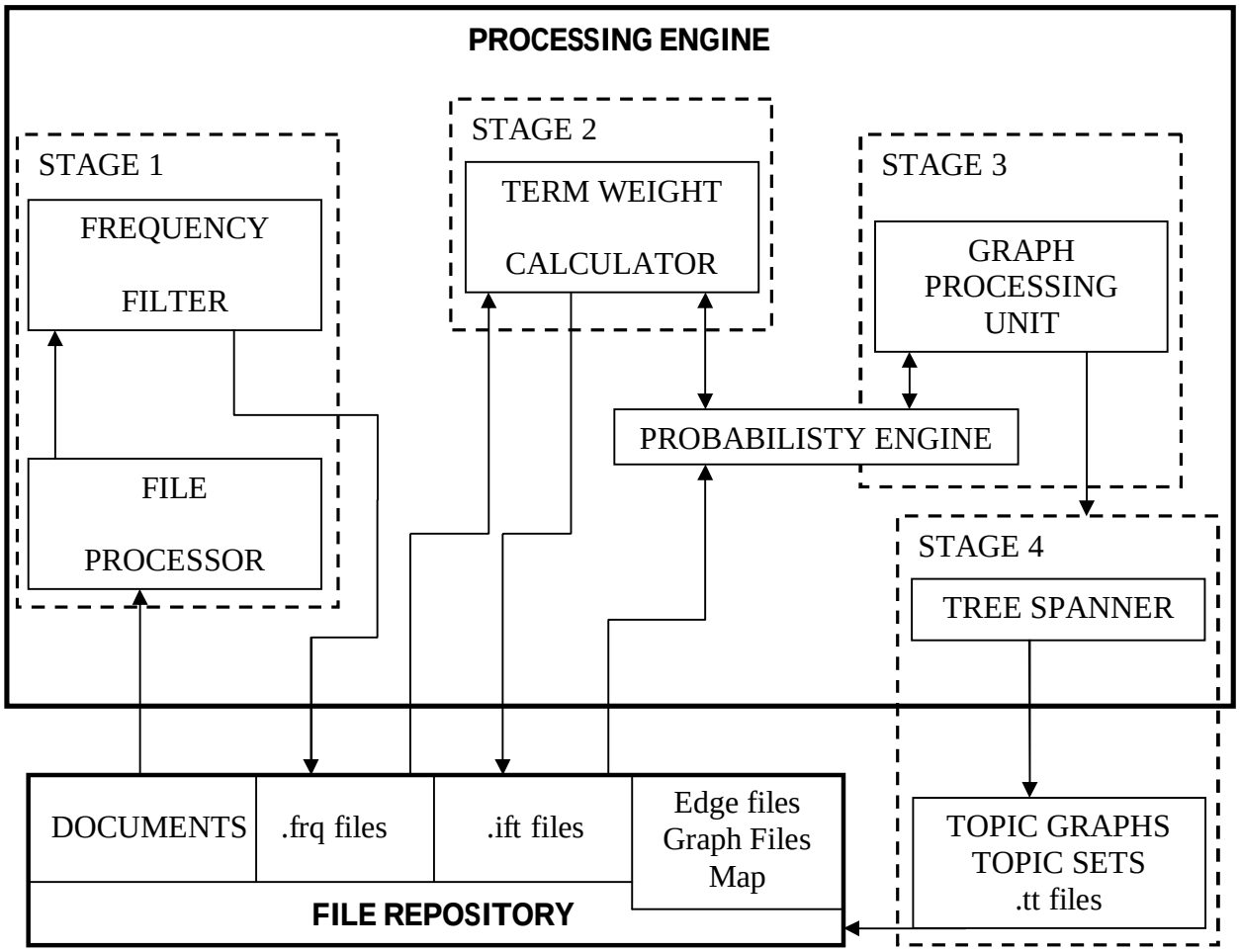

Figure 3.1: TGS System Architecture

Figure 3.1 signifies the system architecture of a simple topic sets creator. The system architecture is divided into two parts – The Processing Engine and the Repository.

The Processing Engine as the name specifies does all the work starting from filtering files and driving the topic sets from the input corpus. The processing engine holds the four stage architecture which gives the final output. Both input and the output are used from and stored in the file repository of the system.

All files – input documents, .frq, .ift, Graph, Edge, Hash Map and .tt – are stored in the repository. The repository or the File Repository could be any kind of a storage space like a database, online files repository – GitHub, SVN, Bit Bucket etc, or just a folder on the local host or some common server. Use of some kind of a personal digital library could be encouraged as it presents multiple advantages – no physical library, round the clock availability, multiple access and uses, information retrieval, finer preservation and conservation, possibly infinite space (for a considerably low cost)[13].

3.1 Stage 1: File Filtering

Stage 1 of the architecture involves the base input corpus. The corpus is a group of documents that could be any of the following:

1. Newspaper and/or magazine Articles
2. Wikipedia blogs
3. Online blogs

Apart from the above (which we used for testing) the corpus could include any legible English language articles popularly talking about some base topic and its constituents but with proper sentence construction. Documents written in modern urban slang or popular short hand abbreviations are discouraged so as not to get unnatural results.

The output of the file filtering stage is the '.frq' files. These assign an identity – some integer id – to every file and contain a list of keywords and their term frequencies.

Term Frequency: Tf(t, d) is the raw frequency of a term in its document, i.e. the number of times the term appears in the document[15].

File Processor: The file processor stems down the available text to words. During the stemming process we cut down plurals, verb forms etc to their base forms and we get rid of high frequency stop words like articles, prepositions [14].

Frequency Filter: We calculate the 'Tf' for every keyword and using a certain threshold we filter out those terms that pass a certain threshold frequency. For example the average frequency of a term in a document is 15; we remove all those key words that have 'Tf' less than 15.

3.2 Stage 2: Term Weight Processing

Stage 2 gives a certain popularity score to every word based on the following quantities. The output of the term weight processing engine are '.ift' files. The .ift files have a table with each term its Tf, Idf, term weight and the Df. **Term Weight Calculator** is the only component of stage 2 that achieves this target.

Inverse Document Frequency: The inverse document frequency is a measure of whether the term is common or rare across all documents. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient [15].

$$Idf\ (t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \qquad (3.1)$$

N: The total number of documents in the corpus.

Documents Frequency: It is the number of documents where the term t appears. In equation (.1) $|\{d \in D : t \in d\}|$ is the Document Frequency or Df.

TfIdf: The term weight of the TfIdf is calculated as the product of Tf and Idf($\mathrm{Tf}(t,d) * \mathrm{Idf}(t,D)$). A high weight in Tf * Idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the Idf's log function is always greater than or equal to 1, the value of Idf (and TfIdf) is greater than or equal to 0. As a term appears in more documents; the ratio inside the logarithm approaches 1, bringing the Idf and TfIdf closer to 0 [15].

3.3 Stage 3: Term Graphs Arrangement

Stage 3 assembles the term graph – important key words based on certain Tf and Term Weight thresholds – and formulates edges between them considering that each term is a node for these edges. Creating edge at this stage is just based on the fact that two words of an edge occur together in multiple documents. Based on the number of occurrences we calculate $p(E)$ that is the probability of the edge which is essentially a ratio - O/U – the ratio of the occurrences over the union of sets that contain the terms that bind the said edge E.

Stage 3 produces three files:

1. Graph: A file containing terms and all its connection.
2. Edge: A file that keeps track of all edges and its probabilistic weights.
3. Hash Map: A file that keeps track of all terms and the documents it appears.

Graph Processing Unit: Not to be confused with a GPU (Graphical Processing Unit), a graph processing unit in this particular system gives the term graph. It assembles edges together between vertices and assigns weights to these edges.

Probability Engine: The probability engine calculates the p(E) for every edge, filters out those below a certain threshold and creates "TRUE EDGES" out of all those available edges. Thus a true edge is and Edge between two terms in a term graph, whose probabilistic weight surpasses a certain threshold set by the creator.

3.4 Stage 4: Topic Graphs, Trees and Sets

The fourth and the ultimate stage of the system give us a file with comprehensive topic graphs and topic sets. The output is '.tt – topic templates' files as mentioned before. The files presents a hierarchical tree structures of topics arranged together establishing a more systematic parent child relationship between terms. We analyze these relationships further to give topic sets.

We define the two subjects of our final output as follows:

Topic Graphs: *Topic graph or topic trees are inter-related term derived from our base corpus, arranged in a hierarchical fashion much similar to a family tree thus establishing a parent child relationship between words that appear in a term graph.*

The thing to remember about topic trees is that, direct siblings and parent – child definitely appear together with high probability in documents together in the corpus. A parent

and its direct child will be present together across certain high number of documents, but parent its grandchildren may or may not. Similarly two siblings of the same parent will appear together in significant number of documents but cousins may not.

Topic Sets: *Topic sets are a set of terms such that, each term in the set has a high probabilistic relationship with more than two terms in the set.*

Thus Topic Set,

$S = \{x | \forall x \in S, x \in TG \; and \; \exists y > 2 : x \; and \; y \; have \; a \; true \; edge \; between \; them\}$

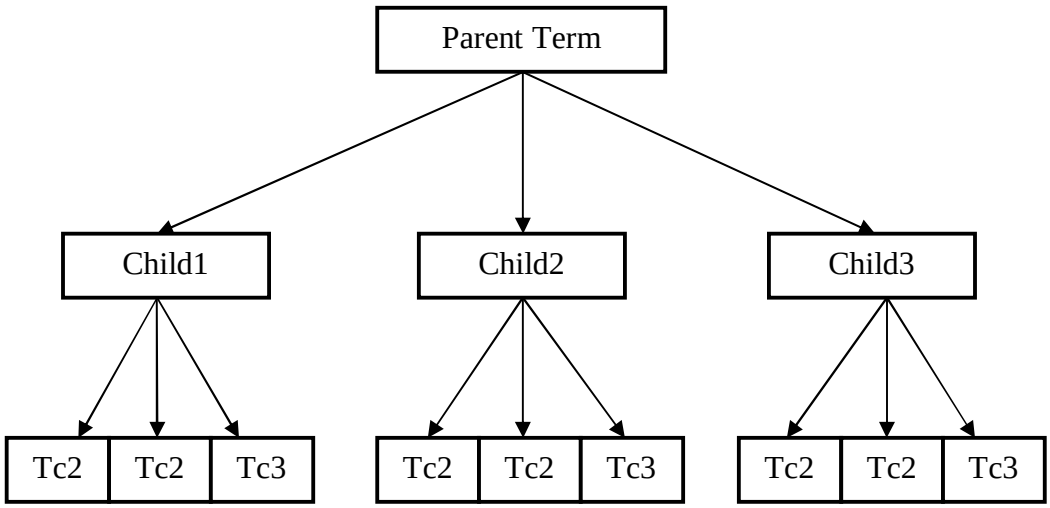

Figure 3.2: Topic Tree

Figure 4.2 is a one such topic graph. It is not essentially a tree because even we do not depict it; an edge definitely exists between two siblings of a parent. A tree structure just helps in a hierarchical arrangement. A topic set thus will have parents, its children and grandchildren. Cousins won't be a part of the same topic template thus giving categorical divisions when it comes to every term.

Tree Spanner: The tree spanner clusters the term graph into topic graphs and further divides the topic graph into topic sets.

TGS Repository: The TGS repository is a child repository of the File repository that finally stores all the topic graphs and topic sets that we would get from a base corpus.

CHAPTER 4

TOPIC GRAPHS

In this chapter we elaborate what Topics graphs are; how topic sets are formulated from a topic graphs aided by a basic logical example – in this sense logical adheres more to obvious than sensible. We start with no more than two documents. Explain how and why we do what we do at every process and at the end present a pseudo code algorithm to implement our theory. To make the application more apparent in layman terms, we will try to establish the similarity between that family tree structure and our topic graph formulations as we mentioned in our introduction chapter.

4.1 The Corpus

For the sake our non computerized human implementation of the system that we propose, we have chosen four paragraphs rather than entire documents to make the working of the system apparent and simple to a layman. These for paragraphs – called as documents henceforth – are summarized descriptions of the last four *Harry Potter* books. They are follows:

Doc 1: ***Harry Potter*** *and the* ***Goblet*** *of* ***Fire****.*

Book four of the ***Harry Potter*** *tests* ***Harry*** *in the most unusual way not only his abilities to cope with life threatening challenges but also his friendship with* ***Ron Weasely*** *and* ***Hermoine Granger****.* ***Hogwarts*** *and the* ***ministry*** *of* ***magic*** *after a hiatus of almost a century organize the* ***tri wizard tournament*** *between three best known magical schools –* ***Drumstrangs*** *and* ***Beauxbatons****.* ***Albus Dumbledore*** *who has sensed the signs of the eminent return of Lord* ***Voldemort****, has employed ex-auror* ***Alastor Madeye Moody*** *as the new* ***defense*** *against the* ***dark arts teacher*** *with a view to protect* ***harry****. New characters and plots are introduced in the fourth with most awaited of the main villain of the series in this book. Things will change for* ***Harry*** *and his friends.*

Doc 2: ***Harry Potter*** *and the* ***Order*** *of the* ***Phoenix****.*

Lord ***Voldemort*** *has returned and though the* ***ministry*** *of* ***magic*** *is arrogant enough to ignore it,* ***Dumbledore*** *has summoned* ***Sirius Black*** *and the* ***Order*** *of the* ***Phoenix*** *to organize a resistance against the* ***dark arts****.* ***Harry, Ron*** *and* ***Hermoine*** *return to* ***Hogwarts*** *where the* ***ministry's*** *motivation to curb* ***Dumbledore's*** *so called lies has taken an unexpected stand.* ***Dolores Umbridge*** *has been appointed the new* ***Defense*** *against the* ***Dark arts teacher*** *and the High* ***Inquisitor*** *to inculcate some discipline among the failing standards of the school. If that's not enough* ***Harry*** *is constantly facing nightmares which actually is a direct connection to* ***Voldemort's*** *mind. Tough times await* ***Harry*** *as the* ***Hogwarts*** *he knows will never be the same.*

Doc 3: ***Harry Potter*** *and the* ***Half Blood Prince***

The death of ***Sirius Black*** *and the revelation of Lord* ***Voldemort's*** *return have sparked the same panic and unrest in the magical world as it was fifteen years ago.* ***Dumbledore*** *and* ***Harry*** *embark on a mission to discover the life and lies of* ***Tom Riddle*** *a.k.a Lord* ***Voldemort****. Their journey leads them into the world of* ***Horcruxes*** *–* ***Dark*** *objects that store a* ***wizard's/witch's*** *soul making him undefeatable. Life in* ***Hogwarts*** *is back to its usual self though with the imminent danger of the death eaters apart from the fact the* ***Severus Snape*** *as the new* ***Defense*** *against the* ***dark arts teacher****.* ***Harry*** *thus struggles to come with terms* ***Snape's*** *latest victory. New challenges await* ***Harry, Ron*** *and* ***Hermoine****, some not associated with the dangers of the real world as they come of edge. This book will indeed prove to a cliff hanger.*

Doc 4: ***Harry Potter*** *and the* ***Deathly Hallows***

The last battle, the final war. ***Snape's*** *betrayal which led to the death of* ***Albus Dumbledore*** *has sparked fall of* ***ministry*** *and* ***Hogwarts*** *into the hands of the death eaters.* ***Harry, Ron*** *and* ***Hermoine*** *embark on the mission set by* ***Dumbledore*** *to find and destroy Lord* ***Voldemort's Horcruxes*** *which will lead to his defeat. On their journeys they discover the existence of the* ***deathly hallows*** *which are believed to be objects that would make the owner a master of death. A race issues between the good and the bad over the possession of these* ***deathly hallows*** *which lead to the biggest war* ***Hogwarts*** *has ever seen.*

4.2 File Filtering

We filter these four documents in stage 1. We remove all the high frequency stop words, and create a table with each document and its keywords and their term frequencies.

For the sake of easing our non computerized calculations we have only considered proper nouns that appear in the documents mentioned above. Our system does not classify between nouns, adjectives, pronouns etc. since we do not want to enter into the realms of Natural Language Processing. So we remove all stop words and unnecessary words. We keep the obvious key words in the documents. Then we assign the term frequencies to each term. We can set a certain frequency threshold o filter out low frequency words from the document. In this case we set it to 2.

$\boldsymbol{Filter\ documents\ (t, d)}$

$if\ (tf(t, d) < 2)$

$remove\ t\ from\ d$

$\boldsymbol{end}$

Algorithm 4.1: Filter Documents

Document 1 - 01	Document 2 - 02	Document 3 - 03	Document 4 – 04
Harry 5	Harry 4	Harry 4	Harry 2
Potter 2	Order 2	Voldemort 2	Deathly 3
	Phoenix 2	Dark 2	Hallows 3
	Voldemort 2	Snape 2	Hogwarts 2
	Ministry 2		Dumbledore 2
	Dumbledore 2		
	Hogwarts 2		
	Dark 2		

Table 4.1: contents of a .frq file: Terms and frequency

Table 5.1 is a typical '.frq' file which contains all the term that qualify a certain set frequency threshold and their frequencies. Thing to note is that each column of the table is a separate file each accompanied by the document id.

4.3 Term Weight Processing

We get four '.frq' files from the first stage with terms and their frequencies. We apply and calculate the Document Frequency (Df), Inverse Documents Frequency (Idf), Tf*Idf of each term from '.frq' files. Thus we have a popularity quotient for which term which signifies how important the term is in its document.

	Tf	**Df**	**Idf**	**Tf * Idf**	**Doc Id**
Harry	5	4	0	0	1
Potter	2	1	0.602	1.204	1

Table 4.2.1: '.ift' file for Doc 1

	Tf	**Df**	**Idf**	**Tf * Idf**	**Doc Id**
Harry	4	4	0	0	2
Order	2	1	0.602	1.204	2
Phoenix	2	1	0.602	1.204	2
Voldemort	2	2	0.301	0.602	2
Ministry	2	1	0.602	1.204	2
Dumbledore	2	2	0.301	0.602	2
Hogwarts	2	2	0.301	0.602	2
Dark	2	2	0.301	0.602	2

Table 4.2.2: '.ift' file for Doc 2

	Tf	**Df**	**Idf**	**Tf * Idf**	**Doc Id**
Harry	4	4	0	0	3
Voldemort	2	2	0.301	0.602	3
Dark	2	2	0.301	0.602	3
Snape	2	1	0.602	1.204	3

Table 4.2.3: '.ift' file for Doc 3

	Tf	Df	Idf	Tf * Idf	Doc Id
Harry	4	4	0	0	4
Deathly	3	1	0.602	1.806	4
Hallows	3	1	0.602	1.806	4
Hogwarts	2	2	0.301	0.602	4
Dumbledore	2	2	0.301	0.602	4

Table 4.2.4: '.ift' file for Doc 4

4.4 Term Graphs Arrangement

We process all of the '.ift' files from stage 2 and prepare term graph. Term graphs are formulated using three files – The Edge file, the hash map and the Graph. The edge file contains all the edges with their associated probabilistic weights. The Hash map has all the terms and a list of all documents it is contained in. And the Graph contains all the vertices and its adjacency lists.

The first file created is the Hash Map which contains all the term and each term is associated with a Document Set – Ds. Ds of term t contain the Ids of all the documents the term t belongs to. We formulate the edges as follows:

E (u, v) is an Edge if u and v are terms from processed '.ift' files and the size of the intersection of the Document Sets of u and v is greater than a certain predefined threshold which is mandatorily more than or equal to 2. We calculate the probability weight of each edge as

$$Pwt(E(u,v)) = |Df(u) \cap Df(v)| \,/\, |Df(u) \cup Df(v)| \quad (4.1)$$

The contents of the three files of this stage for our corpus are as follows:

Hash Map:

Harry: [1, 2, 3, 4]

Voldemort: [2, 3]

Dumbledore: [2, 4]

Hogwarts: [2, 4]

Dark: [2, 3]

Edge:

$Harry, Voldemort: wt = 2, Pwt = 0.5$

$Harry, Dumbledore: wt = 2, Pwt = 0.5$

$Harry, Hogwarts: wt = 2, Pwt = 0.5$

$Harry, Dark: wt = 2, pwt = 0.5$

$Voldemort, Dark: wt = 2, pwt = 1$

$Hogwarts, Dumbledor: wt = 2, Pwt = 1$

Graph:

$Harry \rightarrow Voldemort, Dumbledore, Hogwarts, Dark$

$Voldemort \rightarrow Dark$

$Dumbledore \rightarrow Hogwarts$

4.5 Topic Graphs, Trees and Sets

The last stage produces the Topic Graph/Tree and the Topic sets. We start by traversing every vertex in the graph and looking at every edge in the term graph. For every vertex in the graph we see if those present in its adjacency list belong to the lists of each other. Those thus that have connections between them become the children of the first vertex we started from. Next we check the 'Pwt' of each edge that we have chosen. If the Pwt (Child1, Child2) is more than that of Pwt (Parent, Child2) then Child2 becomes the child of Child 2 in the tree.

$\boldsymbol{CreateTree\ (G,\ t,\ t_i,\ t_j)}$

For $every\ t \in term\ graph$

  For $every\ t_i \in Adjacency\ list\ (t)$

    if $t_j \in t_i$

      $Add\ t_i, t_j\ to\ child\ list\ of\ t$

    if $(Pwt(t_i, t_j) > Pwt(t, t_i)$

      $Add\ t_j\ to\ child\ list\ of\ t_i$

  end

end

Algorithm 4.2: Create Tree algorithm

For every tree that we get from the term graph every LHS and RHS of the parent is a topic set. To limit the number of words in a topic set, our tree are limited to no more than four generations. The '.tt' files of our corpus yield the following results:

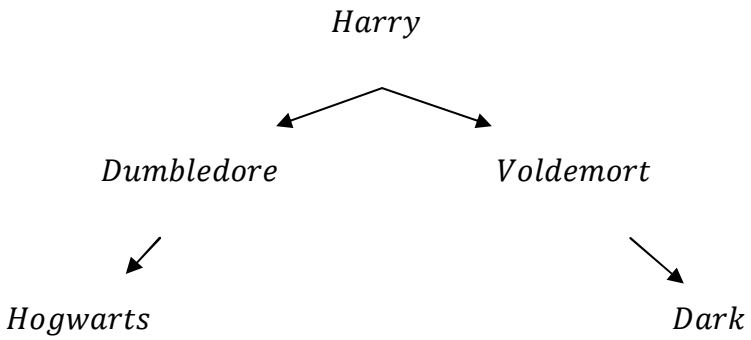

Figure 4.1: Topic tree in the corpus from 5.1

We get two topic sets from the above tree. Our main parent 'Harry' has two children and two grand children. Each pair of child – grandchild along with the parent is a topic set as follows:

1. $[Harry, Dumbledore, Hogwarts]$
2. $[Harry, Voldemort, Dark]$

Now we can arrange any document related to harry potter based on these two topic sets. Each set will return a specific set of documents related to the subjects in the sets. For example Document 3 will not be a search result for set 1 and Document 4 will not be returned when the search is concentrated towards set 2. On a more fantasy note, students at Hogwarts school of Witchcraft and Wizardry can have a more categorized search in the Hogwarts Library based whether their interest lies in Albus Dumbledore and Hogwarts or Lord Voldemort and the Dark arts.

CHAPTER 5

EXPERIMENTAL EVALUATIONS

The evidence to prove the success of our enterprise, we designed and implemented a thorough application adhering to our System Architecture. The application was programmed using the objected oriented concepts where each of the important units of the system like Graph, Edges, vertices, terms and words were systematically organized as classes. Java was the OOTP language used for the purpose of demonstration with Eclipse Kepler being the programming tool.

We took assistance of certain predefined 'JARS' for the purpose of information and data handling. 'Lucene' [17] being the key framework for deriving quantities like Tf, Idf for terms. We also used existing classes of java to arrange and organize information that suited best to needs. The use of Hash Map, Array List, Hash Sets, stacks and queues is the best example for this.

Our experimental corpus consisted of primarily news articles in '.txt' format. The corpus was a host of documents ranging from 100 words to 2000 words. Our results constitute the findings of topics sets that exists for 10, 20, 50 and 100 articles. For part one of the testing process we manually categorized the documents based on their subject matter. We tested our

finding across each category. Part two of the process included processing all the documents randomly together.

To observe the changes in the number of topic sets produced we set different threshold values for the following quantities: Tf, Tf * Idf, Cardinality of intersection of adjacency lists and the Pwt of edges. Some thresholds like corpus size and term frequency were effective for the entire application. Others like Tf*Idf were only stage specific. For setting and resetting every threshold value we found different sets of topic graphs.

5.1 Test 1: N=10, Term Frequency (Tf) > 1, Intersection Size > 1

We tested five sets of 10 documents. Each of these sets of documents was randomly chosen out of the base corpus of 200 that we currently collected. These documents are NEWS articles collected from different news providing websites. They range for different topics like politics, movies, celebrity, sports etc. We got the following results.

1. **Files 1 to 10 in main Corpus**

Dortmund
    Children: Real Madrid
Real
    Children: Dortmund Maria Borussia Di League Champions
Madrid
    Children: We Klopp BVB

[Dortmund Real Madrid]
[Real Maria Borussia Di]
[Real Borussia Di]
[Real League Champions]
[Madrid We BVB]
Number of trees = 3
Number of Sets = 4

2. Files 11 to 20 in main Corpus

Dortmund
Children: Real Madrid
Real
Children: Dortmund
Ibrahimovic
Children: Chelsea PSG League Paris Champions
Chelsea
Children: Hazard Oscar Luiz Eden Mourinho French Ibrahimovic

[Dortmund Real Madrid]
[Ibrahimovic Chelsea PSG League Paris]
[Ibrahimovic PSG League Paris]
[Ibrahimovic League Paris Champions]
[Chelsea Hazard Oscar Luiz Eden]
[Chelsea Oscar Luiz Eden]
[Chelsea Luiz Eden]

Number of trees = 4
Number of Sets = 7

3. Files 51 to 60 in main Corpus

Brooke
Children: Mueller Charlie Sheen Rossi Brett Bob Radar Sheen's
Mueller
Children: Brooke
Charlie
Children: Richards Charlie's Denise Sam Sheen's

[Brooke Mueller Charlie Sheen Rossi Brett Bob Radar Sheen's]
[Brooke Charlie Sheen Rossi Brett Bob Radar Sheen's]
[Brooke Sheen Rossi Brett Bob Radar Sheen's]
[Brooke Rossi Brett Bob Radar]
[Brooke Brett Bob Radar Sheen's]
[Charlie Richards Denise Sam]
[Charlie Charlie's Denise Sam Sheen's]
[Charlie Denise Sam Sheen's]
[Charlie Sam Sheen's]

Number of trees = 3
Number of Sets = 9

4. Files 79 to 88 in the main Corpus

Mahatma
    Children: AAP Arvind Kejriwal Delhi
AAP
    Children: Mahatma Gandhi Congress Fakhruddin BJP Party Aadmi Aam
Gandhi
    Children: Rae Bareli Sonia Justice Archana
Soldier
    Children: America Marvel Captain Winter
America
    Children: Soldier

[Mahatma AAP Arvind Kejriwal Delhi]
[Mahatma Arvind Kejriwal Delhi]
[Mahatma Kejriwal Delhi]
[AAP Gandhi Congress Fakhruddin]
[AAP Congress BJP]
[AAP BJP Party Aadmi Aam]
[AAP Party Aadmi Aam]
[AAP Aadmi Aam]
[Gandhi Rae Bareli Sonia Justice Archana]
[Gandhi Bareli Sonia Justice Archana]
[Gandhi Sonia Justice Archana]
[Gandhi Justice Archana]
[Soldier America Marvel Captain Winter]
[Soldier Marvel Captain Winter]
[Soldier Captain Winter]

Number of trees: 5
Number of sets: 15

5. Files 103 to 112 in the Main Corpus

Samsungs
    Children: S5 Galaxy Samsung
S5
    Children: Samsungs
Singh
    Children: Yuvrajs Yuvraj Sri Yuvi Cup World T20 India Indian
Yuvrajs
    Children: Singh Twenty20 Indias Sunday
Yuvraj
    Children: Dhoni Raina Kohli

[Samsungs S5 Galaxy Samsung]
[Samsungs Galaxy Samsung]
[Singh Yuvrajs Yuvraj Sri Yuvi Cup World T20 India]
[Singh Yuvraj Sri Yuvi Cup World T20 India Indian]
[Singh Sri Cup World T20 India]
[Singh Yuvi Cup World T20 India Indian]
[Singh Cup World T20 India Indian]
[Singh World T20 India Indian]
[Singh T20 India]
[Singh India Indian]
[Yuvrajs Twenty20 Sunday]
[Yuvraj Dhoni Raina Kohli]
[Yuvraj Raina Kohli]

Number of trees: 5
Number of sets: 13

The first five test subjects presented the following observations:

1. Each set of 10 documents had more than one topic tree and the number of topic sets obtained from these was more than the number of trees in the documents.
2. Corpus sets 1 and 3 contained documents that were related to same base subject and presented topic sets strictly related to that base subject. For example: Set 1 which had

articles about the recently concluded soccer match between Real Madrid and Borussia Dortmund created trees and topic sets that didn't stray away from these base subjects.

3. Corpus 2 on the other hand though again about soccer, had articles about two different events that occurred. Of the seven topic sets were created for this particular corpus, one was for the first event and the rest were for the second event. Observing these topic sets, it was clear that they were event separated i.e. no topic set of a particular event had any term related to the other event.
4. Corpuses 4 and 5 on the other hand have documents related to entirely different subjects. 4 contain articles about politics and movies while 5 contain articles about technology and sports. Again the topic sets and tree recovered from these sets are separated based on the base subjects.

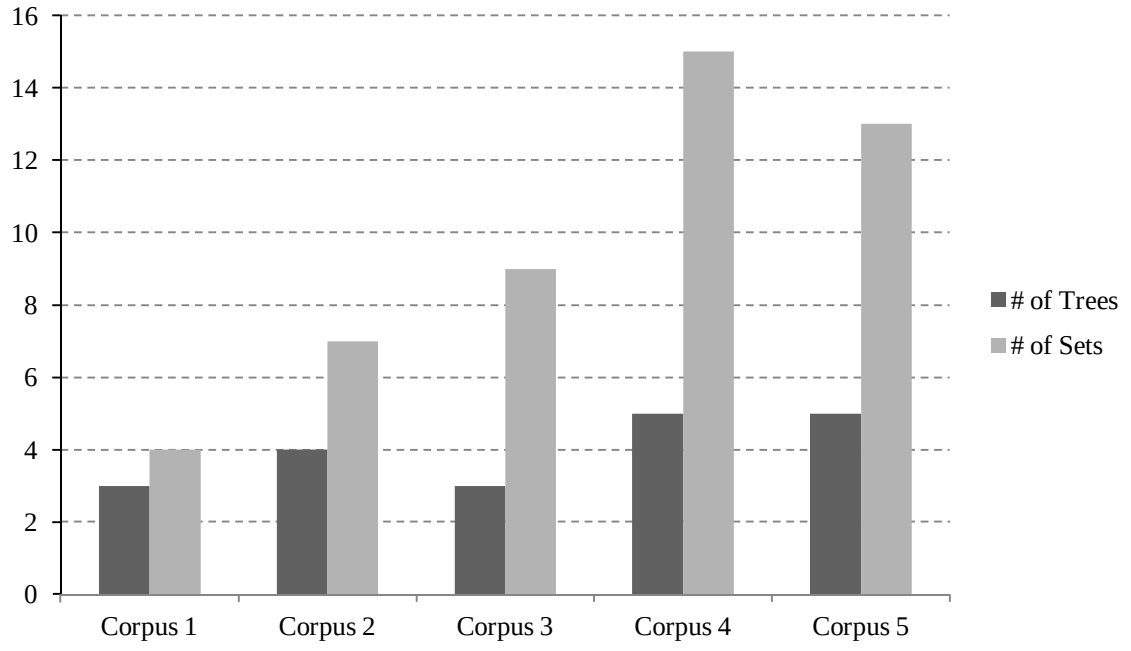

Figure 5.1: Number of Topic Trees and Topic Sets for five different corpuses. N=10

5.2 Test 2: N=20, Tf >= Average Frequency, Intersection Set Size > 3

Again we tested five sets but this time, each set had 20 documents, randomly chosen out of the base corpus of 200 that we currently collected. Only this time, our thresholds were stingier. We set the Term frequency threshold to the average Tf of all the terms inside the documents and in Stage 3, the set intersection between the document lists of two had the minimum size of 3 i.e. two terms were set to have an edge in the term graph only if they appeared in at least four documents or more. We got the following results.

1. **Files 1 to 10 in main Corpus**

Chelsea
    Children: PSG Ibrahimovic
PSG
    Children: Chelsea
Dortmund
    Children: Real Madrid
Real
    Children: Dortmund
Madrid
    Children: League Champions

[Chelsea PSG Ibrahimovic]
[Dortmund Real Madrid]
[Madrid League Champions]

Number of trees: 5
Number of sets: 3

2. **Files 25 to 44 in main Corpus**

Prix
    Children: Bahrain Grand
Bahrain
    Children: Prix Williams Ferrari Alonso

[Prix Bahrain Grand]
[Bahrain Williams Ferrari Alonso]
[Bahrain Ferrari Alonso]

Number of trees: 2
Number of sets: 3

3. **Files 47 to 66 in the main Corpus**

Mueller
    Children: Charlie Sheen Brett
Charlie
    Children: Mueller

[Mueller Charlie Sheen Brett]
[Mueller Sheen Brett]

Number of trees: 2
Number of sets: 2

4. **Files 71 to 90 in the main Corpus**

Man
    Children: America Soldier Marvel Captain Iron Winter
America
    Children: Man Agents Avengers SHIELD
AAP
    Children: Kejriwal Delhi BJP
Kejriwal
    Children: AAP

[Man America Soldier Marvel Captain Iron Winter]
[Man Soldier Marvel Captain Iron Winter]
[Man Marvel Captain Iron Winter]
[Man Captain Iron Winter]
[Man Iron Winter]
[AAP Kejriwal Delhi]
[AAP Delhi BJP]

Number of trees: 4
Number of sets: 7

5. Files 96 to 115 in the main Corpus

Samsungs
    Children: S4 S5 Galaxy Samsung
S4
    Children: Samsungs
S5
    Children: S
Singh
    Children: Yuvrajs Yuvraj Sri World India Cup
Yuvrajs
    Children: Singh T20
Yuvraj
    Children: Indian Dhoni Yuvi

[Samsungs S4 S5 Galaxy Samsung]
[Samsungs S5 Galaxy Samsung]
[Samsungs Galaxy Samsung]
[Singh Yuvrajs Yuvraj Sri World India]
[Singh Yuvraj Sri World India Cup]
[Singh Sri World India]
[Singh World India Cup]

Number of trees: 6
Number of sets: 7

This set of test subjects presented the following observations:

1. The ‘number of Topic Trees’ to “number of Topic Sets’ ration for each set of 20 documents varied. The fact that we set certain lower limits on the Term Frequencies and the document lists intersection made that the term graphs obtained in stage 3 were smaller for this testing phase than those observed in the first testing phase.
2. The size of the topic trees obtained was proportional to the number of topic sets obtained for that particular set of topic trees. By size of Topic Tree, we mean that no parent in the trees of a particular corpus had minimal number of children. For instance, no tree in corpus 1 had more than two children and thus even though we had five trees, we had only three sets. This pattern of Maximum number of children to number of sets proportion was observed throughout this test case.
3. But the fact that topic sets of trees corresponding to different base subjects did not mingle with each other – which was evident in the first test case – was observed in this test case as well.
4. Thus not maintaining direct proportionality for increase in size of corpuses, Term frequency and Intersection set size had non proportional results as far as sets and trees were concerned.

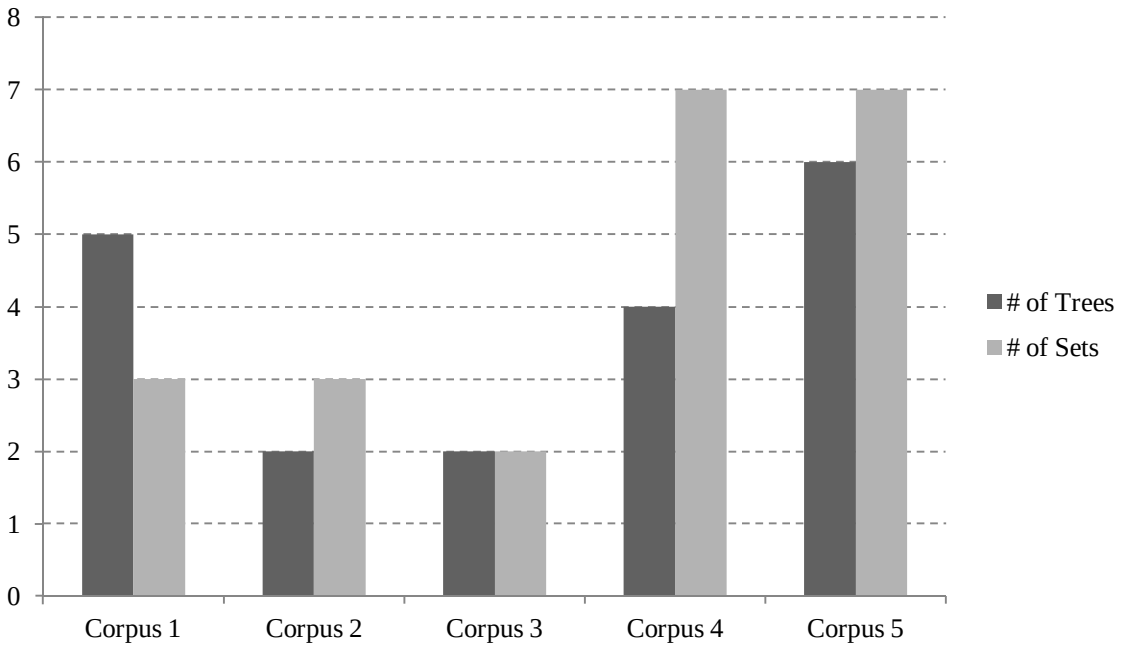

Figure 5.2: Number of Topic Trees and Topic Sets for five different corpuses. N=20

5.3 Test 3: N=50, Tf >= Average Frequency, Intersection Set Size > 3

Next up, we keep the thresholds as they were before but the number of documents in each corpus is 50. We have three sets of corpuses, but the number of documents has drastically increased. Although the thresholds can remain the same, because the number of articles for each base subject like soccer, politics etc varies between 10 and 30 and we don't have the exact number for every subject since have been randomly chosen.

1. **Files 5 to 54 in the main Corpus**

Mueller
Children: Charlie Sheen Brett
Charlie
Children: Mueller
Williams
Children: Bahrain Ferrari Nico Alonso

Bahrain
    Children: Prix Grand Williams Raikkonen Mercedes
Prix
    Children: Championship World
League
    Children: Madrid Champions
Madrid
    Children: League
Chelsea
    Children: PSG Ibrahimovic
PSG
    Children: Chelsea

[Mueller Charlie Sheen Brett]
[Mueller Sheen Brett]
[Williams Bahrain Ferrari Nico Alonso]
[Williams Ferrari Alonso]
[Williams Nico Alonso]
[Bahrain Prix Grand]
[Prix Championship World]
[League Madrid Champions]
[Chelsea PSG Ibrahimovic]

Number of trees: 9
Number of sets: 9

2. Files 61 to 110 in the main Corpus

AAP
    Children: Kejriwal Delhi BJP
Kejriwal
    Children: AAP
Marvel
    Children: Man America Soldier Captain Avengers Iron Winter SHIELD
Man
    Children: Marvel
America
    Children: Agents World
World

Children: Yuvrajs Yuvraj Cup T20

S

Children: S4 S5 Samsungs Galaxy Samsung

S4

Children: S HTC

S5

Children: Android

[AAP Kejriwal Delhi]
[AAP Delhi BJP]
[Marvel Man America Soldier Captain Avengers Iron Winter]
[Marvel America Soldier Captain Avengers Iron Winter SHIELD]
[Marvel Soldier Captain Avengers Iron Winter SHIELD]
[Marvel Captain Avengers Iron Winter SHIELD]
[Marvel Avengers Winter SHIELD]
[Marvel Iron Winter]
[Marvel Winter SHIELD]
[World Yuvrajs Yuvraj]
[World Yuvraj Cup T20]
[S S4 S5 Samsungs Galaxy Samsung]
[S S5 Samsungs Galaxy Samsung]
[S Samsungs Galaxy Samsung]
[S Galaxy Samsung]

Number of trees: 9
Number of sets: 15

3. Files 37 to 86 in the main Corpus

Charlie

Children: Mueller Sheen Brett

Mueller

Children: Charlie

Marvel

Children: Man America Soldier Captain Avengers Iron Winter SHIELD

Man

Children: Marvel

America

Children: Agents World
AAP
Children: Kejriwal Delhi
Kejriwal
Children: AAP
Bahrain
Children: Ferrari Alonso
Ferrari
Children: Bahrain

[Charlie Mueller Sheen Brett]
[Charlie Sheen Brett]
[Marvel Man America Soldier Captain Avengers Iron Winter]
[Marvel America Soldier Captain Avengers Iron Winter SHIELD]
[Marvel Soldier Captain Avengers Iron Winter SHIELD]
[Marvel Captain Avengers Iron Winter SHIELD]
[Marvel Avengers Winter SHIELD]
[Marvel Iron Winter]
[Marvel Winter SHIELD]
[AAP Kejriwal Delhi]
[Bahrain Ferrari Alonso]

Number of trees: 9
Number of sets: 11

The particular test case presented the following observations:

1. Since the documents were randomly divided amongst three or four different base subjects, there were quite a few numbers of topic trees and sets for every corpus of 50 documents. Here we observed that the number of topic sets formed were more than or equal to the number of topics trees created.
2. Also there was no inter-mixing of any topic trees or sets. Thus based on the first three tests we could say that, the intersection of topic sets achieved from topic trees formed out

of document sets belonging to two different genres, subjects or fields is found to be empty. Out topic sets thus can be said to very subject specific but categorized.

3. The third observation that we could figure out was that for the same set of thresholds (Tf > average, Document List Intersection set size > 3) as followed in test case 2, the increase in number of documents led to increase in the 'Number of sets' to 'Number of trees' ratio which also was constantly more than or equal to 1. The fact that the number of documents belonging to each field or base subject was unknown didn't make much of a difference.

5.4 Test 4: N=100, Tf >=$\frac{Average\ Frequency+Min}{2}$, Intersection Set Size > 4

This test case considered two corpuses of 100 documents each. We consider a new threshold for Term Frequency, that was midway between the average and the minimum frequency observed in particular document. Also the intersection set size was at least 5 in this case. The corpuses that were used were as follows:

1. Files 1 to 100 in the main Corpus
2. Files 21 to 120 in the main Corpus
3. Files 11 to 60 and 71 to 120 in the main Corpus
4. Files 5 to 54 and 61 to 110 in the main Corpus
5. Files 1 to 25, 31 to 55, 61 to 85 and 91 to 115 in the main Corpus

Figure 6.3 presents the results for these test cases in a graph format. Since the number of sets and trees observed were of a large quantity, it would have been really tedious to present the contents of every set.

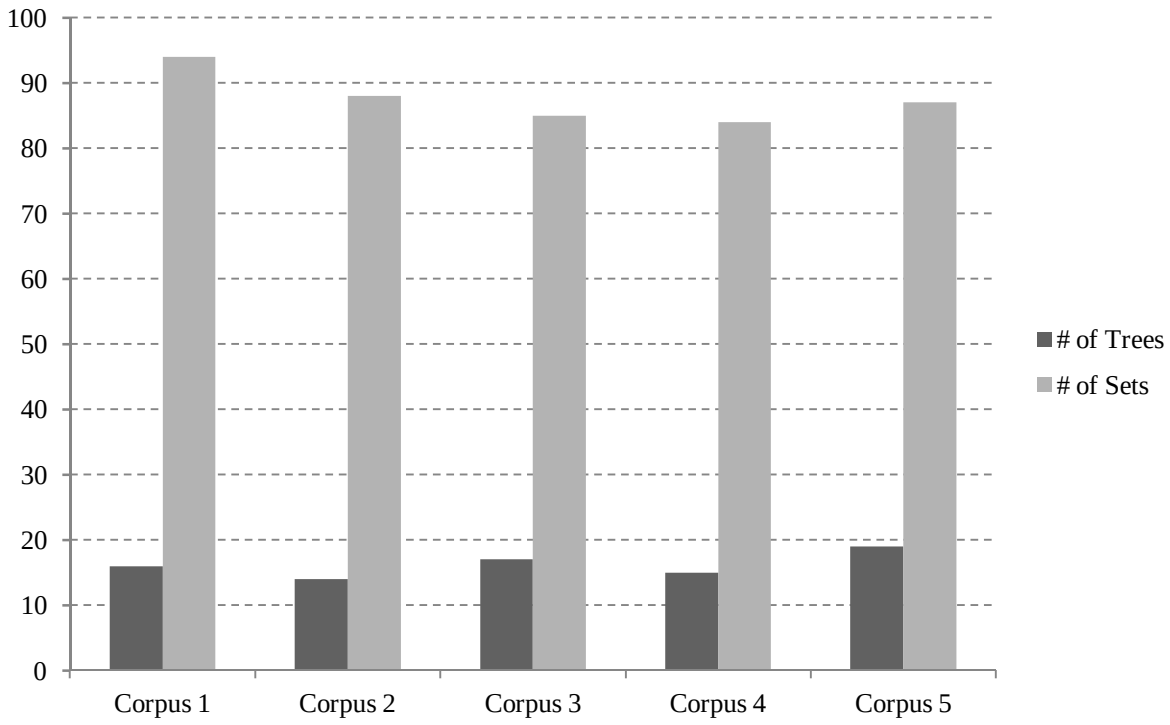

Figure 5.3: Number of Topic Trees and Topic Sets for five different corpuses. N=100

This particular test case presented with the following observations:

1. The number of sets to number of trees ratio was significantly higher than what was observed in the previous case. The thing to note here is that the main contributing factor towards this was decreasing the Tf threshold and not the number of documents.

Based on the four test cases we performed over a variety of documents spanning over different fields and prospects of the real world, we formulated various topic sets and topic trees. We observed that in most cases and for most trees, the number of sets was significantly more than trees. We can thus infer that a single topic tree can create at least one topic sets.

CHAPTER 6

RELATED WORK AND FUTURE PROSPECTS

Haphazardly arranged information is not information but just data of no importance. Information retrieval is an abundantly improving commodity today especially over the web. With the rise of social networking people expect more from the internet more than ever before. Ranking is another evolving aspect to organize information over the web. A set various standards exists to rank and index information. These theories are not limited to the internet but our aspects of our materialistic lives as well. Our introduction of topic graphs and topic sets can be used as complementary to both ranking and indexing techniques. Ranking and indexing documents by categorizing them on the basis of the topic sets that we would provide would enhance the information retrieval process. The same can be done to improvise the ranking and the indexing of social micro-blogging web-services like twitter and facebook. Visual information retrieval can be applied the same theory as well. Images, videos and other multimedia searches can divided based on the same concept of topic graphs. Any information that has some subject based classification associated with it can be ranked, indexed, categorized and classified basis of some sets of topic graphs.

Topic Maps: Topic Maps is a standard for the representation and interchange of knowledge, with an emphasis on the find ability of information. Topic maps were originally developed in the late 1990s as a way to represent back-of-the-book index structures so that multiple indexes from different sources could be merged. However, the developers quickly realized that with a little additional generalization, they could create a meta-model with potentially far wider application [16].

A topic map represents information using

- Topics, representing any concept, from people, countries, and organizations to software modules, individual files, and events,
- Associations, representing hyper graph relationships between topics, and
- Occurrences representing information resources relevant to a particular topic.

Topic Maps are similar to concept maps and mind maps in many respects, though only Topic Maps are ISO standards.

Ontology: In computer science and information science, an ontology formally represents knowledge as a hierarchy of concepts within a domain, using a shared vocabulary to denote the types, properties and interrelationships of those concepts [7][8].

Ontologies are the structural frameworks that are used in information organization. Their utilization ranges from various fields artificial intelligence, the Semantic Web, systems engineering, software engineering to biomedical informatics, library science, enterprise bookmarking, and information architecture. Ontologies can be used for knowledge representation

about a host of topics or just a part of them. The creation of domain ontologies is also fundamental to the definition and use of an enterprise architecture framework [7] [8].

Substantial work has been performed on the translation of natural language questions to formal queries using ontology or a database [5] [3] [4] [6]. While these approaches have been shown to yield remarkable results, it is not clear if users always want to specify a full natural language question. In fact, the success of commercial search engines shows that users are quite comfortable with using keywords. Thus, it seems important to also develop approaches which are able to interpret keywords

Future aspects of Topic graphs can include an assortment of uses ranging from basic everyday uses to the realms of the World Wide Web. We wish to pursue the use of topic sets to develop a new indexing scheme for any web based search. Based on the parent child relationship of the words, we can present a ranking scheme for topic graphs. We can use our topic sets to urge a crawler to find more documents that subject to a particular topic sets. These in turn can be ranked and index with the terms of the graphs to provide a more categorically based ranking and help in better information retrieval.

CHAPTER 7

CONCLUSION

We present “Topic Graphs” and “Topic Sets”, a probabilistic relationship based association words to cluster and categorize topics together into a hierarchical tree based format. This tree format can be further used to create topic sets which present templates of words that have high associations with each other. These sets contain words that have a high probability of appearing together across a number of documents in the world. These words thus are strongly related to each other.

To prove the existence of these topic sets we began by processing a certain corpus of documents. We filtered out unnecessary and unwanted words out to keeps the more common but popular and important words in each document. We then used graph theory to formulate relationships between these more popular terms. We treated every term as a vertex and the weighted edges between them defined the strength of their relationships. Based on this relationship we created topic graphs which essentially are topic trees. The parent child relationship between these topic graphs helped us to formulate the required topic sets.

We used a well defined and categorized corpus to test our theory. We created a through application in Java for every stage of the system to test our theory. The application yielded comprehensive topic graphs and topic sets. To prove the logical existence of the theory we also presented an on paper example for a small corpus of four documents each with an average of 120 words. We extracted results that provided the evidence for the nature of the thesis.

Towards the end we presented various applications along with related and future aspirations for our theory to grow further in the web and non technical world. We believe that such kind of categorizing will help precise and efficient searching of information.

BIBLIOGRAPHY

[1] Lewis, Chris. "Information Patterns." *Irresistible Apps*. Apress, 2014. 81-97.

[2] Weisberg, Jacob. "Bubble trouble: Is web personalization turning us into solipsistic twits." (2011).

[3] Lopez Lopez, Vanessa, Michele Pasin, and Enrico Motta. "Aqualog: An ontology-portable question answering system for the semantic web." *The Semantic Web: Research and Applications*. Springer Berlin Heidelberg, 2005. 546-562.

[4] Cimiano, Philipp, Peter Haase, and Jörg Heizmann. "Porting natural language interfaces between domains: an experimental user study with the orakel system." *Proceedings of the 12th international conference on Intelligent user interfaces*. ACM, 2007.

[5] Popescu, Ana-Maria, Oren Etzioni, and Henry Kautz. "Towards a theory of natural language interfaces to databases." *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003.

[6] Bernstein, Abraham, and Esther Kaufmann. "GINO–a guided input natural language ontology editor." *The Semantic Web-ISWC 2006*. Springer Berlin Heidelberg, 2006. 144-157.

[7] Gruber, Thomas R. "A translation approach to portable ontology specifications."

Knowledge acquisition 5.2 (1993): 199-220.

[8] Fensel, Dieter. *Ontologies*. Springer Berlin Heidelberg, 2001.

[9] Pariser, Eli. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[10] Zhang, Yuan Cao, et al. "Auralist: introducing serendipity into music recommendation." *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012.

[11] Cormen, Thomas H., et al. *Introduction to algorithms*. Vol. 2. Cambridge: MIT press, 2001.

[12] Kolosovskiy, Maxim A. "Data structure for representing a graph: combination of linked list and hash table." *arXiv preprint arXiv:0908.3089* (2009).

[13] Gertz, Janet. "Selection for preservation in the digital age." *Library Resources & Technical Services* 44.2 (2000): 97-104.

[14] Rajaraman, Anand, and Jeffrey David Ullman. Data Mining. Cambridge University Press, 2011.

[15] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Vol. 1. Cambridge: Cambridge university press, 2008.

[16] Pepper, Steve, and Graham Moore. "XML topic maps (XTM) 1.0." *TopicMaps. Org Specification xtm1-20010806* (2001).

[17] Gao, Rujia, et al. "Application of Full Text Search Engine Based on Lucene." *Advances in Internet of Things* 2 (2012): 106.

[18] Chen, Chun, et al. "Ti: an efficient indexing mechanism for real-time search on tweets." *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011.

[19] Kules, Bill, Jack Kustanowitz, and Ben Shneiderman. "Categorizing web search results into meaningful and stable categories using fast-feature techniques." *Digital Libraries, 2006. JCDL'06. Proceedings of the 6th ACM/IEEE-CS Joint Conference on*. IEEE, 2006.

[20] White, Ryen W., Bill Kules, and Steven M. Drucker. "Supporting exploratory search, introduction, special issue, communications of the ACM." *Communications of the ACM* 49.4 (2006): 36-39.

[21] Rose, Daniel E., and Danny Levinson. "Understanding user goals in web search." *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004.

[22] Baeza-Yates, Ricardo, Carlos Hurtado, and Marcelo Mendoza. "Query recommendation using query logs in search engines." *Current Trends in Database Technology-EDBT 2004 Workshops*. Springer Berlin Heidelberg, 2005.

[23] Pratt, Wanda. "Dynamic organization of search results using the UMLS." *Proceedings of the AMIA Annual Fall Symposium*. American Medical Informatics Association, 1997.

[24] Landauer, Thomas, et al. "Enhancing the usability of text through computer delivery and formative evaluation: the SuperBook project." *Hypertext: A psychological perspective* (1993): 71-136.

[25] Maarek, Yoelle S., et al. "WebCutter: a system for dynamic and tailorable site mapping." *Computer networks and ISDN systems* 29.8 (1997): 1269-1279.

[26] Zamir, Oren, and Oren Etzioni. "Grouper: a dynamic clustering interface to Web search results." *Computer Networks* 31.11 (1999): 1361-1374.

[27] Zamir, Oren, and Oren Etzioni. "Web document clustering: A feasibility demonstration." *Proceedings of the 21st annual international ACM SIGIR conference on Research and*

development in information retrieval. ACM, 1998.

[28] Hearst, Marti A., and Jan O. Pedersen. "Reexamining the cluster hypothesis: scatter/gather on retrieval results." *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1996.

[29] Hearst, M., J. Pedersen, and D. Karger. "Scatter/gather as a tool for the analysis of retrieval results." *Working notes of the AAAI fall symposium on AI applications in knowledge navigation*. 1995.