

ABSTRACT

Improving Access to Information through Conceptual Classification

by Sahar Bazargani

July, 2011

Director of Thesis: Dr. M. H. Nassehzadeh Tabrizi

Major Department: Computer Science

Overwhelming the users with large amount of information on the Web has resulted in users' inability to find the information and their dissatisfaction with available information searching and filtering systems. On the other hand, the information is distributed over many websites and a large part of it (for example news) is updated frequently. Keeping track of the changes in huge amount of information is a real problem for users.

Due to the great impact the information has on people's lives and business decision-making, much research has been done on the efficient ways of accessing and analyzing the information. This thesis will propose a conceptual classification method and ranking of the information in order to provide better user access to a wider range of information, it also provides the information that may help in analyzing the global trends in various fields. In order to demonstrate the effectiveness of this method, a feed aggregator system has been developed and evaluated through this thesis.

To improve the flexibility and adaptability of the system, we have adopted the agent-oriented software engineering architecture that has also helped facilitating the development process. In addition, since the system deals with storing and processing large amounts of

information, that requires a large number of resources the cloud platform service has been used as a platform for deploying the application. The result was a cloud based software service that benefited from the unlimited on-demand resources.

To take advantage of the available features of public cloud computing platforms, those supporting the agent-oriented design, the multi-agent system was implemented by mapping the agents to the cloud computing services. In addition, the cloud queue service that is provided by some cloud providers such as Microsoft and Amazon was used to implement indirect communication among the agents in the multi-agent system.

Improving Access to Information through Conceptual Classification

A Thesis

Presented to the Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Software Engineering

by

Sahar Bazargani

July, 2011

©Copyright 2011
Sahar Bazargani

Improving Access to Information through Conceptual Classification

by

Sahar Bazargani

APPROVED BY:

DIRECTOR OF THESIS: _____
M. H. Nassehzadeh Tabrizi, PhD

COMMITTEE MEMBER: _____
Junhua Ding, PhD

COMMITTEE MEMBER: _____
Sergiy Vilkomir, PhD

COMMITTEE MEMBER: _____
Qin Ding, PhD

CHAIR OF THE DEPARTMENT OF COMPUTER SCIENCE:

Karl Abrahamson, PhD

DEAN OF THE GRADUATE SCHOOL:

Paul J. Gemperline, PhD

TABLE OF CONTENT

LIST OF TABLES	v
LIST OF FIGURES	vi
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: BACKGROUND AND MOTIVATION.....	6
2.1 RSS	7
2.2 Atom.....	8
2.3 Weaknesses of Feed Readers/Aggregators.....	9
2.4 Information Processing and Knowledge	11
CHAPTER 3: THE FEED ANALYZER SYSTEM.....	13
3.1 Conceptual Information Classification	15
3.2 Library of Congress Subject Heading.....	17
3.3 Providing Efficient Access to Information through Sorting the Result.....	19
3.4 Implementing the Solution in Feed Analyzer.....	22
3.4.1 Classification of the Required Information.....	22
3.4.2 Leveraging Other Technologies	24
CHAPTER 4: AGENT-BASED SOFTWARE ENGINEERING.....	25
4.1 Multi-Agent System	25
4.2 The Reasons behind choosing Agent-Based Software Engineering for Feed Analyzer	27
4.3 Agent-oriented Software Engineering in Feed Analyzer System.....	30
4.4 Use Case Model.....	31
4.5 Designing the System	33
CHAPTER 5: MULTI-AGENT SYSTEM IN THE CLOUD	Error! Bookmark not defined.
5.1 Advantages of the Cloud Computing	38
5.2 Cloud Computing Layers and Categories.....	40
5.3 Advantages of Developing Multi-Agent System as a Cloud Service.....	42
5.4 Brief Description of the Cloud Platform Services.....	45
5.5 Comparison of Cloud Platforms	49
5.6 Feed Analyzer's System Implementation in Cloud Environment	51
CHAPTER 6: IMPLEMENTATION OF FEED ANALYZER SYSTEM.....	53

6.1 Importing Library of Congress Subject Heading Dataset to the System.....	53
6.1.1 Populating Subject Tables.....	54
6.1.2 Subject Codes.....	55
Subject code generation and challenges	57
6.1.3 Extracting Subject Headings from Downloaded File	58
Resource Description Framework	59
Parsing the Library of Congress subject heading file.....	61
6.2 Feed Analyzer Cloud Roles.....	64
6.2.1 Manager Worker Role.....	64
6.2.2 Feed Reader Worker Role.....	64
6.2.3 Feed Indexer Worker Role	66
6.2.4 Feed Analyzer Service Web Role	67
Subject ranking method in feed analyzer.....	70
6.3 Indexing and Searching Web Feeds	72
6.3.1 Using SQL Server 2008 Full Text Search Engine	75
6.3.2 Methods of Conceptual Searching in Feed Analyzer System.....	79
Method 1	79
Method 2.....	83
Method 3.....	84
6.3.3 Conceptual Searching Implementation	93
6.4 Database Diagram	98
CHAPTER 7: THE RESULTS	100
7.1 Comparing the Result of Conceptual Searching with Keyword Searching	100
7.2 Ranking and Sorting the Subject Report	106
Calculating the subject's rank	109
7.3 Ranking and Sorting the Feed Report	110
7.4 Impact of Using an Agent-Oriented Approach and Cloud Computing.....	110
7.5 Comparison with Other Systems	111
CHAPTER 8: CONCLUSION AND FUTURE WORK.....	115
8.1. Conclusion.....	115
8.2 Future work	117

REFERENCES	119
APPENDIX A: SEARCH RESULTS OF RANDOM SUBJECTS USED IN EVALUTION OF THE THESIS RESULT.....	123
APPENDIX B: COMPARISON OF SEARCH RESULT FOR ALL SUBJECTS	131

LIST OF TABLES

1. Table 5.1: Agents' Mapping to the Azure Roles.....51
2. Table A.1: Result Set Size, Searching Subjects and their Alternate Labels.....123
3. Table A.2: Result Set Size, Searching Subjects and their Related Terms.....125
4. Table A.3: Result Set Size, Searching Subjects and their Narrower Terms.....127

LIST OF FIGURES

1. Figure 2.1: Feed Icon, Downloaded from [9].....	6
2. Figure 3.1: Subject Relations in the Thesaurus.....	19
3. Figure 4.1: Feed Analyzer Use Case Diagram.....	33
4. Figure 4.2: Agents and Their Communication in Feed Analyzer System.....	35
5. Figure 4.3: Sample Scenario in the System from Agents' Point of View.....	37
6. Figure 5.1: Cloud Computing Architecture.....	41
7. Figure 6.1: Conceptual Design of Subject Heading in the Feed Analyzer.....	55
8. Figure 6.2: RDF Graph for Eric Miller Description, from "RDF Premier" [43]	60
9. Figure 6.3: Subject Heading Tables and their Relationships after Normalization.....	63
10. Figure 6.4: Conceptual Design of Feed Information.....	66
11. Figure 6.5: Feed Analyzer System - New Favorite Subject Page	68
12. Figure 6.6: Sample Subject Report in the Feed Analyzer System.....	69
13. Figure 6.7: Result Set of sys.dm_fts_index_keywords_By_Document Function	83
14. Figure 6.8: Execution Plan of Searching One Subject in Feeds with Method 2.....	85
15. Figure 6.9: _SubjectFeedRelation Table Structure.....	87
16. Figure 6.10: Comparison of Query Time per Subject between the Searching Method.....	89
17. Figure 6.11: Sample Subject Report.....	98
18. Figure 6.12: Feed Analyzer's Database Diagram	99
19. Figure 7.1: Comparison of Search Result (Subject with and without Alternate Labels).....	102
20. Figure 7.2: Comparison of Search Results (Subject with and without Related Terms).....	103
21. Figure 7.3: Comparison of Search Results (Subject with and without Narrower Terms)....	103
22. Figure 7.4: Figure 7.4: Comparison of Subject Search and Conceptual Search	104

23. Figure 7.5: A User's Subject Report for April.....	107
24. Figure 7.6: A User's Subject Report for May.....	107
25. Figure 7.7: A User's Subject Report for June.....	108
26. Figure B.1: Comparison of Search Results for All Subjects in Test Set (with and without Alternate Labels).....	131
27. Figure B.2: Comparison of Search Results for All Subjects in Test Set (with and without Related Terms)	132
28. Figure B.3: Comparison of Search Results for All Subjects in Test Set (with and without Narrower Terms).....	133
29. Figure B.4: Comparison of Subject Search and Conceptual Search for All the Subjects in Test Set.....	134

CHAPTER 1: INTRODUCTION

As Herbert Mayer wrote in his book “How to Analyze Information” [1] we live in an information revolution decade and so much information about a variety of subjects is produced and published on the Web. The advances in Web and information technology have had a great impact on people’s lives and society. Information can be accessed and used by people in a variety of ways that allows them to analyze their decisions and actions in order to achieve better results, more success, and less strife in difficult situations. This information can have considerable impact on many peoples’ lives and in a broader sense, on the world’s business, economy, and society. Social or political events which are broadcasted through news over the Web affect business decisions. The simplest example of this is with weather forecasts, which can impact an individual’s plans for a trip or a company’s scheduled event.

Although using information accessible through the speedy Web appears to be very desirable and effective, increasing the amount of information available on the Web – the majority of which is uncategorized - has led to considerable difficulty in accessing the most desired data. Everyone has experienced the feeling of being overwhelmed by search results in everyday life, even when using the most famous search engines on the Web.

In May 2008, the Internet content syndication council [2,3] announced that Net Craft crawler had crawled 165.7 million Internet sites by April 2008. This amount is obviously less than the total number of websites on the Web, but it shows that the vast amount of information which is published on the Web is enormous. One of the main reasons behind the fast and dramatic growth of information in recent years is the growth of information producers on the Web. Many more people have access to the Web and produce new information in many different forms including, but not limited to, weblogs and social networks [4].

Despite all the efforts of researchers and search engine companies in improving information clustering and retrieval, the rate of information growth on the Web is much faster than the rate of relevant improvement. The main drawback of dealing with huge amount of uncategorized information is avoiding users from efficient access to information they need and in some cases, inability to access the information they look for.

However, this is not the only problem users have. Information on the Web is distributed over a substantial number of websites. In the May 2008 report of the Internet content syndication council, the estimated number of websites was 45 billion [3].

In addition, the content of many of these websites is updated frequently without a schedule. Therefore, users often need to check these websites (of course not every website, but all of the ones they are most interested in) more frequently. Usually, news websites fit into this category. The frequency of their updates depends upon the events. For example, the content of a sports news website may be updated more frequently during Olympic Games or other international games when sports game scores needs to get updated continuously. These two characteristics of information on the Web lead to an exhaustive and unpleasant experience for users if they want to keep track of the most current news, headlines, or updates to this type of information.

From a business point of view, this problem is more serious since information has a great impact on the performance of their services and on their profits. A survey [5], conducted by the Avanade company [6] in 2010, showed that despite all of the challenges that business and IT executives had with the overflow of information, they still believed in the importance of the most current data in improving their business forecasts, and in decreasing the uncertainty in making decisions and improving their companies' positions in comparison to that of their competitors.

According to the survey [5], the first priority of company executives was faster access to data in order to understand their customers' expectations of their services and to keep up with their customers. Twenty-three percent of the data that their companies received was produced by customers and 12% of it was public data. Companies extracted this information from many resources including email, customer databases, social media news feeds, RSS feeds, and online portals. However, 43% of all companies surveyed were not satisfied with the filtering tools that they use for extracting information and 46% of these companies had the unfortunate experience of making incorrect decisions as a result of invalid or out-dated data. The survey concluded that addressing the failure in filtering data is crucial in order to allow businesses to react to business conditions in a sufficient timeframe.

The above users' problems in dealing with the accessing the information on the Web can be organized into two categories:

- A huge amount of uncategorized information which is distributed onto many websites. This problem leads to users' inability to find specific information and the long time required to read and process the information.
- A high frequency of updates of large parts of information on the Web without a specific schedule.

Therefore, with the amount of information and their distribution onto the web, it would be difficult, and in many times impossible, to keep up with the number of changes to all or large parts of this information. This thesis will propose a conceptual classification method and ranking of the information in order to provide better user access to a wider range of information, it also

provides the information that may help in analyzing the global trends in various fields. In order to demonstrate the effectiveness of this method, a feed aggregator system has been developed and evaluated for its performance. To improve the system's architectural flexibility and adaptability, we have adopted the agent-oriented software engineering that has also helped facilitating the development process. In addition, since the system deals with storing and processing large amounts of information, that requires a large number of resources the cloud platform service has been used as the development platform. The cloud programming model and available service has supported the agent-oriented approach used in designing the system. Therefore, the system has been developed as a software cloud service that has benefited from available public cloud platform services.

This thesis will be organized as follows:

Chapter 2 will provide a background on the users' problems with the large amount of frequently updated information. In addition, the Web syndication as an information broadcasting method for frequently updated data will be described. The Chapter will conclude by explaining the impact of growing information on software that uses Web syndication for the reading and collecting of information.

Chapter 3 will discuss the solution that is provided by this thesis.

Chapter 4 will present the agent-oriented software engineering definitions and concepts. It then will describe employing this software engineering model in the development process of the proposed system and will provide the design of the system from an agents' point of view.

Chapter 5 will describe the cloud computing technology. It will then explain the reasons behind choosing cloud computing for deployment of a multi-agent system. The Chapter then will

provide a brief description of three famous public cloud platform services and compares these cloud services in order to identify the best choice for development and deployment of the application. This Chapter will conclude with providing the design of the system from the cloud point of view.

Chapter 6 will present the design and implementation details of the Feed Analyzer system and the challenges in importing subject headings and in indexing the feed information.

Chapter 7 will show the results of this system and will compare it with other feed reader or analyzer systems.

Chapter 8 will present the conclusions and further work.

CHAPTER 2: BACKGROUND AND MOTIVATION

Due to the importance of the information's impact on people's lives and on business, it is necessary to improve the performance of a user's access to the information they need. Fortunately, a broadcasting method called Web syndication has been introduced in order to help the users to keep track of large amount of frequently updated information.

Web syndication is a kind of content distribution method which publishes a website's updated content to other sites and provides users with the new, added, or updated information. "web feed" (or News feed) is a data format used for providing users with frequently updated content [7]. Originally, web feeds were used in syndicating information between different websites but in recent years, they mostly are used among websites and with applications (Web or desktop) called feed reader or feed aggregator [8].

Web feeds contains the short summary of a story on a website with a link to the actual content on that website [8]. It often is displayed in the form of an XML document. The common web feed icon which can be seen in many Web pages is shown in Figure 2.1. In order to receive web feeds, users need the feed reader/aggregator software (Web or desktop application) [4]. The feed reader is an application which pulls the feed contents from the website that the user has subscribed to [4], and it reads feed contents from different sources and collects them into one place. Therefore, users don't need to frequently check the websites.



Figure 2.1: Feed Icon, Downloaded from [9].

Web feed is used for publishing the content of different kinds of websites with continuous frequently updated data including news, podcasts, and blogs [4]. Therefore, instead of checking websites daily or hourly, users can use a feed reader/aggregator and subscribe to the web feeds of any websites, blogs, or articles of interest to them. A feed often includes a short story of the content, a link to the main story, and some other information including the last update time of the feed. An older way of notifying users about the updates to a website's content had been through mail subscription. However, web feeds have more benefits in comparison to mail subscription. The most important advantage is providing an automatically updated list of feed entries which is separated from users' emails. Therefore, users do not need to manually find the feeds in their inbox. Moreover, subscribing and unsubscribing to web feeds are easy and safe.

There are two main web feed formats: RSS and Atom. In the following two sections, the structure of these formats is described in detail. RSS is older and is commonly used by many websites. For the purpose of this thesis, the implemented version of our Feed Analyzer only supports RSS feeds. However, the system was developed in a way that can be adapted to new feed formats with minimal amount of effort.

2.1 RSS

RSS stands for Really Simple Syndication and is a web feed format for publishing Web content which is updated frequently (mostly without a schedule) [10]. It was first invented in 1997 by Netscape. However, the latest RSS version (RSS 2.0) was donated to Harvard Law School by David Winer [10].

RSS is downloaded as an XML file which must be compatible with XML 1.0 specification [11]. An RSS XML document has a <rss> element at the top level followed by a single <channel> element. The Channel element contains the metadata from the channel and the

content of the RSS channel. The Channel element has three required elements and at least 16 optional elements. Following are the required elements of a RSS channel [10]:

- title: The name of the channel usually equals the name of the publisher website.
- link: URL of the website.
- description: The statement that describes the channel.

A channel contains one or more <item> elements. An item contains a "story" or usually a short form of a story and a link to the full content of that story on the website, whereas <item> element may include the complete story. In this case, it does not include the title or link sub elements. Following is a list of most the important sub elements of <item> from RSS 2.0 definition [10]:

- title: The title of the story.
- link: URL of the actual content of the story.
- description: The outline of the item.
- author: The Email address of the writer of the story.
- pubDate: The publication date of the item.

2.2 Atom

Atom is a Web syndication XML-based document format which includes feed's metadata and feed entries. The top level element in Atom document is <feed> which contains metadata and feed entries [12]. The required elements of <feed> element are [12]:

- id: A identifier of the feed story
- title: title of the story
- updated: last updated time of the feed

The <feed> element has a sub element which contains the information about the feed entries and is called <entry>.

The main difference between RSS 2.0 and last version of Atom (Atom 1.0), these two is that although RSS 2.0 allows for plain text or HTML in its content, it does not provide metadata information about the types of the content, while Atom has a careful design of payload [13]. Its content must be labeled explicitly according to its content type (plain text, HTML, XHTML and other forms of the Web contents) [13].

Despite the advantages of using feed readers in collecting frequently updated information from multiple sources, growth of information has created an enormous problem for the users of these applications.

2.3 Weaknesses of Feed Readers/Aggregators

Although using feed reader/aggregator frees users from checking websites frequently in order to keep track of the changes in information, the vast amounts of uncategorized information still remains unresolved. Even users of feed readers/aggregators are dissatisfied with the fast growth of the retrieved web feeds in their feed readers. The reason behind users' dissatisfaction is the fact that by subscribing to a website, the feed reader reads all of the feeds which were published by that website and users simply might not be interested in all of them.

Although the numbers of feeds which are retrieved in an update from different websites at different times are not the same, a range of three to 110 was seen among those which had been read by the Feed Analyzer System. More than 80 percent of updates downloaded about 10-20 web feed items. These values show that in most cases with each update of web feeds from a website, 10 to 20 feeds will be added to the system. Therefore, if a user has subscribed to 10 websites, in each update of the feeds from these websites, roughly 100 to 200 feeds might be added to the user's feed list. Even if the update only occurs once a day, retrieving 100-200 feeds would be an exhaustive task.

Some feed readers/aggregators provide a feed organizing feature for their users. For example, News Crawler [14] allows its users to manually define folders for feed sources in order to group and organize the feeds in terms of their publisher. There are some news aggregators like Google News Aggregator [15] which provide a limited set of predefined categories to organize news. Sharp Reader [16] is an example of the types of feed readers which allow their users to filter feeds by the single term search. However, most feed readers only offer the keyword search.

Only one feed reader application was found which provides a category and subcategory creation feature (New Analyzer [17]). For each user's category, the system only shows the feeds which include the category's label or labels of the subcategories under that category. Unfortunately, this feed aggregator is commercial software and only provides information related to the stock market.

Google Reader [18] is one of the most popular feed readers. However, at the time of this thesis write-up, it only groups feeds based upon their publisher and also performs single keyword filtering. Moreover, it has another good but limited feature for tracking a topic, person or events on four news websites (eBay, Google Blog, Google News, Twitter). Unfortunately, users of this

popular feed reader may experience a problem with the large amount of feed information in their lists and they have requested better filtering options. In addition, a filter add-on has been developed for this feed reader which allows users to define the lists of their like and unlike keywords to filter the feeds by the like keywords and remove the feeds which include unlike keywords [19].

Another powerful feed filtering mechanism is provided by Yahoo Pipe [20]. It allows users to define a structure called pipe. A pipe can filter any form of Web data including feeds, sort the data and group them together. Although it is a flexible tool to filter and categorize information, it has its drawbacks. First, it is not easy to integrate it with all of the feed readers available in the market. Second, the task of creating a pipe is not easy for most normal users. This group of users should leverage shared pipes which has been created and shared by other users. But if they want customization in a pipe, they have to change it through the pipe's tools. The last problem is almost the same as the problem with the search feature of other feed readers: keyword searching. We believe the Keyword searching is not the best mechanism for searching a subject. The reason behind this claim is described later in the chapter.

2.4 Information Processing and Knowledge

In order to find a solution for users' problems with the large influx of information, it is necessary to first have a complete understanding of Web users' needs. Herbert Mayer [1] states that the real requirement of making a good decision or improvement of decision-making is not information but instead the knowledge within information is what people really need. Therefore, people analyze the information in order to extract the knowledge within them and use that acquired knowledge to make decisions and solve problems [1]. According to this fact, if information was processed and a form of knowledge was produced, users wouldn't need to spend

a long time in gathering and processing vast amount of information in order to acquire knowledge that they need.

Even if a low-level knowledge was provided for users, as long as it improves and facilitates the process of decision-making, it has value for users, because this knowledge reduces the amount of time they have to spend analyzing and processing information. This idea was the motivation for the development of our Feed Analyzer system as a way to improve the performance in utilizing the large amount of information on the Web.

CHAPTER 3: THE FEED ANALYZER SYSTEM

As was previously mentioned in previous chapters, information that has been published and broadcasted on the Web has been playing an important role in the personal and business lives of the people in recent years. However, expanding the volume of information on the Web has made it increasingly difficult for users to access to this information in a fast and convenient manner, and sometimes they are unable to even access the information they are seeking.

As mentioned in Chapter 1, a considerable group of websites frequently update their content. Although Web syndication has helped users track frequently updated data, the rapid expansion and sheer volume of information has had a negative impact on the effectiveness of the feed readers and aggregators.

The weakness in the filtering and searching mechanism to access the information, forces the users to review an extremely large list of information including many unrelated items with no connection to their intended favorite subjects. It is necessary to find a solution in order to facilitate accessing large amounts of information. Therefore, this thesis will provide a new method in facilitating access to wider range of information through an information classification method. To examine the effectiveness of this method, software system has been developed to collect, classify, and rank the web feeds from various websites.

The reason behind choosing web feed dataset for implementation of the proposed conceptual classification was that Web syndication is a popular method for tracking frequently updated information on multiple websites. In addition, as the Avanade's survey [5] showed that the feeds are one of the alternative forms of information utilized by business executives for decision making and evaluating their customers' feedback and trends. Therefore, providing an efficient mechanism to access these web feeds may help large groups of individual and business

Web users. Although web feeds include the title and a brief description of the news and articles, they also contain summary of the main points of their actual contents that can be used for the classification. The advantage of using web feeds instead of the actual content is that the size of feeds are much smaller when compared with the actual contents , therefore, storing and processing them is possible with fewer resources. In contrast, the disadvantage of using web feeds instead of actual content is that the information provided is less comprehensive than the full content. However, even the best search engines are unable to extract all of the information on the Web. Therefore, providing a mechanism that would facilitate accessing news and stories based upon their main points improves the efficiency of the information accessing.

Based upon the above discussion, the Feed Analyzer system is a feed aggregator which frequently reads the web feeds from various websites in the RSS format, stores them in the database. The reason behind calling the system “Feed Analyzer” doesn’t indicate that the system analyze the web feeds. The name was chosen to distinguish this system from general feed aggregator. In addition, there are some feed readers and aggregators (for example Market News Analyzer [17]) which perform improved searching and classification of the web feeds and are called Feed Analyzer.

As previously mentioned in Chapter 2, in addition to RSS, there are other feed formats available; yet in order to examine the classification and ranking methods, implementing one feed format will be sufficient. Moreover, the system has been implemented in a way that adapting it to a new feed format will not require major changes.

The Feed Analyzer system classifies web feeds based upon the users’ selected subjects and generates two level reports in order to facilitate accessing the web feeds.

3.1 Conceptual Information Classification

The proposed solution includes classification of large number of web feeds and ranking them. Through classification, the feeds are categorized based upon one of their properties. A web feed contains the title, description, published date, the author, and the feed source which publishes the web feed. Most available feed readers/aggregators provide categorizing information by the published date or the feed source. However, as mentioned before, these features do not necessarily satisfy users, as they require classifying the feed information based upon the semantic of the actual content of the feed. Therefore, users do not need to sift through a long list of feeds in order to find the stories or articles on a specific subject. This type of feed classification allows users to filter feeds by subject in an efficient manner.

On the other hand, using the keyword search to filter information by subject is not the most ideal method, as in the real world, the subject is a concept, not merely a label, and that label is only used to represent the subject. Furthermore, more than one label may be used to represent a subject. For example, consider the subject “Electronic Commerce.” Library of Congress suggests 11 Alternate Labels for this subject including, but not limited to, eBusiness, eCommerce, and Internet Commerce [21]. Alternate Labels are the labels which represent a subject and may be used instead.

However, users may know only one or two of these labels, while other previously known alternates of the subject are used in the news or stories to refer to that specific subject. Therefore, by utilizing keyword search, users may miss those feeds that refer to a subject by its Alternate Labels, which is one of the main reasons that prevent users from accessing certain parts of the information on the Web.

Investigating more than 350,000 records in a subject dataset that has been published by Library of Congress [22] has shown that almost 38% of subjects have at least one Alternate Label. This means that for 38% of the subjects in this dataset, there might be information that cannot be accessed by searching a single subject label. It is interesting to note that there are subjects in that subset that display more than 30 Alternate Labels, though most subjects generally have less than 10 Alternate Labels.

The fact that keyword search is not the ideal option for filtering and classifying the information by subject has been discussed by the authors of “Conceptual Classification to Improve a Web Site Content” [23]. They applied a conceptual classification of documents to the Web Usage Mining process in order to improve the process by considering the semantic of the contents of the web site, and helping the web site administrators to improve the usability of their web sites. Their proposed classification method involves defining the important concepts in their website and identifying the value of a relationship between a concept and its Alternate Labels by an expert. The value in combination by the frequency of the concept terms in a Web page is used to calculate the degree of possibility that a concept is represented in that Web page.

However, their method is applicable for a system with limited number of concepts in the information, because it requires identifying the list of main concepts in the information by an expert. Moreover, this requirement can't be satisfied in a system which deals with a large amount of information which grows/changes constantly. In our case, the web feeds may covers various unrelated concepts and making it impossible to identify the concepts of the feeds by a human.

Also, in the keyword classification method, another part of valuable information may get lost: the information that does not refer to a specific subject directly but refers to its semantically related subjects. A subject may have different kinds of relationships to other subjects. For

example, the subject “Electronic Commerce” is a general subject for “Internet banking”, “Internet marketing”, “Internet advertising”, and various other subjects [21]. On the other hand, the subject “Marketing” is semantically related to the “Selling” [24], and this represents another kind of subject relationship. In this way, news about “Internet banking” is considered to be news about a specific type of “Electronic Commerce”, and users should be able to access them. However, with keyword search, users may not find that news unless they search all of the related subjects to “Electronic Commerce”.

The fact that users are unable to access large parts of feed information has been the motivation behind introducing and implementing this conceptual searching method and leveraging it into the feed classification in this thesis.

Therefore, the Feed Analyzer provides a conceptual classification mechanism based upon a powerful subject thesaurus: Library of Congress Subject Heading [22]. The thesaurus provides a multi-parent tree structure of subjects, and in this tree, a subject may have three kinds of relationships with other subjects. As a result of the importance of subject heading thesaurus structure in this conceptual searching algorithm, the next section will provide a description for the thesaurus structure.

3.2 Library of Congress Subject Heading

Librarians have been working on organizing and defining subjects for a long time in order to catalog bibliographic data. To facilitate searching and locating bibliographic information by subject, they have defined many subject heading thesauruses. There are various subject headings available that have been generated by many experts in the field, and some of them have been published on the Web, though not all of them are available free for public consumption. Library of Congress (LC) has worked since 1898 on the subject heading (LCSH) to be used for

cataloging materials and has owned one of the most authorized subject headings in the world. Library of Congress also provides a service to download a large part of their subject heading for free [22].

Although another subject heading had been checked before selection [25], the subject heading of Library of Congress is more convenient for the project since it has more records and the hierarchy of subjects is well-defined.

The Library of Congress subject headings, much like other subject heading databases provide a hierarchy structure for subjects. Each node has some relations to other nodes, and the collection of these relations form a hierarchy structure for the entire subject heading. In order to explain this structure it is necessary to define some of the concepts in this area. The following is a brief explanation of these concepts:

- Broader Term (BT): Is a general form of a subject, represents a super class in the hierarchy [26].
- Narrower Term (NT): NTs of a subject heading are more specific subjects for the subject and represent subclasses in the hierarchy [26]. NT relation can be extracted from BT relation.
- Related Term (RT): Is a subject that is semantically related to the concept which is defined with this subject [26].
- Used For Term (UF): UF is also called “Alternate Labels,” and it includes other terms that are used to refer to the concept that is defined with this subject. For example, for subject “Electronic Commerce,” Used for Terms are E-business, E-commerce, Electronic Business, etc. [21].

Figure 3.1 shows these BT, NT, and RT relations in a tree structure where BTs are the subject node's parents. NTs are subject node's children and RTs are nodes at the same level (conceptually) with the subject node:

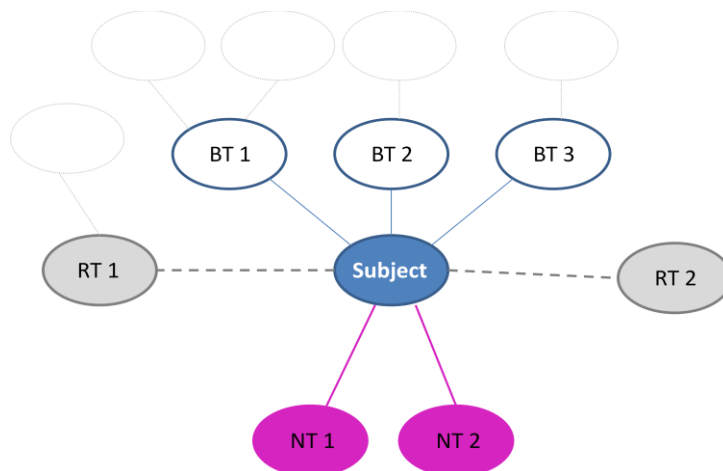


Figure 3.1: Subject Relations in the Thesaurus.

In this conceptual searching, only Related Terms and Narrower Terms are used. It is necessary to mention that the conceptual searching algorithm that is used in the feed classification has no limitation on the type of information. However, in this study it was implemented for web feeds.

3.3 Providing Efficient Access to Information through Sorting the Result

Unfortunately, despite the fact that employing a conceptual searching algorithm facilitates access to a larger part of information, the difficulty with large amounts of information arises in other levels:

- I. Inside a category: When the amount of information about a subject grows larger. For example, by bolstering cloud computing, many authors and websites publish articles, stories, and news about this new technology, as it is a popular topic of that particular day. Therefore, the amount of published information on the Web that refers to this subject increases dramatically.
- II. In category level: When the number of the subjects (categories) in the system increases as a result of the variety of subjects that are discussed in articles, news or other forms of published information on the Web.

However, from the users' points of view, the problem remains the same as before: difficulty in reading and processing large amounts of information.

To solve this problem, the Feed Analyzer follows human behavior in the similar situation. When people deal with large amounts of information and do not have sufficient time to read and process it all, they prefer to read the most important or popular ones, which is a reasonable trend.

Therefore, in order to improve the performance of search process, it is necessary to order the subjects which represents the categories in the system and feeds in a category by their value to users. This approach allows users to prioritize the most important categories and feeds. To implement this technique, it is required to calculate the value of a subject in comparison to other searched subjects and the value of a feed in a category in comparison to other feeds in that category.

When the system classifies information by subject, the most important information in a category is the one that is more relevant to the subject. Sorting information according to its

relevancy to the subject of the category helps users find the most relevant feeds. Therefore, it would be easy and fast to read and follow the most relevant information about a subject.

In terms of large list of subject categories, users need to find the most important categories. In this level many parameters may be involved in defining the value of categories from the users' points of view. Some users prefer to define the order of categories according to their interests. Although providing this feature is easy from the developer's point of view and is desired by many users, it is not used by the Feed Analyzer, because expanding the number of categories makes it difficult to organize a long list of subjects.

In addition, as was previously mentioned, implementing this method is simple but there is another good approach for ranking and ordering this list. It implements the behavior of most of the users. From most users' points of view, important news or events are those that are discussed more in the media and websites. In other words, the most important topics are those that are referenced more, and although this method is not the most ideal method, it is compatible with the general behavior of most people and rooted in the belief that the important news and stories are those that are believed to be by most people. This belief, however, is not always true and may be impacted by the views, policies, and decisions of the media, but it is used by many automatic news aggregators and generally has satisfied the users of these systems.

On the other hand, in terms of business decision making, this method has many advantages. As mentioned before, feedbacks and expectations of the customers are important for business executives. However, the common feedback and expectations of larger groups of customers have higher priority in comparison to the requests of the smaller group of customers (for example one or two customer(s)).

Sorting subject categories by the number of references to the subjects in the news feeds facilitates identifying the customers' most common requests, feedback, and trends. For example, a seller decides to adjust the prices or arrangement of his products according to the taste of a large group of customers in order to increase the profits. By this sorting method, the seller can identify the most popular areas in his work field and use them in arranging the products.

Customer preference is one of the aspects that may be realized through the aforementioned sorting methods. Subjects are sorted according to a rank value that indicates the subjects' popularity in the feeds, or more precisely, the number of references to the subjects in the web feeds.

3.4 Implementing the Solution in Feed Analyzer

In order to provide the explained features, the Feed Analyzer allows users to select a list of favorite subjects. The system then customizes the reports according to the subjects in the users' favorite subject lists. Along with those subject lists, it includes their related subjects (RTs and NTs). The subject list is sorted by a ranking value that is calculated based upon the number of references to the subjects in the web feeds. This is performed by placing the top subjects in the web feeds higher in the report, and those are the ones that are referenced in the news more than others. The ranking method will be described in more detail in Chapter 6.

For each subject in this report, different report is generated that contains the feeds related to that subject and the feeds in this report are sorted according to their relevancy to the subject. Consequently, this report affords users the ability to browse the actual article or story.

3.4.1 Classification of the Required Information

Feed Analyzer is a system that classifies web feeds according to an authorized, extensive subject thesaurus. A conceptual classification method has been designed to improve the

classification. Unfortunately, both datasets involved in the classification are huge. The count of subject headings in the system is approximately 353,307 and at the time of this writing, the feed item table now contains more than 80,000 records. Classification of more than 80,000 feeds against more than 350,000 subjects that involve performing the text search is a very exhaustive task.

Looking at the requirements of the system reveals a helpful fact: It is not necessary to classify all of the web feeds in the system. Instead, the users of the system only need a report of the feeds that reference the subjects of interest, and these subjects are stored in the users' favorite subject lists. In addition, the described conceptual classification algorithm requires performing classification for the semantically related subjects to the subjects in those lists.

It indicates that in order to decide that, a feed should be classified under a subject or not, the subject term, its Alternate Labels, RTs, and its NTs should be searched in the description field of the feed. This task is long and the performance of the system is considerably low with this method. In addition, it involves large amounts of processing and storage resources and therefore, increases the cost of the system maintenance.

Since the Feed Analyzer's responsibility is to generate reports for the feeds that have references to the subjects in the users' favorite subject lists. Other subject categories and their contents are not important for the system. This is interesting and important, because instead of creating categories for all of the subject headings (more than 300,000 subjects), the system needs to create categories for the subjects in the users' favorite subject lists. Employing this approach reduces the required processing in the system and therefore the number of required resources is decreased. This improves the performance of the system and decreases the cost of system maintenance.

3.4.2 Leveraging Other Technologies

Providing the high performance processing and classification mechanism for large numbers of feeds based upon an extensive subject heading has increased the complexity of the system. Development of a complex system is a challenging task. On the other hand, achieving the goal of the system requires performing multiple tasks. With consideration of the complexity and the heavy load of processing, the system requires multiple entities working together to reach its goal(s). To overcome the complexities in the development of the system, achieving its goal and providing high-performance flexible system, an agent-oriented software engineering has been used, resulting in a multi-agent system.

In addition, processing large amounts of information is a heavy task that requires huge resources (CPU and Memory). Therefore, expanding the amount of information leads to an increase in the number of resources that the system needs. However, failing to provide the required resources may have a negative impact upon the performance of the system. Increasing the IT resources of a system involves employing more IT professionals and increasing the cost of system maintenance. To improve the scalability of the system with the never-ending growth of information and to reduce the cost of deployment and maintenance of the system, the cloud has been selected as the deployment environment for the system. Therefore, the Feed Analyzer has been developed as a cloud project including four services. In addition, the system also has benefited from the cloud storage services for the exchanging messages among distributed entities (agents).

In the next chapter, agent-based software engineering will be explained and applied in the development of the Feed Analyzer system.

CHAPTER 4: AGENT-BASED SOFTWARE ENGINEERING

Development process of software involves many complexities and difficulties, including difficulties in analyzing the customer's requirements, managing the development process. Software engineering was introduced as a way to overcome these difficulties and complexities [27]. By increasing the complexities of the system due to advances in hardware and software technologies, the need for distributed software system which could adapt to changes in the environment pushed the software engineering methodologies toward the agent paradigm which provided more flexibility and a higher level of autonomy. In terms of autonomy, the main focus of this paradigm is on specifying the agents' goals and decision-making process [27].

Generally, an agent-based (multi-agent) system simplifies the complexity in the distributed system by breaking down the main complex problem into multiple less complex problems which are easier to solve. The system can provide useful information about the different stages of the process and can apply this information to improve system's functionality and decision making capabilities [28].

From the design point of view, applying an agent-based approach can help to increase cohesion in the agents and can decrease coupling between the agents in the system. With this technique, change in an agent should have lowest possible impact on the other agents of the system.

In the next section, some basic definitions of the agent-oriented methodology are provided to help the readers understand this software engineering method.

4.1 Multi-Agent System

Before moving forward it is necessary to understand two main concepts in the agent-oriented approach: A Multi-Agent System (MAS) and an agent.

In the simplest terms, the agent-based system or multi-agent system is a collection of the agents working together in order to achieve the overall systems goal. Jennings and Wooldridge in their article “Agent-Orient Software Engineering” [29] defines the agent as: “an agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives”. The environment of an agent is a collection of other agents or other software services and applications. Using the above definition, a multi-agent system can be defined as a software system which is composed of multiple independent agents which represent multiple activities [30]. These activities need to interact with one another and with their surrounding environment. As mentioned before, generally the agent-oriented paradigm is being utilized more in the development of complex software systems and therefore, a multi-agent system represents one or multiple complex behaviors.

Traditionally a multi-agent system paradigm was based upon a bottom-up design which concentrated more on the independency in the design of the agents. However, in a recent article [28], Kishore refers to the multi-agent system as any system that is composed of multiple autonomous agents and has the following characteristics:

- a. Each agent has a limited view of the system and doesn't have enough capabilities to solve the entire problem.
- b. The system does not have global control over the environment.
- c. The system's data is not centralized in one agent.
- d. The computations of the system's agents are asynchronous.

Since multi-agent system is composed of multiple and almost independent components without global views of the system, conflicts or deadlocks may occur. In order to remove the conflicts and deadlock coordination is necessary to make sure that all the agents work properly to satisfy the main goal of the system. One of the advantages of coordination among agents is the improvement in the system's efficiency that is achieved by sharing knowledge and information between agents. This knowledge is utilized by agents to perform their tasks or to reach their goals. Common coordination techniques in a multi-agent system are organizational structuring, exchanging metadata, multi-agent planning, negotiation, and other forms of coordination [28].

4.2 The Reasons behind choosing Agent-Based Software Engineering for Feed Analyzer

As previously mentioned, agent-oriented software engineering is used to facilitate the development process of the complex, distributed software systems which capability to adapt to the changes in their environment.

On the other hand, in relation to the definition of the Feed Analyzer system in chapter 3, this system could be considered a complex software system with characteristics compatible with an agent-oriented system. The following reasons support the choice behind agent-based software engineering as a development methodology for Feed Analyzer system.

- a. The Feed Analyzer is a complex system which is responsible for retrieving feed information from a large number of sources (websites), converting them to the system internal format, indexing the feeds in order to facilitate the searching, filtering, and analyzing processes. At the same time, it should provide a Web service to interact with the users. In addition, the system may process and analyze a large amount of information which may take a long time and may require an abundance of resources. To provide a fast

service for users with a low error rate, the system needs an optimized method for processing information.

As a result of the frequent updating of information by their sources, the Feed Analyzer should check feed sources frequently with a reasonable interval between each update. If the interval between searches is greater than actual interval between feed updates by the website, the system could miss some important information, as old feed items could be removed from the feed channel and then the system would not have access to them. The last two constraints require employing multiple searching entities when the number of feed sources in the system increase and a searching entity cannot update the feeds of all websites on time. These characteristics and constraints indicate the complexity of the Feed Analyzer.

b. According to the requirements of the system, the Feed Analyzer needs to perform multiple tasks which lead to reaching the main goal of the system. Therefore, the Feed Analyzer can be defined as a composition of the following activities, each define a specific task:

- Reading feeds from multiple websites (goal: reading feeds)
- Analyzes and classifies large amount of frequently updated information against a well-defined hierarchy of subjects (goal: classification)
- Providing a user-friendly interface with a statistical report of subjects and feeds (goal: report generation and user interface handling).

These tasks can be assigned to multiple agents to work together in order to achieve the system's main goal.

- c. Flexibility to the changes in the environment or the changes in the system's requirements is important in the Feed Analyzer system. In order to cover the large number of websites on the Web, the system should be able to adapt to any new feed broadcasting standards. With the fast improvement in the computer technologies, emerging new protocols for broadcasting information to which address problems and weaknesses of the old protocols should be expected. An alternate feed format was previously described in chapter 2, called Atom. The Atom protocol was introduced to address limitations of the RSS feed and to support the broader range of new standards.

To support various feed formats, a special mechanism should be implemented to parse information in each format. One of the main advantages of employing agent based approach to develop the Feed Analyzer is facilitating the process of adding a new entity to the system. This flexibility also includes the flexibility at the system architecture level without the necessity of changing the entire system.

For example, from the agent point of view, the system has a feed reader agent whose goal is retrieving feed information from feed sources. The reader agent may have two different types: an RSS reader agent or an Atom reader agent. With this design, supporting any new feed standard merely requires adding a new type of reader agent in order to parse information that has been encoded according to that new standard. With this change, other entities in the system would not be changed or the changes would only be slight.

- d. Developing the Feed Analyzer as a distributed system has many advantages for the maintenance process and the scalability of the system, because the system needs to perform multiple tasks and each task has a different resource requirement. For example, reading the feeds needs more network resources and less processing resources, while indexing the feeds requires more processing resources and less network resources.

In developing a distributed system, considering these requirements facilitates the maintenance and scalability of the system because with the growth of feed information, the indexer needs more processing resources, and the administrator can move this module to a system with higher processing power.

In addition, as will be described in the next chapter, cloud computing could improve the scalability of distributed services and could reduce the entire cost of the system. Therefore, by implementing the system as a distributed service, it is possible to take advantage of cloud computing.

- e. In addition to the above reasons, applying an agent-based approach to the Feed Analyzer system has resulted in designing a flexible system which can be adapted to any system with the same requirements: a software that collects information from different sources through various standards, classifies the information based on an extensive subject heading and rank the subjects according to the number of references for the subjects in the retrieved dataset.

4.3 Agent-oriented Software Engineering in Feed Analyzer System

In order to design the Feed Analyzer as a multi-agent system, a simple agent-oriented methodology which is called Gaia [31] has been used. Gaia is a top-down methodology for the

analysis and design of the agent-oriented system at the micro-level (designing the agent's structure) and macro-level (organizing the agents in the system). However, this methodology has a limitation: the agents' abilities and the relations among the agents should be static in runtime. There is another famous agent-oriented methodology (MaSE [31]) with fewer limitations, but Gaia is a convenient choice for Feed Analyzer, because the system does not require dynamic agent abilities and relations in runtime.

In Gaia, the analysis process includes identifying the roles within the system, modeling the relations between them, and defining the attributes of each role (attributes of a role are responsibilities, permissions, activities, and protocols). The Design process in this methodology includes mapping the roles to the agent types, identifying the required number of agents' instances of each type, determining a service model for implementing the agents, and the acquaintance model as a tool to show communication among agents.

Before starting the analysis process, it is necessary to identify the use cases in this system and design the use case diagram. In the next section, a use case diagram of the Feed Analyzer is displayed and a description of the use cases is provided.

4.4 Use Case Model

Figure 4.1 shows the Feed Analyzer's use case diagram. Below is a brief description of each use cases in this diagram:

- **Manage Feed Source:** May add the feed sources to the system database. A member of the system will send the add request to the system with the URL of the feed source. In the future, the system might require a mechanism to find the new feed sources and to automatically add them to the database.

- Manage Favorite Subject Lists: May add a new subject to the member's favorite subject lists or may delete a subject from the list. These requests are sent by the member role. This manager may also provide the search of the subject headings.
- Generate Feed Reports: May generate two main reports: A Subject report and feed reports. The first report may be generated based upon the member's favorite subject list.
- Manage Feed Sources: Will identify the feed sources that require a data update and will add them to the Feed Reader Queue. The System manager may be responsible for any task that has an impact on the overall behavior of the system and that cannot be assigned to other use cases as a result of the increase in the cohesion of the system.
- Read Feed: May get a feed source URL from the Feed Reader Queue and retrieve the web feed RDF file from the feed source. It may convert the feed channel information from XML to the system internal format and store the information in the database. After the successful storage of feed information, the feed reader will set the last search time of the feed source in the database into the feeds' saving time. This field will be used by the Feed Manager to identify the feed sources that require data updates.
- Index Feed: May update the system index table (_SubjectFeedRelation) when the new web feeds is added to the system. Due to the large amount of feed information and subject headings, the Feed Analyzer may need an index table. Chapter 5 will discuss in detail the indexing process and the reasons behind choosing this method.

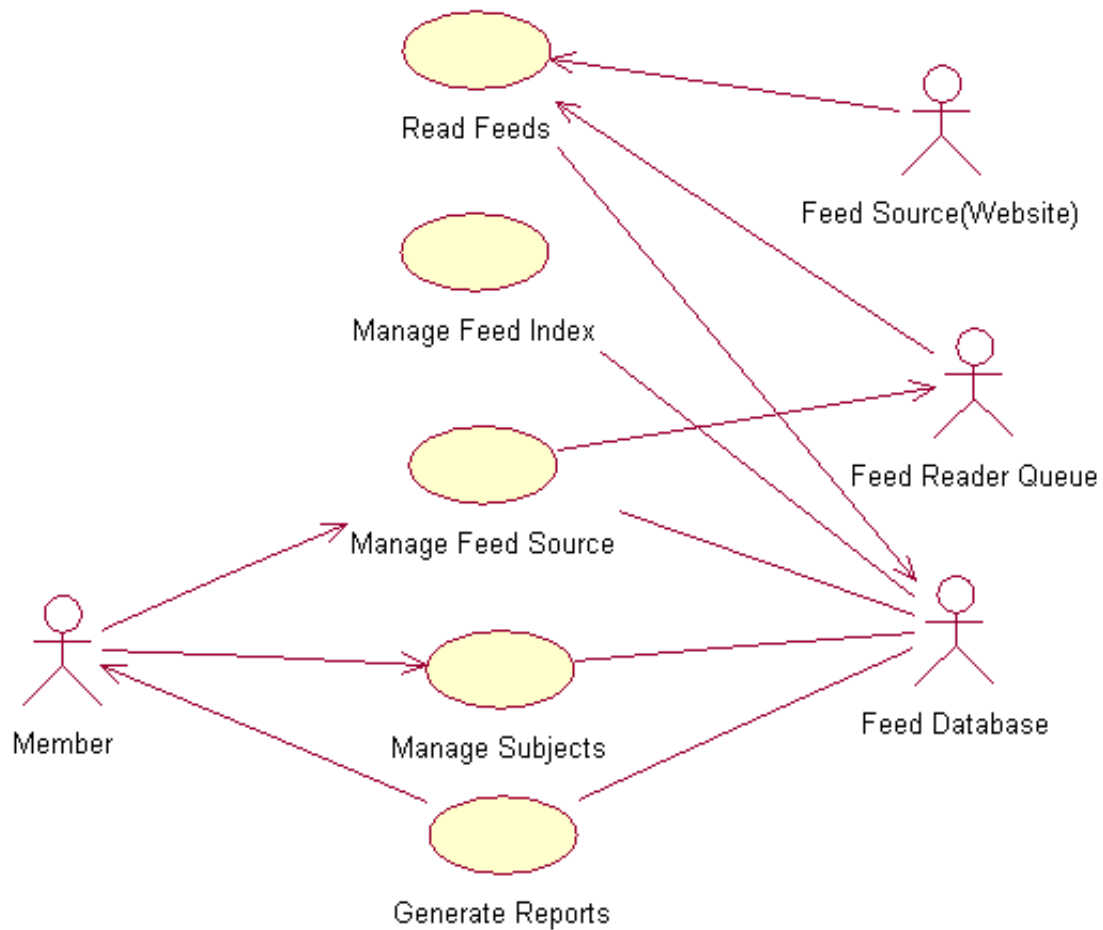


Figure 4.1: Feed Analyzer Use Case Diagram.

4.5 Designing the System

By creating the use case model, identifying the agent roles in the system will be easy, as an agent can support one or more use cases. However, we should design the agents in a way that the cohesion inside the agents was high and coupling among agents was low. The following is the list of agents in Feed Analyzer system:

- a. Manager Agent: Manages and optimizes the functionality of the system by identifying the feed sources that require data update.
- b. Feed Reader Agent: Reads RSS web feeds from the feed sources that have been selected by the Manager Agent for the data update. It also converts the feed format from XML to the system internal format in order to facilitate the indexing and classifying of the feeds. The converted feed information is saved in the database. In a Feed Analyzer system that supports various web feed standards, the Feed Reader Agent may have different subtypes (For example, the RSS Feed Reader Agent and the Atom Feed Reader Agent.).
- c. Feed Indexer Agent: Keeps the index table update when a new feed is added to the system or when a subject is removed from the user's favorite subject list. The indexing mechanism is described in more detail in the chapter 5.
- d. Feed Analyzer Service Agent: Provides a user-friendly Web interface for the members of the system. It allows users to add new feed sources to the system, search subjects in the subject heading, and modify their favorite subject list. In addition, this agent provides an analytical report of subjects in the user's favorite subject list and a report of feeds for the subjects.

In order to reduce the coupling among the agents, the agents don't have direct communication and all the communications in this system are performed through intermediate storages. Two intermediate storages are used in this system. The first one is a table (System Message table) in the database that is used as storage for any type of messages in the system. This table is designed in a way that may be used by the all of the agents of the system in order to send messages to other agents. During the debugging and evaluation of the

system, other kinds of messages are added to this storage in order to help the developer track the behavior of the system.

The second intermediate storage is a queue which is filled by the Manager Agent and is consumed by the Feed Reader Agent. It contains the references to the feed sources that require data updates.

Figure 4.2 shows the system's agents and their communications. The lines in this figure represent the communications among agents and the intermediate storages.

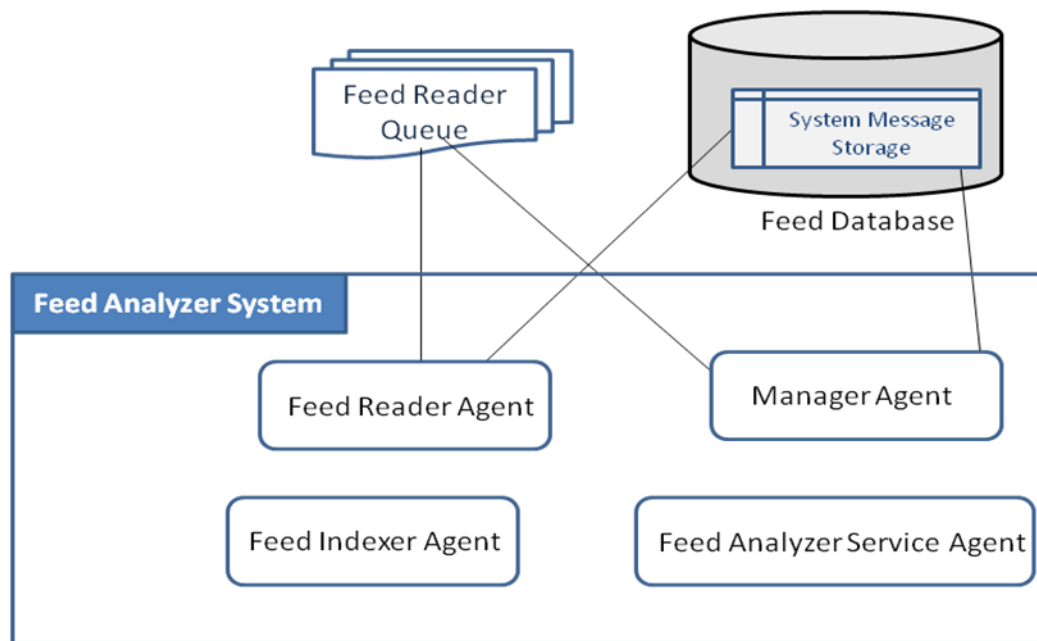


Figure 4.2: Agents and Their Communication in Feed Analyzer System.

To explain the functionality of the system from agents' point of view, a complete scenario in the system is represented in Figures 4.3.a to 4.3.e. In Figure 4.3.a, a member adds a feed source to the system. The Feed Analyzer Service Agent adds the feed source to the feed source storage in the database. In Figure 4.3.b, the Manager Agent checks the last searched time

of the feed sources in the database and since the new feed source has not been searched, the Manager adds a reference to this feed source to the Feed Reader Queue.

In Figure 4.3.c, the Feed Reader Agent reads the reference to the feed source and retrieves the feed file from the feed source. The Feed Reader Agent then processes the file and converts the feeds from XML to the system internal format. The converted file is saved in the feeds' storage in the database and the last searched time of the new feed source is then reset to the current time.

In Figure 4.3.d, the Feed Indexer checks the feeds' storage to find any new feeds which have been added to the system since the last indexing process. The feeds for the new feed source are among the new feeds (there might be other new feeds from other feed sources in the system which have been added from last indexing process). The Feed Indexer processes the feeds against subjects in the members' favorite subject lists and their related subjects. The result of this process is adding relations among a group of subjects and a subset of new feed items. It is possible that no relation was added in this step, since new feeds do not contain any subjects from the specified subject list.

In Figure 4.3.e, the member requests the feed report (in the implemented system, this step is equal to viewing the home page of the system by the member). The Feed Analyzer Service Agent generates the report by searching the index table and ranking subjects according to their occurrence in feed items.

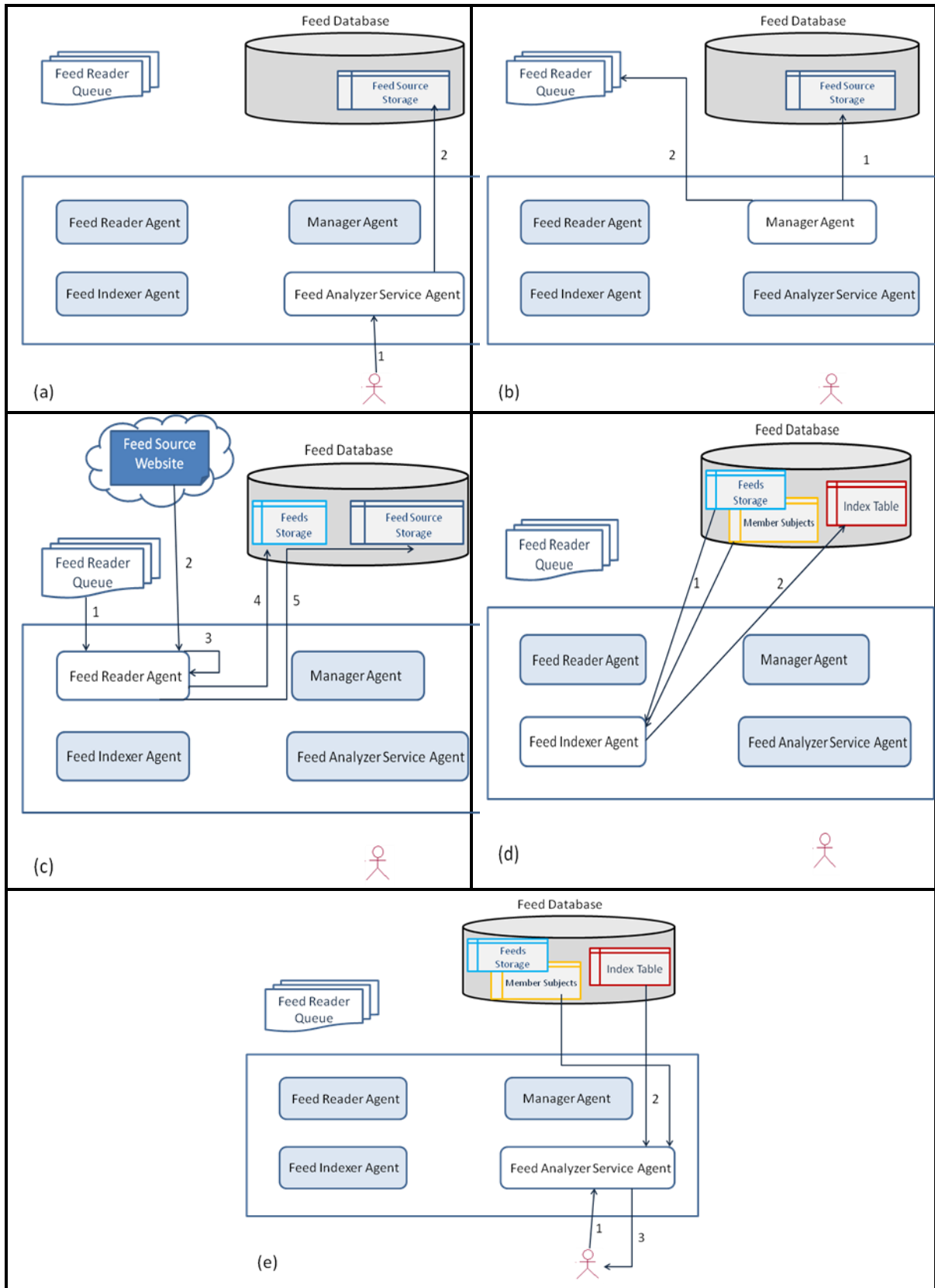


Figure 4.3: Sample Scenario in the System from Agents' Point of View.

CHAPTER 5: MULTI-AGENT SYSTEM IN THE CLOUD

Cloud computing is a new computing approach which provides scalable, virtualized resources as a group of services over the Internet [30]. It usually refers to on-demand hardware and software systems (services) that are delivered to the users. In this model, data are stored and computations are performed in a collection of datacenters that are owned and maintained by service providers or by the company (in case of the private cloud). Many companies provide cloud services including Amazon, Google, Microsoft, and IBM

The main advantages of cloud computing are scalability, availability, and cost saving [30]. In addition, customers can use storage and computing resources but they do not need to invest in nor have knowledge about infrastructure. However, it is necessary to consider the fact that migration to the cloud may not be the best choice for every system. The cloud computing model is comparable to main frame computing with two major advantages:

- i. The infinite power and resource
- ii. The ability to use the PC or other computing devices in order to connect to the cloud as opposed to the limited terminals in the mainframe model.

5.1 Advantages of the Cloud Computing

Cloud computing offers the following advantages [32]:

- a. Reduce in the cost of the system especially for new small businesses and start-up companies, because by deploying their computing activities on the cloud, these companies do not need to invest large amount of funds in the IT infrastructure and

employ highly knowledgeable IT professionals. Also, with the consideration of system's scalability in cloud computing, these businesses can start with small resources and scale up any time they need. While in traditional model, they needed to predicate resources for the future development of their system, and most likely, they have to create powerful infrastructure for future development. This structure requires spending money on the resources which may not be used until the system expands.

In addition, cloud computing has reduced the cost of deployment for the group of businesses that only need IT resources for a short amount of time. An example of this type of businesses is a small retailer who may need many network and processing resources for only the short amount of time before the New Year or only on some special occasions.

- b. On-demand and almost immediate access to the resources and the ability to customize the resources based upon the customers' needs.
- c. Allowing the companies to focus on their business instead of dealing with IT infrastructure complexities.
- d. Providing faster and easier ways for enterprises to scale up/down their services.

Providing a cost-effective, scalable, on-demand, and customizable service for IT resources, would make cloud computing an ideal option for developing and deploying software services that deal with extensive processing and large amount of data.

5.2 Cloud Computing Layers and Categories

Cloud computing is a collection of the services that form the architecture of the cloud. Borko Furht, in the first chapter of “Handbook of Cloud” [30], defines five layers (Figure 5.2) for a cloud. Following is a brief description of these layers from bottom to top [30]:

- **Data Storage as a Service (dSaaS):** provides the storage and the bandwidth to utilize that storage.
- **Virtualization:** Virtualization is partitioning the resources of a server into multiple isolated virtual machines in order to enable multitasking. To provide flexibility and availability in these computation services, virtual machines (VM) are used. Virtual machines are the software implementation of a machine that simulates the physical machines for the applications. VM separates computing and physical infrastructures and provides platform customization for the users [33]. Virtualization is one of the most important building blocks of cloud computing.
- **Infrastructure as a Service (IaaS):** Refers to the storage and computing resources as a service. IaaS provides virtualized platform, storage, and network facilities for customers while the underlying system is managed and maintained by the provider. Examples of this service are Amazon S3 storage service and EC2 computing platform [34].
- **Platform as a Service (PaaS):** Provides a software layer (Operating system and application hosting) over IaaS. Therefore, users can easily deploy their applications without the necessity of buying and managing underlying systems. An example of platform as a service is providing the IDE (Integrated Development Environment)

with data security and application hosting that can be used by developers as a way to develop and deploy their applications. Examples of this kind of services are Microsoft Azure Services Platform and Google App Engine [34].

- Software as a Service (SaaS): Refers to software services that are accessible through the cloud computing and may be used by a thin client like browsers. The remote software services that are offered through the cloud mostly belong to this layer of cloud computing model [30]. Gmail and SalesForce are two examples of the cloud software as a service [34].

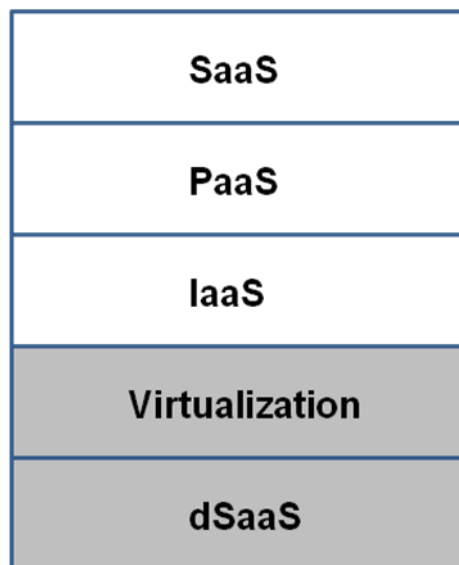


Figure 5.1: Cloud Computing Architecture.

It is important to note that cloud services are different from cloud computing. Cloud computing is a collection of technologies and IT infrastructures that enable cloud services [30]. David Chappell [35] believes that there are three categories of cloud services. His definition of these services may provide a better understanding of this technology [35]:

- **Software as a Service (SaaS):** The entire application runs in the cloud and users access this application through on-premise applications. An example of this kind of service is Salesforce.com [35].
- **Attached Services:** In this category, an on-premise application which runs on client system uses functions and services in the cloud. An example of this kind of service is Apple's iTunes [35].
- **Cloud Platform:** Customers of this service are developers who use cloud-based services in order to develop applications.

On the other hand, other researchers [33] believe that the three categories of cloud service providers are: IaaS, PaaS, and SaaS. These services are commonly known as cloud services.

5.3 Advantages of Developing Multi-Agent System as a Cloud Service

The multi-agent system is a collection of the multiple autonomous agent applications that work asynchronously in order to reach the main goal of the system. Implementing the multi-agent system as a cloud service has many advantages for developing and maintenance of the system. The reasons behind this claim are listed below:

- a. As was previously mentioned in Chapter 3, the multi-agent model is used for solving complex problems. In many cases, solving a complex problem involves heavy processing or working with a large amount of data. Therefore, the system requires extensive computing and storage resources in order to perform these processing. As a result, cloud services can be an ideal solution for developing and deploying this kind

- of multi-agent systems. This is because developers of the system can employ a large amount of network, CPU, and storage resources of the cloud without needing to worry about buying and managing hardware and IT infrastructure. In addition, cloud computing may reduce the cost of the deployment of a software system, especially for the new businesses.
- b. The multi-agent system is a distributed paradigm for designing complex systems, where multiple agents work asynchronously in a system in order to reach the system's main goal. On the other hand, an agent model is widely implemented as a service-oriented system and is considered to be successful supporter of Service-Oriented Architecture (SOA) [36]. Substantial research has been done to adapt and integrate these two approaches and to employ them in order to resolve the problems in the distributed systems. Therefore, a multi-agent system may be organized as a combination of Web services. Since service is one the fundamental concepts of the cloud, cloud computing can be used as a convenient model for development and deployment of the distributed multi-agent systems.
 - c. As was previously mentioned, one of the important parameters in designing a multi-agent system is handling communication between agents. Since failure in handling the agents' communication could cause the deadlock in the system or lead to failure in achieving the system's goal, one of the important factors in designing the multi-agent system is communication handling.

Some cloud computing platforms provide a messaging mechanism between distributed components through the cloud storage services. Microsoft and Amazon offer a queue structure for interchanging information among services in the cloud.

The queue may be used for indirect communication among the agents. Therefore, supporting a convenient way for communication between agents is another reason in choosing the cloud as infrastructure platform for the multi-agent system.

In addition, implementing the multi-agent system as a cloud service provides adds more flexibility for the system as compared with the traditional models. Employing the cloud approach allows an agent-oriented system to add or delete agents' instances with a minimum change in the system. For example, in Microsoft Azure, increasing a role's instances is accomplished by changing a configuration setting of the cloud project. This setting is used by Fabric Controller when the Azure cloud project is created in the cloud [37].

The above reasons show that cloud computing supports an agent-oriented approach. By developing a multi-agent system as a cloud service scalability in the system is increased while the deployment of the system becomes easier. Moreover, in some projects, using the cloud infrastructure for deployment of the system reduces the cost of the system.

On the other hand, development of the multi-agent system in the cloud is achieved easily by mapping each agent to a service in the cloud. Due to the variety of service types in some cloud platforms such as Microsoft Azure, the process involves selecting the service type for each agent according to the agents' requirement and the features of each type of service.

In conclusion, the advantages of developing the multi-agent system as a cloud service in increasing scalability, facilitating communication handling among agents and using the vast amount of on-demand resources, is the impetus behind the development of this Feed Analyzer as a cloud software service. Considering the types of cloud services, the multi-agent system should be developed as a cloud software as a service (SaaS). As was previously mentioned, a cloud

SaaS requires a cloud platform service (PaaS) in order to run. Therefore, in section 5.4, three famous cloud platform services will be described and compared, while in Section 5.5, the reasons behind choosing the Microsoft Azure platform for the multi-agent software will be discussed.

5.4 Brief Description of the Cloud Platform Services

Dzung Doan, in his thesis [34] “A developer's survey on different cloud platforms”, provides a comprehensive survey on the comparison of the many several typical cloud services from Amazon Web Service to Google App Engine and Microsoft Azure. Following descriptions of these services that are extracted from this survey:

- Amazon Web Service: Amazon Web Service (AWS) [38] is a Web service that provides IT infrastructure and platforms for the companies. Using cloud computing services, users can select any combination of infrastructure, storage, software platforms, and tools and simply pay for only what they use without any long-term commitment. The AWS services include Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon Simple DB, Amazon Simple Queue Service (Amazon SQS) [34].

EC2 is the main service which provides the processing capabilities. We may consider EC2 as a virtual environment which allows developers to load multiple instances of their custom applications into it and manage their applications. In order to use this service, developers need to create Amazon Machine Image (AMI), which is a package including all of the required configurations for setting up an instance (libraries, platforms, and etc) [34].

- Google App Engine: Google App Engine (GAE) [39] only provides an infrastructure for deploying Web applications. GAE supports common Web technologies along with automatic load balancing and simple managing interface. In addition, it also provides that

functionalities necessary for working with storage such as querying, sorting, and working with transactions. GAE also includes API user authentication via Google Accounts. Despite the fact that GAE local development environment facilitates development of the cloud application for developers, GAE supports both Python and Java as programming languages [34].

Google provides a simple service for developing and deploying applications. However, this simplicity has led to difficulties in deploying some more flexible server applications such as those involving backend processing data [34].

- Microsoft Windows Azure: Windows Azure [40] is the Microsoft's cloud computing environment that provides on-demand storage and compute capabilities for hosting, managing and scaling Web services and applications in the cloud. Windows Azure is the base component in the Microsoft Azure platform which is a group of services each provides a specific cloud service. Users' applications and data are run and stored on top of this platform. Other Azure platform services are Windows Azure App Fabric, SQL Azure, and Windows Azure Market Place. However, this thesis will only focus on Windows Azure.

Like Google, Microsoft provides a complete local development environment (Azure SDK) for cloud developers whether they use visual studio or other development environment. In addition to Microsoft development languages, Windows Azure SDK supports Java, Ruby, and PHP languages [34]. Windows Azure includes five components [34] including Compute, Storage, Fabric Controller, Content Delivery Network (CDN) and Connect.

The Compute component runs different types of applications in the cloud. Microsoft Azure provides a programming model for the cloud system. Each application type in Azure is called a role. A cloud system in Microsoft Azure includes multiple roles and can run

multiple instances of each role. The latest version of Windows Azure features three types of roles: Web role, worker role and VM role. Web role mostly represents Web-based applications. Inside each Web role there is a preconfigured IIS 7. Worker role runs different kinds of windows-based applications. In contrast to the Web role, the worker role code is not hosted by IIS. It is usually used for back-end processing, while the Web role is mostly used for user interaction purposes. VM role runs an image of Windows 2008 R2 and is a solution for migrating on-premise applications to the cloud.

One of the most important components in Microsoft Azure is Fabric Controller. This component manages all machines and software on them in a Microsoft data center [37]. There is a fabric agent on each machine in a datacenter and the Fabric Controller controls and manages all applications on all machines by communicating with these fabric agents [37].

Dzung Doan [34] believed that despite the fact that Microsoft Azure and Amazon Web Service (AWS) are general platforms with many similarities in the way they offer services to developers, they have considerable dissimilarities. He described AWS as the simple, independent services that all work together well. In contrast, he introduced Windows Azure as a powerful cloud operating system which facilitates developing new service.

Moreover, as he mentioned [34] AWS provided a basic and open service without any programming model while Microsoft Azure enforced users to apply its role-based model to develop their applications and to define their service configuration so the azure Fabric Controller can deploy and run their service in azure data center. However, many applications and services fit very well into this model.

Aside from these services, Amazon provides many other interesting services including support for various operating systems such as Windows 2003 and Open Solaris, Elastic Block Store [34], Amazon SimpleDB [34], and Amazon CloudFront [34].

On the other hand, Google App Engine [39], provides a Platform as a Service only for Web applications that use CGI where developers should employ a component called BigTable, developed by Google as a replacement for RDBMS [34], Despite all of the efforts of Google to offer a language similar to SQL and to provide other functionalities that developers used to have in RDBMS, this component has considerable differences with RDBMS and is dramatically unfamiliar for developers [34].

There are similarities between services of these cloud providers [34]. These similar services are listed below:

- Microsoft Table and AppEngine DataStore or Amazon Simple DB
- Microsoft Queue and Amazon SQS: both offer message queuing services that allow multiple applications to asynchronously use queue in order to exchange messages.
- Blob and Amazon S3
- Azure SDK and Google App Engine SDK

However, the author of the survey [34] believes that with the powerful foundation inherent in Microsoft Azure, it is expected that Azure becomes an important platform in the cloud computing market.

5.5 Comparison of Cloud Platforms

The comparison of the three cloud services in the previous section showed that Google App Engine is a cloud platform for developing and deploying Web service (mostly CGI-based) [34]. However, an agent-based system not only includes Web codes but it also may include windows-based code and should not be restricted to one particular technology. In addition, Google App Engine supports limited programming languages. Conversely, AWS and Azure offer cloud service for a wider range of applications.

As mentioned before, one of the key requirements of an agent-based architecture requires a reliable way to handle communications among agents which involves interchanging information with respect to the independency of the agents. Reliability in communication among distributed, unpredicted agents means that any suggested technology should be able to handle asynchronous accesses. Queue is a good option here. Microsoft Queue and Amazon SQS both provide reliable queue services for transferring information among multiple components in the cloud application. Therefore, we need to focus on these two cloud services in order to find best cloud platform for an agent-based system.

On the other hand, AWS does not enforce any programming model for applications in the cloud, while Azure developers need to use its programming model including roles and their instances. However, the Azure development model as a combination of multiple role types and their instances does not conflict with the agent-based approach. In contrast, such separation at the design level helps developers to maintain an organized cloud system from a technology point of view. Moreover, using available preconfigured IIS, developers do not have to deal with searching for a suitable template for deploying interactive Web applications (like AWI in Amazon Web Service).

One of the advantages of using Microsoft Azure for the development and deployment of agent-based systems is load balancing between roles. While in many cloud platforms developers need to handle load balancing among components of the distributed system, the Fabric Controller in Microsoft Azure provides a convenient mechanism for managing the load among multiple instances of a role. Load balancing is an important factor in designing agent-based system. Distributing load of a system among multiple instances of any role may help developers to only focus on load balancing at the conceptual level.

By using Azure load balancing, one may still need to handle load distribution in application at the conceptual level. For example, [41] discusses on distributing load in an agent-based large image processing application. In that project, the authors have recommended a model for distributing processing load of a large image among multiple agents in order to improve system's processing type by the parallel processing of different pieces of one image. This method involves many complications, and I believe that by employing Fabric Controller's load balancing, it helps developers concentrate on the more complicated parts of the program instead of dealing with distributing users' requests for instances of a Web application.

In conclusion, Microsoft Azure was selected as an ideal option for developing the Feed Analyzer agent-based system. Choosing Microsoft Azure does not mean that the Azure platform is the best choice for cloud applications. In contrast, we believe that developers should evaluate cloud services against their own system's characteristics and then select based upon their requirements. Because cloud services are new technology, there is not commonly accepted standard for these cloud services.

In addition, different cloud providers offer various services. Although it is possible to find similarities among some of them, there is much dissimilarity among them in the details. These

details should be considered when selecting a cloud platform that best suits an application. For example, although it was mentioned that GAE only offers services for specific types of applications, it is actually suitable for many applications such as specific kinds of consumer Web applications [34].

5.6 Feed Analyzer’s System Implementation in Cloud Environment

The first step in implementing the agent-oriented Feed Analyzer system in the Microsoft Azure is mapping agents to Azure cloud roles. As was previously described, Microsoft Azure has three roles: Web role, worker role, and VM role. Table 1 shows the mapping between agents of the system and Azure role types. The role name column in the table contains the names of the azure roles in the Feed Analyzer cloud project.

Agent	Azure Role Type	Role Name
Feed Analyzer Service	Web Role	User_WebRole
Manager	Worker Role	Manager_WorkRole
Feed Reader	Worker Role	FeedReader_WorkRole
Feed Indexer	Worker Role	FeedIndexer_WorkRole

Table 5.1: Agents’ Mapping to the Azure Roles.

As Table 5.1 shows, the ”Feed Analyzer Service” agent is the only agent implemented as a Web role. Other agents are implemented as worker roles. The reason behind this mapping is obvious: The Feed Analyzer Service agent is the user interface of this system and the only agent that provides Web service for users. Hence, it should be implemented as a Web application, and as was previously described, Web roles are utilized to develop Web applications in the cloud and then are deployed in IIS. Other agents do not have interaction with users and are services that perform the processing in the system. Therefore, they should be implemented as worker roles.

In the next chapter the design and implementation of these roles is described in detail. The chapter also covers the algorithms used to generate feed reports and improve the performance of the system.

CHAPTER 6: IMPLEMENTATION OF FEED ANALYZER SYSTEM

The Feed Analyzer reads web feeds (e.g. news or stories) from the feed sources then add them to system's database. In order to populate the feed database with the retrieved information, the system converts the received feed format from XML to the internal system format. The converted information is then used in conceptual classification and report generation.

This Chapter covers the implementation of the system and the challenges that have arisen during the development of the Feed Analyzer cloud system, The chapter is divided into four main sections: The process of importing the Library of Congress (LC) subject headings into the system; the four services (roles) in the Feed Analyzer cloud service; The algorithms examined for the implementation of the conceptual classification of the feeds based upon the Library of Congress subject heading; and the database design of the system.

6.1 Importing Library of Congress Subject Heading Dataset to the System

Detailed in Chapter 3, the Feed Analyzer uses a subset of the Library of Congress subject heading for conceptual classification of feeds. The Library of Congress subject heading Thesaurus is commonly used for cataloging bibliographic data. However, it also contains extensive and useful information about authorized subjects, their Alternate Labels, and their relations to other subjects. With consideration to these relations, the Library of Congress subject heading thesaurus provides a tree structure of multi-parent subject nodes that will be presented in detail.

The advantage of this thesaurus is that Library of Congress offers free downloads and access to a large part of its dataset. In utilizing this powerful structure of subjects, the Feed Analyzer provides a conceptual classification for feed items.

A windows application has been developed in order to import the Library of Congress subject heading to the system and to populate the database that is separate from the Feed Analyzer application. Since Library of Congress publishes a new version of its subject heading once a year, there is no need to include an automatic updating mechanism for subject heading, as this task may be run manually by administrator of the system.

6.1.1 Populating Subject Tables

This section will cover the process of importing the Library of Congress topical term subject heading into the system and populating the subject heading tables.

As mentioned before, the Feed Analyzer must provide a classification mechanism for multiple subjects and their related subjects. The related subjects of a subject include Narrower Terms (NT), and Related Terms (RT). In addition, all the Alternate Labels which are used to display a subject, are listed as the subject's Alternate Labels (AL). Therefore, to implement the conceptual classification method which was described in chapter 3, it is required to store the subject's definition and its AL, NTs and RTs.

In addition to these fields, many of the Library of Congress subject headings have a notes field or a Scope Note. Scope Notes or Notes are any notes or descriptions about a subject that assist the reader in more clearly understanding the subject. This Note is only used in displaying the subjects in user interface to assist users; moreover, during importing subjects from the file, Subject Codes property was added. The next section will detail subject codes, why the system needs them, and the steps that have been taken to generate them.

Before moving to the next section, it is helpful to review the conceptual design of subject entities in the Feed Analyzer system. The conceptual design is shown below in Figure 6.1. All

associations in this design are a kind of “Has” association. Therefore, we can say “A SubjectHeading Has multiple SubjectNT(s)”.

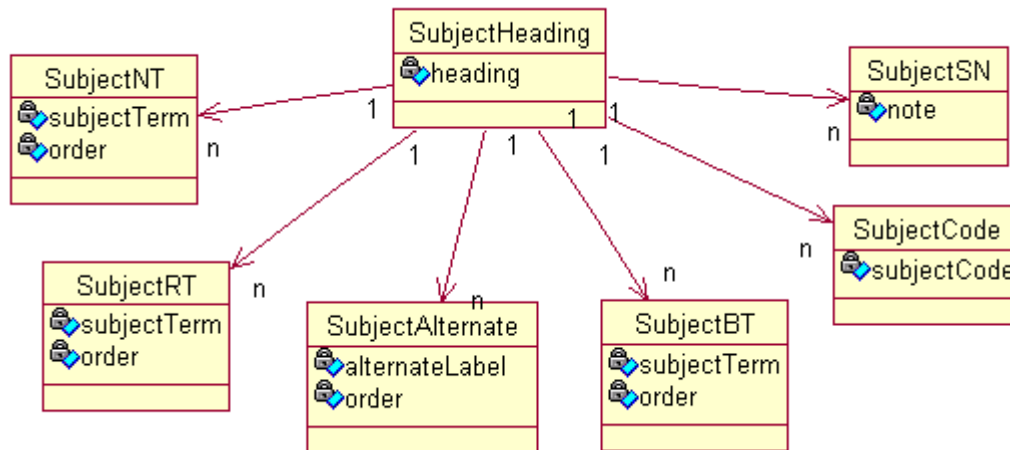


Figure 6.1: Conceptual Design of Subject Heading in the Feed Analyzer.

6.1.2 Subject Codes

To store the subjects’ hierarchy in the system and to use them in the filtering process, the information about subjects should be extracted from the Library of Congress published subject headings file and should be saved in a format that best suits the system’s main goal. Unfortunately, since relational databases do not support hierarchical structure and the SQL Server as the system’s database management system is a relational database, storing and representing the subject heading in the database is not a straightforward task. Still much research has been done in this area that has resulted in a variety of techniques and models in order to show hierarchy in SQL and other relational databases. Joe Celko, in his book [42], has explained some of these models, though not all of them are able to be adapted to the system and the subject headings used. For example, models for showing well-defined trees –such as binary trees - in

SQL may not be used in this system because subjects in the Library of Congress subject heading may have multiple parents.

One of the models that Joe Celko has explained is the Adjacency Lists Model [42]. In this model, a relation between two nodes is shown with a row including a column for the parent and one for the child. Since the Library of Congress subject headings have the information about the BTs and NTs of a node, it is easy to produce an adjacency list for each of the subjects. Actually, as will be shown later in database design, the structure of BT and NT objects and tables are completely equivalent to this model though querying data in this model in order to find descendants (extracting a sub-tree) involves joining multiple tables and therefore has poor performance, especially with large databases such as the subject heading database.

As described later in this Chapter, the imported subject heading includes more than 300,000 records. Therefore, any poor design may have negative impact on the entire search process. Another model that Celko has discussed is the Path Enumeration Model [42] where the full path from the root to each node is saved. In this model, finding descendants involves searching in the path field and not joining the table. Therefore, searching the descendants of a subject requires less process and since, defining an index on the path field in database may improve the search performance, the performance of searching in this model would be considerably better than the performance of searching in the Adjacency List Model.

One important point that approves the above selection is that the Library of Congress uses a classification number in its system. This classification number is used to produce a Call Number for library materials in the catalog process. If a subject has a classification number, the number includes its parent's classification number and a suffix to identify this node. Therefore, a node with multiple parents has multiple classification numbers and the classification number

shows the path from main root to this node. The definition of the classification number in subject headings is almost identical to the definition of path in Path Enumeration Model.

Unfortunately, not all of the Library of Congress subject headings have classification numbers and thus cannot use classification numbers in our system. The biggest drawback of Path Enumeration Model is its inefficiency in updating or deleting a node that involves updating the path of all the descendants of that node. Luckily, in Feed Analyzer system, subjects are only imported once and then they will be used for searching the feeds' information. Consequently, there are no updating or deleting tasks involved. Since using Path Enumeration Model didn't have negative impact on the performance of the Feed Analyzer, a modified version of this model was used to implement the hierarchy structure of the subject heading in the system by adding the subject codes to the system.

Subject code generation and challenges. The subject code represents a path from a root to the subject node. Since a node in the subject heading may have multiple parents, there are subjects in the heading that have multiple subject codes, each emanating from a different path from the root to this node. The Library of Congress subject heading has multiple root subjects. A root subject is a subject that does not possess a Broader Term.

In order to generate the code, the simplest option is to traverse the tree, but traversing a tree with more than 300,000 nodes is a long and arduous process and unfortunately in this experiment the process failed because the computer ran out of memory. Using the Dot Net entity model to implement traversing tree, still would result in loading a large number of subject entities into the memory and despite all of the best efforts to call the garbage collector, the application failed.

Therefore, an attempt was made with a SQL stored procedure to generate codes. Below is the `_spGenerateSubjCode` stored procedure that was used to populate the `_SubjectCodes` table. The `_SubjectCodes` table contains codes for the Library of Congress subject heading in this system. As it is shown in the stored procedure, a node code is a combination of its parent's code and the node's key. The parent's code and this node's key are separated by a dot.

```

CREATE PROCEDURE _spGenerateSubjCode
AS
BEGIN
    insert into _SubjectCodes(sc_subjectKey,sc_Code)
        select sh_Key,sh_Key
        from _SubjectHeading
        left outer join _SubjectNTs
        on sh_Key = sn_NTKey
        where sn_Key is null

    while Exists(select sh_key
                from _SubjectHeading
                where sh_key not in (select sc_subjectkey
                                    from _SubjectCodes))
    begin
        insert into _SubjectCodes(sc_subjectkey,sc_code)
            select sn_NTKey, sc_code + '.' + cast(sn_NTKey as varchar)
        from _SubjectNTs
            inner join _SubjectCodes
            on sn_SubjKey= sc_SubjectKey
            where sn_NTKey not in (select sc_subjectKey
                                    from _SubjectCodes)
    end
END

```

6.1.3 Extracting Subject Headings from Downloaded File

The published Library of Congress subject heading file came in RDF/XML format based. First a brief description of RDF will be provided and then parsing the downloaded Library of Congress subject file will be discussed.

Resource Description Framework. Resource Description Framework (RDF) is a language that is used to represent metadata about resources on the World Wide Web [43]. This metadata includes information about the author, title, modification date, licensing, and etc. In its general form, RDF is used to represent metadata about any resources that may be individually identified on the Web. Each item is defined by a unique Uniform Resource Identifier (URI and a set of properties and their values. Even properties are defined in a form of URIs. Therefore, this information may be used in multiple applications distributed on the Web and may be exchanged among them. Providing a common framework to represent metadata allows applications access information that is generated by other applications.

By using URI to identify resources and properties, RDF displays information about resources in the form of graph. RDF Premier [43] document has an example of a group of statements and RDF representation that may be used to make RDF structure clearer. Here is an example statement:

“There is a Person identified by <http://www.w3.org/People/EM/contact#me>, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr.”

Figure 6.2 shows the RDF representation of these statements [43]:

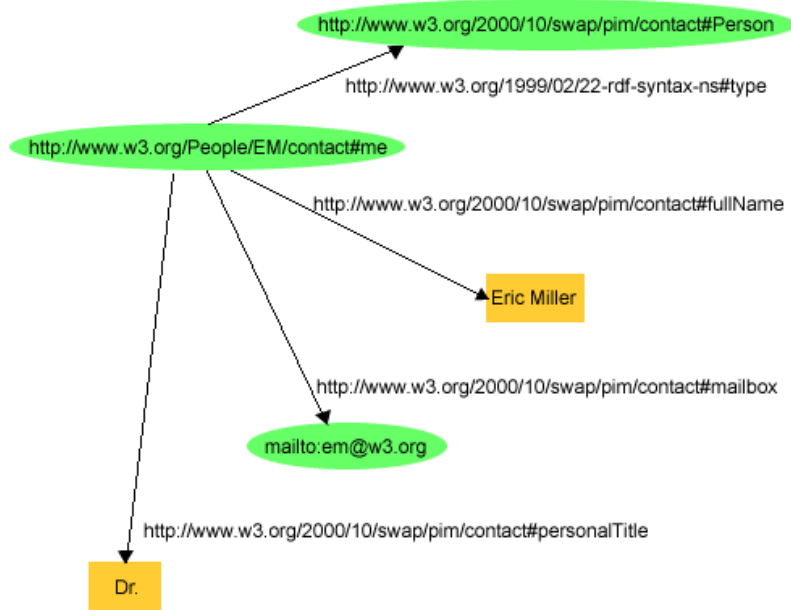


Figure 6.2: RDF Graph for Eric Miller Description, from “RDF Premier” [43].

In this graph, the URI is used to represent individuals (Eric Miller), the type of individual (Person), properties of the item that is represented by arcs, and even values of the properties (mailto:em@w3.org).

RDF information usually is displayed in the form of an XML document. The following XML document is extracted from the RDF Premier Document [43] and displays the XML representation of the above RDF example:

```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

```


</rdf:RDF>

Parsing the Library of Congress subject heading file. In order to import the Library of Congress subject headings to the system, the RDF/XML file has to be parsed. The file includes much more information than is required for the Feed Analyzer system. It includes detailed information about subjects, their URI, and their URIs in other systems along with information about subdivision forms in the Library of Congress subject headings. Yet, the only things required in Feed Analyzer are the subjects' definition and information about the subjects' relations to one another and therefore, just a subset of information already available in the file is needed in the system. The first task in importing the file to the system was to locate a tool to parse the RDF file and then convert the file to the application's subject internal format.

With the wide use of RDF in information sharing systems, there are tools and libraries available for parsing or for creating RDF files. Looking for a consistent, free library in c# has led to dotNetRDF [44], and it has been introduced as the most reliable Dot Net library for working with RDF. However, it had been in its beta version and there had been some claims about the problems that the library has previously had in parsing large files. Since there have been some claims about the functionality and performance issues in parsing the large files by this API, we decided to develop the parser for the Library of Congress subject heading file.

In addition, in this case, we deal with the problem of parsing an XML file with a specific scheme which is not a complicated task and does not require the use of an API, (in the beta version), that is designed to support many different conditions. Therefore, we decided to develop and produce the parser instead of using an available API. One of the main inspirations for writing the parser, is that the despite the power of the API, this usage is only intended for parsing

the file. The more complicated part, actually, is in the extracting of the information from the file and in converting them to the system structure for the subject headings.

Therefore, a windows application has been developed and the code has been optimized to be able to parse such a large file. The size of the downloaded Library of Congress published file was more than 400MB, and the process of parsing file and converting and inserting subject definitions and their relations to the system was not possible in one phase because the structure of the information in the RDF file and the structure of the subject entities in the system after normalization had been proven to be incompatible.

As mentioned before, the information in the RDF file are identified by the URI. To describe the relationship between two subjects, the RDF file provides the URI of the related subjects. As shown in Figure 6.3, the foreign key is used to store information about the relationship between the subjects in this system. By this implementation, to store a relation between two subjects in the database, the records corresponding to the referred subjects should be identified. But at the time of saving a subject, the other referred subjects in one of the relationships of this subject may have not been stored in the system database. Hence, in about half of the situations, the referred subjects in a relation had not been available in the database to be identified. Therefore, while parsing the file, the relationships could not be saved. This was the reason for implementing the process of parsing and converting the subject heading file in two steps.

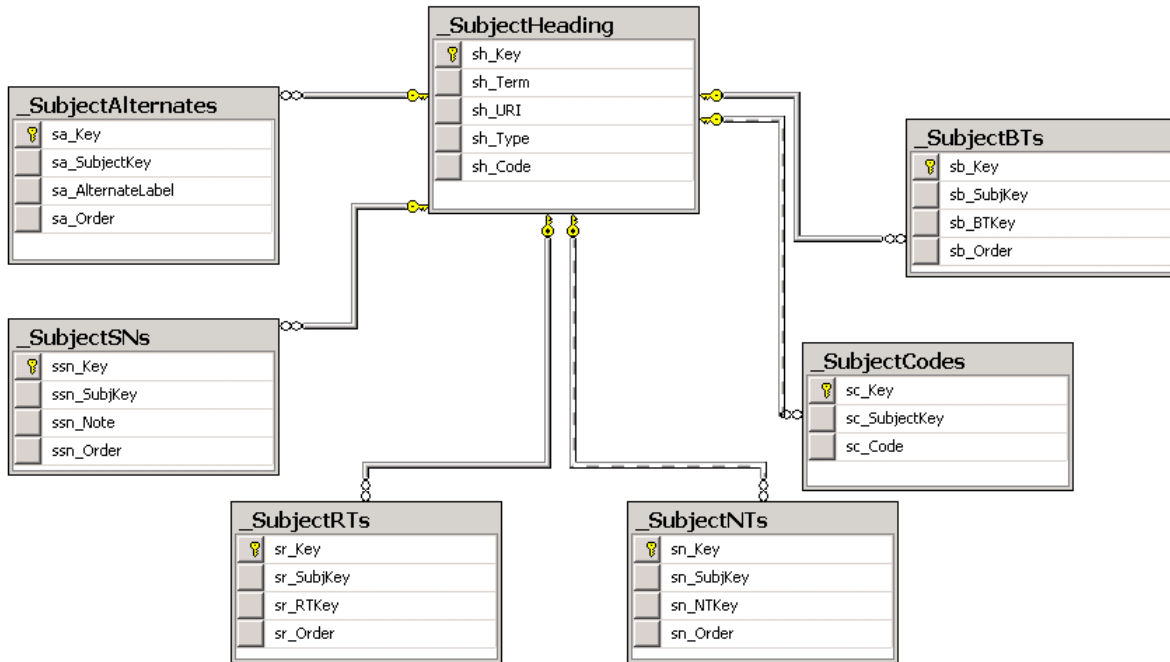


Figure 6.3: Subject Heading Tables and their Relationships after Normalization.

In the first step of parsing and converting the subject heading, the file was parsed and after extracting the required information for each subject, the information was saved in the database. During saving the relationships of a subject, if the referred subjects was not found in database, those subjects' URI were stored in a dummy table with an auto increment ID that was started at 400,000. Therefore, the ID of the subjects in this table was greater than the ID of any subjects in the subjects' main table (**_SubjectHeading**). The reason that the ID was started at 400,000 was that the number of subjects in the file was less than this value. Therefore, any foreign key with the value equal or greater than 400,000 represented a dummy foreign key.

By storing the subject's URI in the dummy table, its ID in that table was used to refer to the subject in the relation tables (NT, BT, and RT). After storing all of the subjects in the database, the dummy table was joined with **_SubjectHeading** in order to locate the subjects' real

keys. The result of this query was joined with all of the relation tables (_SubjectNT, _SubjectRT, and SubjectBT) in order to update the referred subjects' key in these tables.

After populating the system's subject heading, the Feed Analyzer's cloud services was developed. The application needed the subject heading to implement the indexing and classification in addition to the user interface functionalities.

6.2 Feed Analyzer Cloud Roles

In this section, the cloud services in Feed Analyzer are described in detail.

6.2.1 Manager Worker Role

The Manager constantly checks the last search time of the feed sources in the database. Any time the web feeds of a feed source are retrieved and stored in database, the Feed Reader sets the last search time of that feed source to the time of finishing the store process. Therefore, by checking this field, the Manager may identify those feed sources that require data updates.

The Manager adds all of these feed sources to a Cloud Queue object called the Feed Reader Queue, as the content of this queue will be used later by the Feed Reader to retrieve the web feeds. As discussed in Chapter 5, the Cloud Queue is a kind of Microsoft Azure cloud storage and is used as a queue with the rule "first come first serve." The interval between the feed source's data update is defined in the app.config file of the Manager project and may be edited independent of the system.

6.2.2 Feed Reader Worker Role

For each of the feed sources in the Feed Reader Queue, this service reads the web feeds which have been published by that feed source. As mentioned, the RSS feeds come in the form of an XML document. The Feed Reader reads the XML document, parses it, and saves the feed channels and feed items in database. To avoid saving duplicate feed channels, the Feed Reader

compares the last build date of the retrieved feed channel of the feed source with the last modified date of other feed channels in the database that corresponds to this feed source. It then discards the newly retrieved feed channel if it is already available in dataset but sets the last update time of the feed source to the current time.

The Feed Reader Queue is implemented with Microsoft Azure Queue. Microsoft Azure Queue is a robust queuing mechanism that allows asynchronous access to this storage. Applications are able to add messages to the queue in the form of `CloudQueueMessage`, retrieve messages from the queue, and delete them. In the Feed Analyzer system, the Manager adds a message to the Feed Reader Queue for any feed source that needs its data to be read.

Asynchronously, instances of the Feed Reader receive messages from the queue. Since the `CloudQueue` allows reading a message without deleting it from the queue, Feed Reader deletes the message from the queue only after finishing the process of reading and storing the web feeds. Therefore, if an instance of the Feed Reader failed because of any problem, the message will be visible to other instances after a defined amount of time (default visibility timeout).

Figure 6.4 presents the conceptual design of feed sources, the feed channels and the feed items in the system. As shown in this Figure, the Feed Source publishes the Feed Channels. The URL field in the Feed Source entity is where the corresponding website publishes its feeds. Other fields in this entity and their usage already have been discussed.

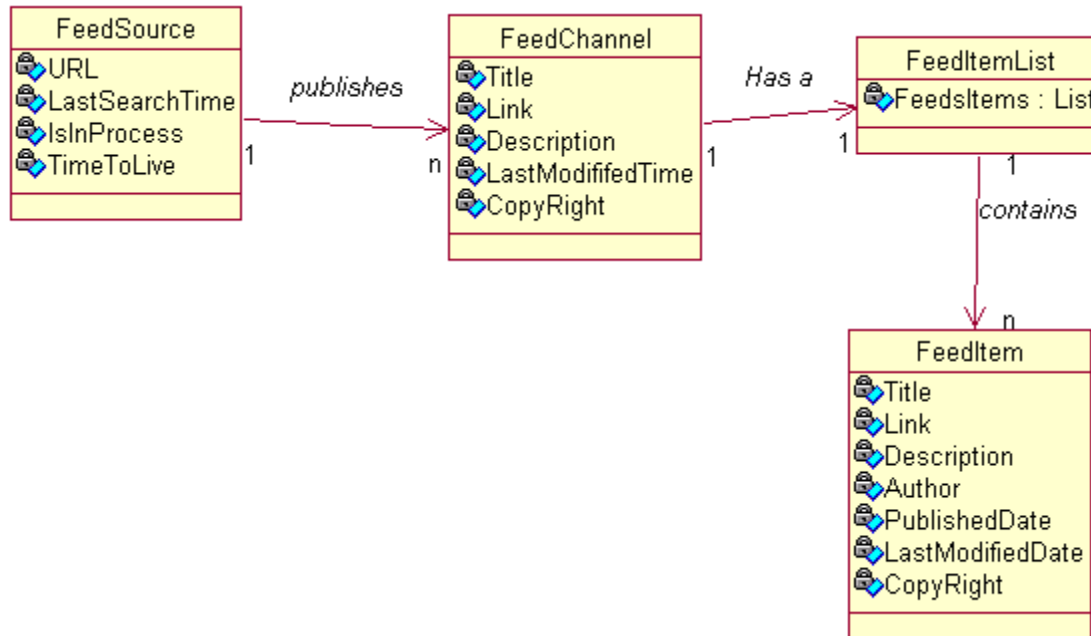


Figure 6.4: Conceptual Design of Feed Information.

As mentioned earlier, in a RSS feed XML document, under the <rss> element, there is a <channel> element. The Feed Channel entity in Figure 6.4 represents that element. The element includes only the required elements of the <channel> element in addition to the Last Modified Time and Copy Right properties. The Feed Reader uses the Last Modified Time property to avoid storing a feed channel that is already available in database. A feed channel contains one or more feed items that are shown as <item> elements in the RSS document under the <channel> tag. Therefore, in the conceptual design, a Feed Channel has a feed item list. A feed item contains the title, description, and author, etc. The most important properties of a feed item that are used later in filtering data are the Title and the Description.

6.2.3 Feed Indexer Worker Role

Processing large datasets is a time and resource consuming task and without indexing, the process could be indefinite. However, even with regular indexing, the performance of a task may

not be accepted. As was mentioned before, in order to generate feed reports, the system needs to search two large datasets: subject headings and feed items. The subject headings dataset in the system contains 353,307 records. At the time of writing this thesis the feed item table contained more than 80,115 records.

Searching on these two databases in order to generate reports has been a lengthy task. However, the process time has been improved considerably with the development of an indexing method in the system. The Feed Indexer has been developed as a part of the indexing mechanism in this system. Due to the importance of indexing in the improvement of the performance of classification in the system, the process will be explained in more detail in Section 6.3, along with the challenges in implementing the process.

6.2.4 Feed Analyzer Service Web Role

This Web role is the user interface of the Feed Analyzer system. Other services perform back-end processing. Therefore, any interaction with users is performed with this service.

The Feed Analyzer Service Web role provides the following services for the users of the system:

- Adding new feed sources to the system. Users may see a list of all available feed sources in the system and if the feed source of interest has not been added to the system yet, they could add the new feed source by providing their URL.

After adding a feed source to the system, the Manager role adds the feed source to the Feed Reader Queue and the Feed Reader retrieves the latest published feed channel from that source. Therefore, with a short delay, the web feeds of that new feed source will be available in the system.

- Searching the subject heading dataset and adding the subjects to the favorite subject lists. Users may search the complete thesaurus of the Library of Congress subject heading for a subject and add that subject to their favorite subject list. The feed report of each user is customized by this subject list. Figure 6.5 shows the New Favorite Subject page that allows users to search the subjects, view their descriptions, and add the subjects to the favorite subject lists.

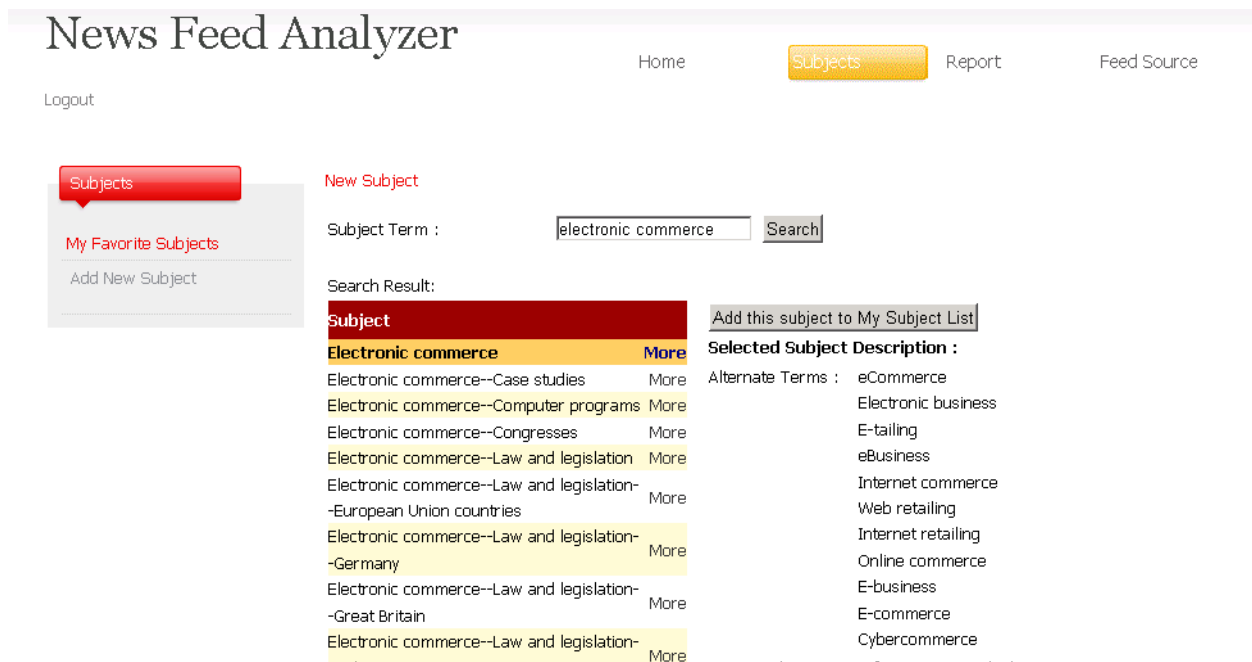


Figure 6.5: Feed Analyzer System - New Favorite Subject Page.

- Subject Report: This report provides a list of the subjects in the users' favorite subject list which have been referred by the web feeds. The report is ranked and sorted by the number of references to the subjects in the web feed dataset. The ranking method is described later in this section.

The subject report also includes the detail information for each subject in the users' favorite subject lists. By selecting a subject in the report's subject list, the detail table displays that subject and its related subjects (RTs and NTs) with their rankings. The report ignores the subjects which have not been referred in the web feeds. Figure 6.6 displays a sample subject report. In this report, the right hand table shows the detailed information of the "Cloud Computing" that is selected in the left hand table.

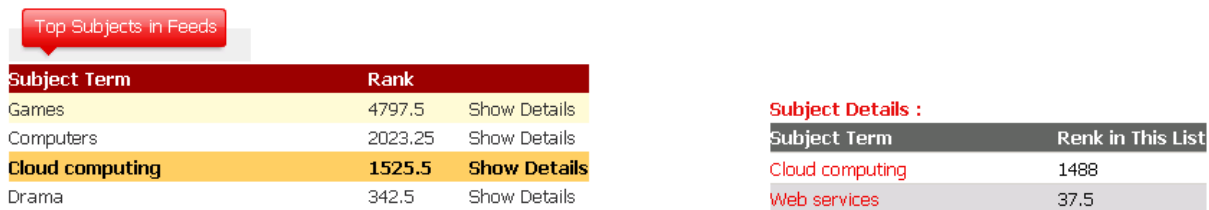


Figure 6.6: Sample Subject Report in the Feed Analyzer System.

- Feed Report: For each subject in the detail subject list of the above report, a feed report is generated by the system. The feed report includes all of the feeds that contains the subject or any Alternate Labels of that subject. The list is then sorted by the rank that CONTAINSTABLE function of Microsoft SQL Server produces. The process of generating this report will be described in detail in the next section.

The report shows the content of the feed items. The content may contain text, image, or link and the report shows them all. It also provides a link to the feed's full story, so users may browse the actual contents of the feeds.

Subject ranking method in Feed Analyzer. To sort the subject categories, Feed Analyzer produces a ranking value for all subjects in the user's favorite subject list. In the rest of this document, the subjects in this list are called main subjects. Since the system uses conceptual classification and searching, the total rank of a main subject is calculated by the following formula:

$$\text{MainSubjectRank} = \text{FeedCount}(\text{MainSubject}) + \sum_{i=0}^k \text{FeedCount}(\text{AltTerms}[i])$$

$$\text{Total Rank} = \text{MainSubjectRank} + \sum_{i=0}^n \text{RT_Rank}(\text{RTs}[i]) + \sum_{i=0}^m \text{NT_Rank}(\text{MainSubject}, \text{NTs}[i])$$

Where:

- AltTerms = Alternate Labels of the main subject
- FeedCount(subject: String) = The function returns the count of feed items which contains the input parameter.
- RT_Rank(relatedTerm: SubjectHeading) = The rank of a RT is calculated with the following formula:

$$\text{RT_Rank} = \text{FeedCount}(\text{relatedTerm.Term}) * \text{RTPower}.$$

The Term property of a Subject Heading represents the main label of the related subject. The RTPower is equal to 0.75.

- NT_Rank(MainSubject: SubjectHeading, narrowerTerm : SubjectHeading) = The rank of a NT is calculated with the following formula:

$$\text{NT_Rank} = \text{FeedCount}(\text{narrowerTerm.Term}) * \text{NTPower}(\text{narrowerTerm.level}).$$

The NTPower = .

As it is shown in the above formula, the total rank of a subject is calculated based on the number of the feeds which contains the subject term, its Alternate Label and its related subjects. However, in calculating the total rank of a main subject, it should be considered that from user's point of view, the value of the web feeds which contains the main subject is not equal to the

value of the web feeds which contain the RTs or NTs. The main subject is the user's selection and the feeds which refer to it are more valuable than the feeds which refer to a subject related to the main subject. In order to consider this fact in calculating the total rank of a main subject, the above formula introduces a value called power. The power represents the power of a feed which refer to the main subject or its NTs or RTs, in increasing the total rank of the subject. The value is defined in terms of the relationship between the subject, that is included in the web feed, and the main subject. By this consideration, the power of the related subjects on the total rank is not equal to the power of main subject on that value. In the above formula, *RTPower* and *NTPower* represent the power of the web feeds which contain the main subject's RTs and NTs in increasing the total rank of the main subject.

In conceptual searching, the main subject has the highest power because this subject is the user's main selection. Therefore, in the above formula the power of the main subject and its Alternate Labels are 1. On the other hand, Related Terms (RTs) are the closest subjects to the main subject in terms of the semantic. Their power is less than main subject but it should be more than NTs. In terms of NTs, the power is decreased by increasing their distance from main subject. Because a Narrower Term (NT) might have other parents than the main subject and the impact of those parents in the meaning of the NT decreases the semantic relation of the NT with the main subject. We propose the constant value of 0.75 for *RTPower*. But the *NTPower* is defined as a function of the distance between the main subject and the NT node. Therefore, the power of the direct NTs of the main subject is equal to 0.5 and it will be reduced by the increasing the distance of the NTs from main subject.

Since performance of searching has a great impact on the performance of the system, in the next section the Feed Analyzer's approaches in indexing and searching the web feed is discussed in detail.

6.3 Indexing and Searching Web Feeds

As mentioned in Chapter 3, the Feed Analyzer is a system that provides conceptual searching of the web feeds. There are many different ways to filter a group of information. However, one of the best ways of filtering information is through conceptual filtering. Search Engines mostly provide searching a keyword or a different grammatical form of that keyword. Yet, by performing a keyword search, users may not access a large part of information that may be needed.

Filtering web feeds involves searching the content of the feeds. In Chapter 2 the structure of a RSS feed was described. According to the description, an RSS feed contains a brief summary of the main content in its "description" field. In order to filter web feeds by their subjects, the description field should be searched using the searching algorithm detailed in Chapter 2. Here, the focus is on the technologies used to provide a high performance and an efficient filtering mechanism.

The description of the feed items is saved in a column of type *nvarchar* in the feed items table which is called *_FeedItem*. In order to increase the performance of searching in a column in a database table, database management systems use indexes. Index in the database is like index in books and helps the Database Management System (DBMS) locate data in a table without the necessity of scanning the entire table [45].

However, normal index has some limitations. For example, SQL Server index may only be defined in fields with a maximum size of 900 bytes and therefore, the maximum size of an

nvarchar field that may have index would be 450 (each character in a *nvarchar* field is two bytes). In the case of text fields (or character fields), performance of searching on a field with a normal index is not always ideal. Normal index is only able to assist with character pattern searches. Otherwise the DBMS will scan the entire table to complete the query.

Although relational database management systems are still the best option for storing and managing structured data, their performance in managing unstructured data is not good [46]. With expanding the need for managing unstructured text data such as text files or websites' content, many text-based search engines have been developed and even the RDBMS companies have worked on integrating free form text search (full-text search) into their systems.

Examples of these vendors are Microsoft SQL Server, IBM DB2, MySQL, Oracle, and PostgreSQL [46]. Among search engine APIs, the most powerful and flexible is Lucene [47] which was developed by Apache [48] in Java. The simplicity and scalability of this library in addition to its high performance in indexing and searching unstructured text data makes it popular among full text search engines and APIs [46]. Lucene tokenizes free form text and stores the extracted tokens in Lucene in nested TokenFilters [46], which are responsible for adding, deleting and modifying tokens.

Full text search engines or libraries, despite all of the differences in their implementations, provide two functionalities: indexing one or more character-based fields and searching in the index. A typical indexing process includes creating an entry for each word or root form of the word (extracted through the stemming process) along with information about the position of the word in the field. This information will be used later during the searching phase. Most indexer components have a collection of stop words that are ignored in indexing process such as 'a,' 'and,' and 'the,' etc.

In recent research [46], a systematic comparison of the retrieval performance of two relational databases (MySQL and PostgreSQL) and a search engine library (Lucene) has been done. The result reveals that despite the easy to use full text capabilities in relational databases, Lucene is more efficient in information retrieval with the lowest indexing time. The researchers [46], have concluded that using the full-text search features of these relational databases is not suitable for Web applications with high traffic.

Although Lucene has been described to be a powerful full text search API, the fact that the Feed Analyzer uses Microsoft SQL Server [49] as DBMS to manage data maintains the SQL Server full-text search as an option for this system. Employing a full text engine that is integrated to DBMS has its advantages. The most important advantage is that populating index is performed automatically by the SQL Server. This is in contrast to Lucene and other full text search APIs that need to manage and add data to the full text index file.

Therefore, it is not necessary to have a separate process for updating the full text index file because of changes in the feed data, which means using less CPU and storage for an application, although almost the same amount of storage may be used by the SQL Server to store the full text index in the database. It is not easy to calculate the cost of required compute and storage resources in Microsoft Azure, as the Azure Storage service is defined by two parameters: the amount of data and the count of storage transactions, the latter of which cannot be easily calculated. Therefore, it is not possible to provide a precise comparison of these two options in terms of their costs.

On the other hand, using a search engine API, it is required to have separate querying process for the full text index and the rest of the data in the database. Because by using the SQL Server full text search engine, querying the full text index of a table and the rest of the data of

that table may be accomplished easily with one select statement in database level, whereas with Lucene, the developer has to query Lucene and the main database separately and handles them at the program-level.

In addition, the SQL Server full text data is saved in database and it is possible to access full text index directly without using Full Text search functions with some built-in SQL Server functions. This allows developers to even query the index table to implement customized searching mechanism instead of using SQL Server full text search methods.

In conclusion, although Lucene has been introduced as a powerful full text index library, the features of the Microsoft Full Text Index, including the automatic population of data, integration with main database, and providing fast and easy full text and regular index searches, has led to the use of Microsoft Full Text Search Engine in handling the indexing of the web feeds.

6.3.1 Using SQL Server 2008 Full Text Search Engine

In the SQL Server 2008, in contrast to SQL Server 2005, SQL Server process performs full text task [50]. In addition, populating the index may be done automatically by the SQL Server, in contrast to Lucene.Net that requires the developer to manually update the index after updating the data in table. On the other hand, Lucene is a full text search library and is more expandable with a higher performance for huge data and large fields (like files). Yet, in the system, the description in the feed item is a text field that will not be as large as text files.

Therefore, as long as the SQL Server 2008 full text search does not have a negative impact on the performance of the search, it is usable. To make the full text indexing concept clearer, the structure of the full text index table will now be described. It is important to mention that the following structure is not a precise scheme in the full text index. It only includes the

information that Microsoft provides for users through executing a system function called `sys.dm_fts_index_keywords_by_document` [51]. The function will be explained more in this section.

In order to add a full-text index to the `fi_description` field in the `_FeedItem`, the following steps have been taken:

- I. A full-text catalog for FeedDB database has been created: Full text indexes are saved in full text catalog and each database may only have one catalog [52]. Therefore, the first step in creating a full-text index is creating a full-text catalog if no catalog is available in the database. The following script has been used to create a full-text catalog in FeedDB database:

```
USE FeedDB
GO
CREATE FULLTEXT CATALOG FeedFTSCatalog
WITH ACCENT_SENSITIVITY = OFF
```

- II. Create the full-text index: The Full-text index provides the ability to search in the character-based fields [53]. The following script has been used to create a full-text index on the Title and description columns of the `_FeedItem` table:

```
CREATE FULLTEXT INDEX ON _FeedItem
(fi_Title, fi_Description)
KEY INDEX PK_FeedItem
ON FeedFTSCatalog
```


The full-text index is associated with the FeedFTSCatalog. In order to have a full-text index a non-nullable, unique field needs to be introduced (usually primary key) to be used as a key index in the full-text index [52]. The keyword KEY INDEX is used to introduce the index name on that field. In this case, PK_FeedItem is the primary key of _FeedItem table and is referred to as the fi_Key field.

As mentioned earlier, populating the SQL Server full text index is accomplished in the SQL Server process and therefore, during the storing of the feed information in the database, the developer then does not have to worry about full text index. It is a considerable advantage for this full text search engine. In order to search the full text index in the SQL Server, Microsoft provides some predicates and functions for querying the full text index [52]. The predicates are CONTAINS [54], and FREETEXT [55], and the functions are CONTAINSTABLE [56] and FREETEXTTable [57].

The difference between CONTAINS and CONTAINSTABLE is that the first one is only a predicate that should be used in the WHERE clause to filter the data. On the other hand, CONTAINSTABLE returns a table of documents that are matched to the searching criteria. The table includes KEY of the full text index and the RANK of the document that is a value between 0 and 1000 and that indicates the relevancy of the rows to the searching criteria [56].

The SQL Server also provides access to the full text index table through some system functions. The sys.dm_fts_index_keywords_by_document [51], is a system function that returns document-level information about a full text index and may be used to retrieve information such as the number of times a keyword has appeared in a document or if a keyword has appeared in a document or not [51]. Although the sys.dm_fts_index_keywords_by_document returns more

information than CONTAINS and CONTAINSTABLE, using it in order to filter data is not recommended. As we will see later, employing this method to filter information is too slow.

The reason that the full text index table becomes a large table is the result of expanding data. For example when the _FeedItem table had 71811 records, the full text index table that was returned by the sys.dm_fts_index_keywords_by_document had 5,529,093 rows. Searching on this table was considerably slow and all the efforts to use information in that table in order to implement the conceptual search failed, as it will be described in the next section.

In the next section, all of the challenges that had resulted from implementing a fast conceptual searching mechanism will be described. Before delving into the methods, it is necessary to present the steps necessary for analyzing the feed information. The goal is to search the user's favorite subjects based upon the subject heading thesaurus in order to locate their occurrence in the web feeds and to rank them. Searching each subject in this algorithm involves not only searching the subject term but also searching its Related Terms and its Narrower Terms within two levels.

Therefore, the first step in this conceptual search is locating the RTs and NTs of each subject in the users' favorite subject lists. As described earlier in this Chapter, the system uses the subject codes in order to facilitate hierarchical searching in the Library of Congress subject heading thesaurus. There are 353,307 subjects in the system, and the subject codes table has 391,967 records, and since the code field in the table is a field of type *varchar*, searching in this field in a table with such a large amount of data may be slow. Therefore, an index has been defined in that field in order to improve the search.

To implement the specified conceptual searching algorithm, the subject power of each subject is needed, using the previously presented calculation method. In the rest of this

document, at times the term “root subject” for any subject in the users’ favorite subject lists will be employed in order to distinguish this subject from related subjects. By having the list of the subject headings needed to search in the feeds, the rest of the steps in analyzing the feeds based upon the users’ favorite subjects are as follows:

- Querying web feeds for any feeds whose description includes terms in the subject list found in the previous step.
- Aggregating the result by the subject term and calculating each subject’s value. The subject value is calculated differently in some of the methods defined in the next section. However, they all use subject power as previously described.
- Ignoring subjects that have not been found in the web feed dataset.
- Finally, aggregating the result by the root subject in order to generate a report of the users’ favorite subjects sorted by the calculated rank.

6.3.2 Methods of Conceptual Searching in Feed Analyzer System

In this section, the methods that have been examined to implement the conceptual searching in the Feed Analyzer are described.

Method 1. In the first method a searching algorithm which used the subjects’ occurrence count in each document was implemented. Therefore, the value of a subject is the subject’s power multiplied by the sum of the subject’s occurrence count in each document as shown in the following pseudo code:

```
For each subj in root subject’s related subject list
  rsRank = rsRank + SUM(occurrence-count) * SP
```

- *SP : Subject Power which indicates a subject's effect on the root subject's ranking.
- *SUM(occurrence-count): Returns sum of occurrence count of a subject in all documents.
- *rsRank: Root Subject's Ranking in the feed analyze report.

The only function in the SQL that provides the occurrence-count of a subject in each document is the sys.dm_fts_index_keywords_by_document. Therefore, in the first method, this function has been used to implement the searching algorithm. The following SQL query shows the method 1 query for searching one subject in the feeds conceptually:

```

CREATE FUNCTION _SearchFeedByRootSubj(
    @subjKey int,
    @subjTerm nvarchar(900),
    @levelCount int,
    @fromDate datetime)

RETURNS
    @ResTable TABLE(rootSubjKey int,
                    rootSubjTerm nvarchar(900),
                    nodevalue int)

AS
BEGIN

    declare @nodeCode nvarchar(900);
    declare @nodeCodeLevel int;
    declare @MaxChildCodeLevel int;

    set @nodeCode = (select top 1 sc_Code
                    from _SubjectCodes
                    where sc_SubjectKey=@subjKey
                    order by sc_Level);
    set @nodeCodeLevel = (select sc_Level
                        from _SubjectCodes
                        where sc_Code like @nodeCode);
    set @MaxChildCodeLevel = @nodeCodeLevel + @levelCount

    INSERT INTO @Res

    select rootKey=@subjKey, rootNode= @subjTerm ,
        nodevalue = nodePower* (titlecount*2 + discount)

```

```

from
(select display_term, nodePower = MAX(nPower),
    titlecount = COUNT(case when column_id = 3 then 1 else null end),
    descound=COUNT(case when column_id = 4 then 1 else null end)
from
(select subjKey = sh_Key, subjTerm = sh_Term, nPower
from
    (SELECT subjectKey = [sc_SubjectKey],
        nPower = POWER(0.5, sc_Level - @nodeCodeLevel)
FROM [FeedDB].[dbo].[_SubjectCodes]
    WHERE (sc_Code like @nodeCode +'.%') and
        sc_Level <= @maxChildCodeLevel
    UNION ALL
    SELECT subjectKey = [sr_RTKey], nPower = 0.75
    FROM [FeedDB].[dbo].[_SubjectRTs]
    WHERE [sr_SubjKey] = @subjKey
    ) as nodeList
inner join _SubjectHeading
on sh_Key = subjectKey
union all
select subjectKey = @subjKey, subjTerm = @subjTerm, nPower = 1
) as searchTerms
inner join
sys.dm_fts_index_keywords_By_Document(db_id(),object_id('_FeedItem'))
as idxTable
on display_term = subjTerm
inner join (select *
from _FeedItem
where fi_LastUpdateDate >= @fromDate) as feeditem
on idxTable.document_id = feeditem.fi_Key
group by display_term
) as t
order by nodevalue desc

return;
END;

```

The above function takes one minute and 38 seconds for subject 'computers' with 18 NTs and 2 RTs. Therefore, the overall number of subjects that is used in the above search is 21 for level count =1. The function is then normalized in order to improve the query time. However, normalization could not reduce the query time.

Investigating the execution plan showed the two slowest parts of the above query:

- Joining the subject list with the `sys.dm_fts_index_keywords_By_Document` function : for the subject term 'computers' with the condition previously specified this process takes 43 seconds.
- Joining the subjects with `_FeedItem` table that contains 80,000 records.

An important fact to note here is that there are many records in `_FeedItem` and in the index table that Feed Analyzer has no interest in. The Feed Analyzer system is only interested in those feed records that include the subjects in the users' favorite subject lists and their related subjects. The users' favorite subjects are stored in a table which is called `_MemberSubject`. From the database point of view, the system is only interested in subjects in the `_MemberSubject` table and their RTs and NTs (within two levels down from the root subject), while the `_FeedItem` and its full text index contain many records that do not match these conditions.

Therefore, one option is to modify this method in order to reduce the amount of information the system needs to search, as this information includes only the feeds that contain the users' favorite subjects and their related subjects. The system should maintain a copy of these feeds in a table other than the `_FeedItem` and search the full text index of that table. Since the users' favorite subject lists are changed by users, the copy of the `_FeedItem` should be updated by changing the users' favorite subject lists and by adding new feed items to the `_FeedItem` table. This update may be accomplished with table triggers or with another agent in the system that frequently updates the table.

Nevertheless, redundancy of data in this technique is high and also the performance of the system will be degraded by increasing the number of feed items and by the variety of the users'

favorite subjects. By employing this method, in the near future, the copied data may expand and the system may encounter the same problem.

In addition, there is another problem with this method when using the `sys.dm_fts_index_keywords_By_Document` function, one noticed by the author during optimizing the method. Figure 6.7 shows a part of the result set returned by this SQL function. As shown in the Figure, the result set only includes keywords found in description field of the `_FeedItem` table, but it does not include any adjacency information about the keywords. Although this information is available in the full text index, there is no way to access them.

Consequently, when using this function it is not possible to search multi-word subjects like “Cloud Computing”. As a result, even if we could solve the issue of the long query time, this second problem precludes it from its ability to generate correct reports, and so the searching method was changed.

keyword	display_term	column_id	document_id	occurrence_count
0x0063006F006D007000...	computer	4	62894	1
0x0063006F006D007000...	computer	4	62897	1
0x0063006F006D007000...	computer	4	62900	3
0x0063006F006D007000...	computer	4	63001	1
0x0063006F006D007000...	computer	4	63023	1
0x0063006F006D007000...	computer	4	63029	1
0x0063006F006D007000...	computer	4	63225	1
0x0063006F006D007000...	computer	4	63300	1
0x0063006F006D007000...	computer	4	63301	1
0x0063006F006D007000...	computer	4	63303	4
0x0063006F006D007000...	computer	4	63305	10
0x0063006F006D007000...	computer	4	63570	1

keyword	display_term	column_id	document_id	occurrence_count
0x0063006C006F00750064	cloud	3	61556	1
0x0063006C006F00750064	cloud	3	61557	1
0x0063006C006F00750064	cloud	3	61559	1
0x0063006C006F00750064	cloud	3	61560	1
0x0063006C006F00750064	cloud	3	61561	1
0x0063006C006F00750064	cloud	3	61564	2
0x0063006C006F00750064	cloud	3	61565	1
0x0063006C006F00750064	cloud	3	61815	1
0x0063006C006F00750064	cloud	3	62849	1
0x0063006C006F00750064	cloud	3	63591	1
0x0063006C006F00750064	cloud	3	63607	1
0x0063006C006F00750064	cloud	3	63608	1

Figure 6.7: Result Set of `sys.dm_fts_index_keywords_By_Document` Function.

Method 2. Since there is no other option to retrieve the occurrence count of a term in a document in SQL, a slight modification to the algorithm has been made in order to implement it. Instead of calculating the subject value by multiplying the subject power by occurrence count of the subject in a document, the value will be calculated with the following formula:

For each subj in root subject's related subject list
 $rsRank = rsRank + GetDocumentCount(subj) * SP$

where

- SP: Subject Power indicates a subject's effect on the root subject's ranking.
- GetDocumentCount(subj): Returns a count of documents that includes the subject.
- rsRank: Root Subject's Ranking in the feed analyzes report.

In the above formula, the occurrence count of each subject in each document was considered to be 1. Our investigation showed that, occurrence count for approximately 75% of the keywords in the result set of the sys.dm_fts_index_keywords_By_Document function is 1 and that half of the others have an occurrence-count of 2. Therefore, this modification does not have a considerable impact on the search result of large group of the terms in this system.

According to the above formula, in this method only a count of documents that contain a subject is needed, therefore the SQL Server full text search functions may be used here. These functions (CONTAINS and CONTAINSTABLE) allow the developer to query for those subjects that contain a term whether a single word or a multi-word term. Their performance as compared with the sys.dm_fts_index_keywords_By_Document function is considerably better.

Though using these functions have produced a better performance, conceptual searching the users' favorite subject lists with the condition defined in the Table 6.1 (under header = Method 2) takes 36 seconds, a considerable improvement in comparison to Method 1. However, this is still not an acceptable result for this system, and as a result, Method 3 has been designed in order to provide a high performance conceptual search in the web feeds.

Method 3. As described, long searching time precludes the use of Method 2 in the Feed Analyzer system. An investigation on the execution plan (Figure 6.8) of the query in that method has revealed that approximately 65% of the effort of searching one subject in the feed

information is generally spent on joining the full text index search result with the `_FeedItem` table and nearly 35% is spent on locating the table for the feed item's most recent update date. This joining is accomplished by the SQL Server when the `CONTAINS` method is used in combination with filtering other fields of the `_FeedItem` table.

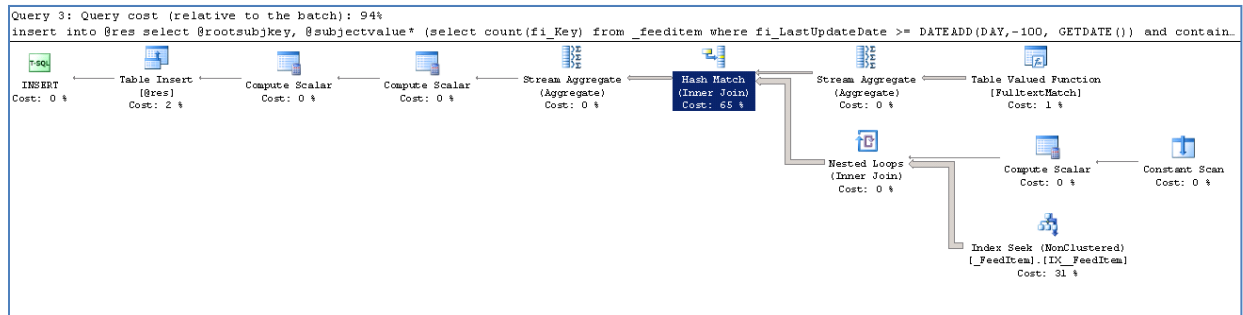


Figure 6.8: Execution Plan of Searching One Subject in Feeds with Method 2.

As shown in the SQL code of Method 1, one of the fields that are used in searching the web feed is the `fi_LastUpdateDate`. This field represents the most recently modified or build date of a feed item. Filtering with this parameter is necessary since users most likely will need to see the feed analyzing report of the web feeds in a specific period of time, and not the report of all of the available feeds in the system.

The fact that joining with a table containing large data is a long process, is obvious. The Feed Analyzer should be a system that is capable of tolerating large feed data, therefore, it should be expected to have large feeds data. On the other hand, the ability to reduce the execution time of this query should not be expected, as joining the result of the full text search with the `_FeedItem` is the most time consuming part of the query. Even using other faster full text search engine APIs may not improve this query because the problem is not in searching the

full text index but it is in joining the index search result with the feed item table. The cost of full text searching is shown in Figure 6.8, and is only 1% of the entire query.

The above conclusion has led to the development of Method 3 for the implementation of the conceptual searching algorithm described earlier in Chapter 3. Method 3 is based in Method 2, but it is based upon the fact that not all records in the `_FeedItem` table and its index are important for the Feed Analyzer system.

The system is only concerned with those feeds and indexes that include or are related to the subjects in users' favorite subject lists and the RTs and NTs of those subjects. For simplification here, the term users' favorite subject lists in the rest of this document will be used for the subjects in the `_MemberSubjects`, their RTs, and NTs (within 2 levels down the root subjects).

Returning to the main discussion regarding Method 3, the subset of feed records and their full text index, extremely valuable for this system, is much smaller than the entire feed records in the `_FeedItem`. Therefore, this system requires an additional layer for the index on the `_FeedItem` table and its full text index. This higher level index table only contains those feed records that refer to the subjects in users' favorite subject lists. Searching in that table is faster since it contains only a subset of records in the `_FeedItem` table. The fact that the users may have a common interest could improve the performance of this method.

Instead of saving a copy of the web feeds which refer to the users' favorite subject list, Method 3 uses a technique to improve performance by reduce the size of the index table. In Method 3, the process of creating the index table includes locating those feed records that contain the users' favorite subjects, creating and storing the relationship between subjects and feed records in the index table. Therefore, instead of storing a copy of the required web feeds in

the index table and search their description field during report generation process, the system searches the user's favorite subject list in the index table and find the corresponding feed items.

The result is a table called `_SubjectFeedRelation` shown below in Figure 6.9.

_SubjectFeedRelation			
	Column Name	Data Type	Allow Nulls
🔑	sfr_key	int	<input type="checkbox"/>
	sfr_SubjectKey	int	<input checked="" type="checkbox"/>
	sfr_feedItemKey	int	<input checked="" type="checkbox"/>
	sfr_feedUpdateDate	date	<input checked="" type="checkbox"/>

Figure 6.9: `_SubjectFeedRelation` Table Structure.

The following is the description of the fields in this table except `sfr_Key` that is an auto increment field and the primary key in the table.

- `Sfr_SubjectKey` is the key of a subject in the `_SubjectHeading` table (`sh_key`).
- `Sfr_FeedItemKey` is the key of the `_FeedItem` table (`fi_key`).
- `Sfr_FeedUpdateDate` is the most recent update date of the feed item and is mapped to the `fi_LastUpdateDate` in the `_FeedItem` table.

Storing the most recent update date of the feed item in the index table improves the searching performance because the application does not need to search the `_FeedItem` table nor join it to the index table in order to filter feeds by their most recent update date. This new index layer has considerably improved conceptual search in the feeds' information as shown in Table 6.1.

	Method 1	Method2	Method 3	
Root Subjects Count	1	4	4	4
Total Subjects Count (including root subjects and their RTs and NTs within specified levels)	21	162	162	447
Level count	1	1	1	2
Query Time (seconds)	98	36	1	1
Query Time per subject (seconds)	4.667	0.222	0.006	0.002

Table 6.1. Comparison of Three Searching Methods

Figure 6.10 compares the query time per subject in three method in a chart. The value for query time per subject in method 3 in the chart is 0.004 seconds which is the average of query time per subject for the two test cases of method 3 as shown in Table 6.1.

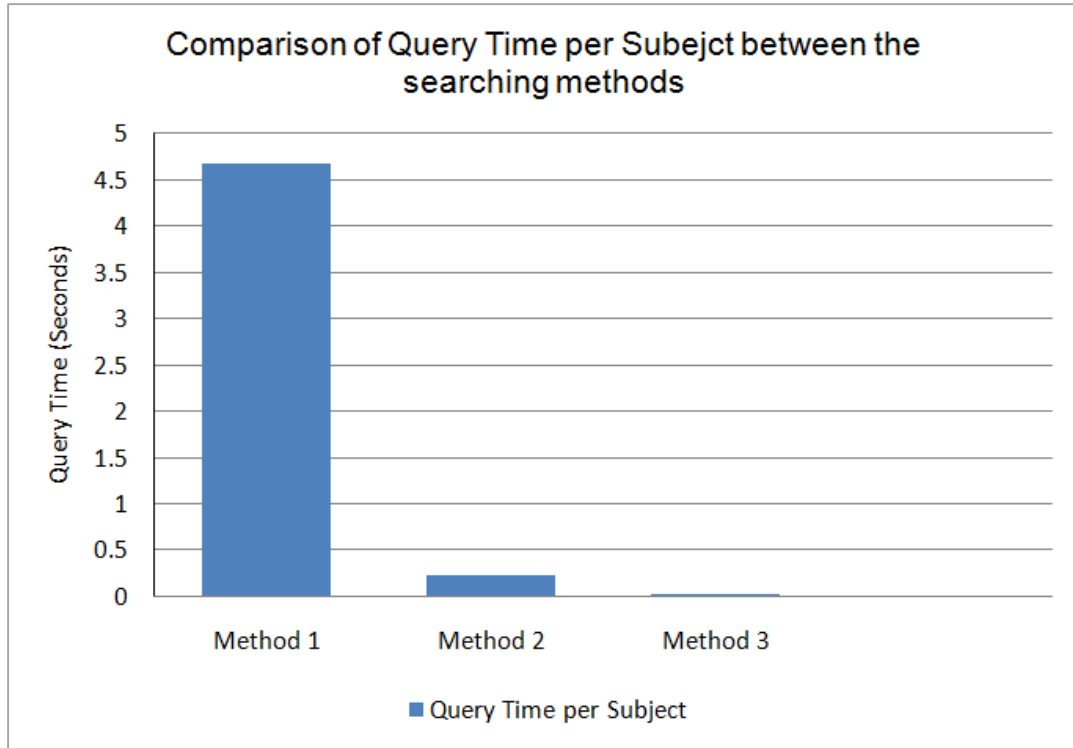


Figure 6.10: Comparison of Query Time per Subject between the Searching Method.

In order to implement Method 3, a mechanism is required to ensure that the new index table remains updated and to reduce the inconsistency between the index table and data in the `_MemberSubjects` and the `_FeedItem` table. The ultimate goal of any index table is to be consistent with the original data or maintain inconsistency with the original data at minimum. In order to implement the process of updating the `_SubjectFeedRelation` table, it is necessary to identify when the index table needs to be updated.

As defined earlier, the index table should include relations among all of the subjects in the users' favorite subject lists and the related feed records in the `_FeedItem` table. Therefore, any changes to the `_MemberSubject` table or the `_FeedItem` table should be reflected in the index table. Of course there is another factor that impacts the `_SubjectFeedRelation` table and that is the changes to the subject heading table or the relations among the subjects. Yet, since subject

headings are to be considered in the system to be a constant parameter, this factor does not need to be considered.

On the other hand, the task of updating the index table is a stand alone process that is not related to any other tasks in the Feed Analyzer system, and it can be divided into the four tasks below:

- I. Updating the index table when the `_MemberSubject` table has been changed as a result of adding a subject from the users' favorite subject lists.
- II. Updating the index table when the `_MemberSubject` table has been changed as a result of deleting a subject from the user's favorite subject lists.
- III. Updating the index when the `_FeedItem` table has been changed as a result of adding new feed items.
- IV. Updating the index when the `_FeedItem` table has been changed as a result of deleting feed items.

From the system's domain model point of view, these tasks are not related to the responsibilities of other agents in the system. Updating the index table is more a data level task than a processing task and the best place for them is in database triggers.

Therefore, the insert and delete triggers of the `_MemberSubject` and the `_FeedItem` tables may update the `_SubjectFeedRelations`. However, updating the index table involves querying the feed items and subject headings dataset and it would take a long time. Testing the system has shown that adding a subject like "computers" with its RTs and NTs (within two levels) requires searching 204 subjects in the feed index, and this task takes one second. Therefore, putting it in

the insert trigger of the table `_MemberSubjects` increase the update time of the users' favorite subject list. On the other hand, updating the index table whenever a new feed item is added to the system increases the insert time of the `_FeedItem` table. The following is one of the conditions that has been tested for evaluating the index update when new feeds are added to the system:

- Total Subjects count (users' favorite subjects plus their RTs and NTs within 2 levels) = 441
- Added Feed Items count = 100
- Update time = 5 seconds

Update time grows when subjects in the users' favorite subject lists increase. Therefore, it would not be reasonable to put this query in the insert trigger because the best choice is adding another agent to the system that is responsible for updating the index when adding new feed items.

This agent has been implemented as the worker role in the cloud application since indexing is a backend processing task. The task, called the `FeedIndexer`, searches for any new feed item every five minutes and then adds relations between these new feed items and the subjects available in the users' favorite subject lists to the `_SubjectRelationTable`. Selecting five minutes as an interval between the index updating process means that the index table is at most five minutes behind the changes in the feed item table.

Reducing the interval in order to decrease the differences among the feed items' real data and the index table has a negative impact on the whole system performance. Updating the index table involves running a query that imposes a load on the DBMS. Performing this task whenever a web feed is added to the system costs a lot for the system. On the other hand, Although many websites modify content in longer intervals than 5 minutes, the interval should not be large. Because this interval is a parameter that defines the system's latency. Increasing the latency in

the system leads to decreasing the validity of the system's reports. Because there might be many web feeds in the system which have not been classified for a long time.

For the case of updating the index as a result of deleting feed items, is easier than updating the index as a result of deleting subjects from the users' favorite subject lists. Because in order to implement the index table update due to deleting a web feed, we set the cascade delete flag of the relationship among the `_SubjectFeedRelation` table and the `_FeedItem` table. Therefore, deleting a feed item from the `_FeedItem` table leads to the deletion of corresponding record(s) in the `_SubjectFeedRelation`.

Unfortunately, updating an index when the subject is deleted from the users' favorite subjects is not that simple. By deleting a subject from the list, not only the root subject but also its RTs and NTs (within two levels) should be deleted from index table as well. However, this is not exactly true, as sometimes a subject is related to multiple subjects in the users' favorite subject lists, whether as an RT or an NT of multiple root subjects. Therefore, the corresponding subject record in the `_SubjectFeedRelation` should not be deleted without first considering its relationship to other root subjects. As a result, a subject can be deleted from index table if and only if it was not in the users' favorite subject lists (including all root subjects, their RTs, and their NTs within two levels).

This task can be done in delete trigger of `_MemberSubjects` table but the process of updating the index is a costly task in the system. Losing the relations of multiple subjects means losing CPU time and memory that has been used by the system during the index update process corresponding to these subjects. As it was said before, deleting a subject from the users' favorite subject lists involves deleting the root subjects, its RTs, and its NTs. The count of these subjects may be extensive. For example the list for a subject such as "Games" contains 372 records.

However, deleting these subjects when they are no longer needed may help the system avoid spending invaluable resources for these subjects, but if a user deletes a subject by mistake despite the system's prompt messages or if another user adds this subject in the near future, maintaining the these subjects' relationships actually may improve the overall performance of the system. Therefore, it has been decided to update the index due to the deletion of a subject from the users' favorite subject lists in the feed indexer worker role in order to perform this deletion once a day.

By this implementation, if any user (even the user who deleted the subject the previous time) adds the subject within 24 hours of its last deletion from the users' favorite subject lists, the system does not have to perform the indexing for the subject and its related subjects again. The implementation can improve the performance of the system, though it requires performing the indexing process for new feeds against these subjects. Yet, when the size of the `_FeedItem` table is large, this process can improve the performance of the system by avoiding to perform the indexing process against the whole `_feedItem` table for these subjects. In the next section, the Method 3 implementation details will be discussed.

6.3.3 Conceptual Searching Implementation

The required steps to generate the feed analysis report for users have been previously detailed in this Chapter. In the following section the implementation details of each of the Method 3 steps will be discussed:

- a. Querying subject headings for the users' favorite subjects, their RTs and their NTs within two levels and calculating each subject's power in the search result of its root subject:

The following SQL Function has been implemented to handle the requirements of this step. The function receives a member key that is the current user's key in the

_Members table, and that returns all of the subjects in that users' favorite subject lists with their RTs and NTs (within the specified levels under the root subjects). Although the level count is constant in the whole system and is equal to 2, the entire system (C# code or SQL code) has been implemented with consideration of the variable level count.

```

CREATE FUNCTION [dbo].[_fGetMemSubjectsByMemberId]
(
    @MemKey int,
    @levelCount int
)
RETURNS
    @ResTable TABLE (rootSubjKey int,
                      subjectKey int,
                      subjectpower float)
AS
BEGIN
    DECLARE @SubjKey int;
    declare @nodeCode nvarchar(900);
    declare @nodeCodeLevel int;
    declare @MaxChildCodeLevel int;

    DECLARE memSubjCur1 CURSOR
    FOR select ms_SubjKey from _memberssubjects where ms_member = @MemKey;

    OPEN memSubjCur1;

    FETCH NEXT FROM memSubjCur1 INTO @SubjKey;
    WHILE @@FETCH_STATUS = 0
    BEGIN
        set @nodeCode = (select top 1 sc_Code
                        from _SubjectCodes
                        where sc_SubjectKey=@subjKey
                        order by sc_Level);
        set @nodeCodeLevel = (select sc_Level
                             from _SubjectCodes
                             where sc_Code like @nodeCode);
        set @MaxChildCodeLevel = @nodeCodeLevel + @levelCount

        INSERT INTO @ResTable
            select @subjKey, subjectKey, subjectpower
            from

```

```

(SELECT subjectKey = [sc_SubjectKey],
      subjectpower = POWER(0.50, sc_Level - @nodeCodeLevel)
FROM [FeedDB].[dbo].[_SubjectCodes]
WHERE (sc_Code like @nodeCode +'.%') and
      sc_Level <= @maxChildCodeLevel
UNION ALL
SELECT subjectKey = [sr_RTKey], subjectpower = 0.75
FROM [FeedDB].[dbo].[_SubjectRTs]
where [sr_SubjKey] = @subjKey
UNION ALL
select subjectKey = @subjKey, subjectpower = 1
) as nodeList

  FETCH NEXT FROM memSubjCur1 INTO @SubjKey;
END

CLOSE memSubjCur1;
DEALLOCATE memSubjCur1;

return
END

```

As shown in the above SQL code, the result of this function not only includes the subjects' key and term but also includes two other integer values: *rootSubjKey* and *subjectPower*. The root subject key is returned in the result set in order to be used later in aggregating the final result by root subject.

The subject report represents a master-detail aggregate report. In the master level it only displays the root subjects with their ranking with regard to their popularity in the feed items. At the detail level, for each root subject, the report represents subject rankings with the same order as the master. As stated before, the subject power indicates the impact of the subjects on the final ranking of their corresponding root subject according to the conceptual searching algorithm:

- a. Querying feeds data for any feeds whose description includes terms in the subject list found in the previous step.
- b. Aggregating the result by subject term and the calculation of each subject's value. The Subject value is calculated by multiplying the subject power calculated in step 1 to the number of feed items including the subject.
- c. Ignoring subjects not found in feeds or that have the subject value = 0.
- d. Aggregating the result by root subject in order to generate the report.

With regard to Method 3, the step 2 only requires searching the subject keys in the `_SubjectFeedRelation` table. Other steps (3-5) impose more filtering and aggregating of the final result. Therefore, the following SQL function is used to implement the feed analyzing report for a user:

```

CREATE FUNCTION [dbo].[_fGetSubjAggregsteReportByMemId]
(
    @MemKey int,
    @levelCount int,
    @FromDate date,
    @Todate date
)
RETURNS
    @Res TABLE (SubjectKey int,
                SubjectTerm nvarchar(800),
                TotalValue float)
AS
begin
    insert into @Res
        select SubjectKey=rootsubjkey, SubjectTerm=sh_Term, TotalValue
        from
            (select rootsubjkey, totalvalue = sum(subjectvalue)
             from _fGetMemSubjectsByMemberId(@MemKey,@levelCount)
             inner join (select *
                        from _subjectfeedrelation

```

```

                                where sfr_feedupdatedate >= @FromDate
                                and sfr_feedupdatedate <= @ToDate
                                ) as sf
                                on subjectkey = sfr_subjectkey
                                group by rootsubjkey
) as subj
inner join _SubjectHeading
on sh_Key = rootsubjKey
order by TotalValue desc;

return;
end

```

The above function generates the report for those feed items whose most recent update date is within @FromDate and @ToDate inclusively.

The generated report for one of the system users is represented in Figure 6.11. The left hand table displays the user's root subjects (subjects that have been selected by the user). The right hand list displays the details of the subject "Computers which is selected in the left hand list. Although "Computers" has more than 203 related subjects (RTs and NTs within two levels under Computers), the right hand list only shows four of them, indicating that the rest of the data has not been found in the description of the feed items. In other words, those subjects had not been discussed in the feeds of the available websites.

Figure 6.11 shows a report for the same user with the details of the "cloud computing" subject. Both lists only include those subjects from the user's favorite subject lists that have been referred in the feeds, and both are ordered by the popularity of subjects that is evaluated by the rank value. Again, the rank is calculated by multiplying the subject power by the number of the web feeds that refer to this subject.

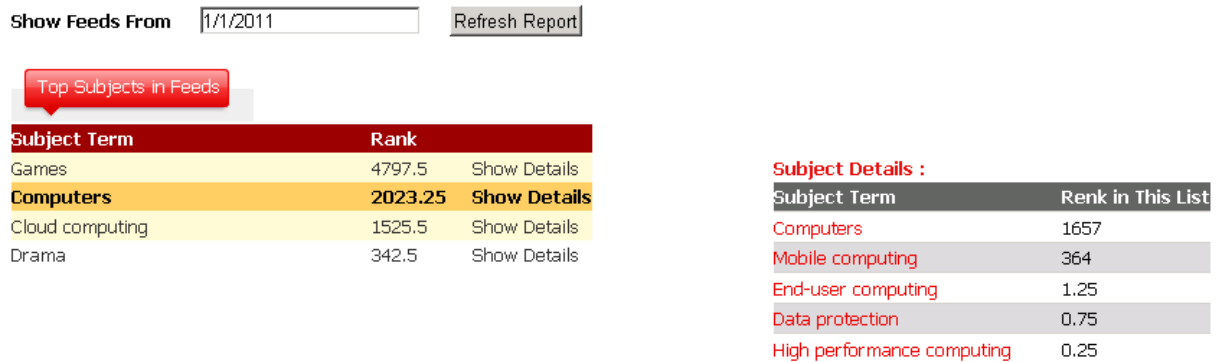


Figure 6.11: Sample Subject Report.

6.4 Database Diagram

Figure 6.12 shows the database diagram of the Feed Analyzer system. Most parts of the diagram have already been covered. The two new tables in this diagram are the `_Members` that contain the information of the system users and the `_MemberSubject` that is the storage of the relationship among the members and their favorite subjects.

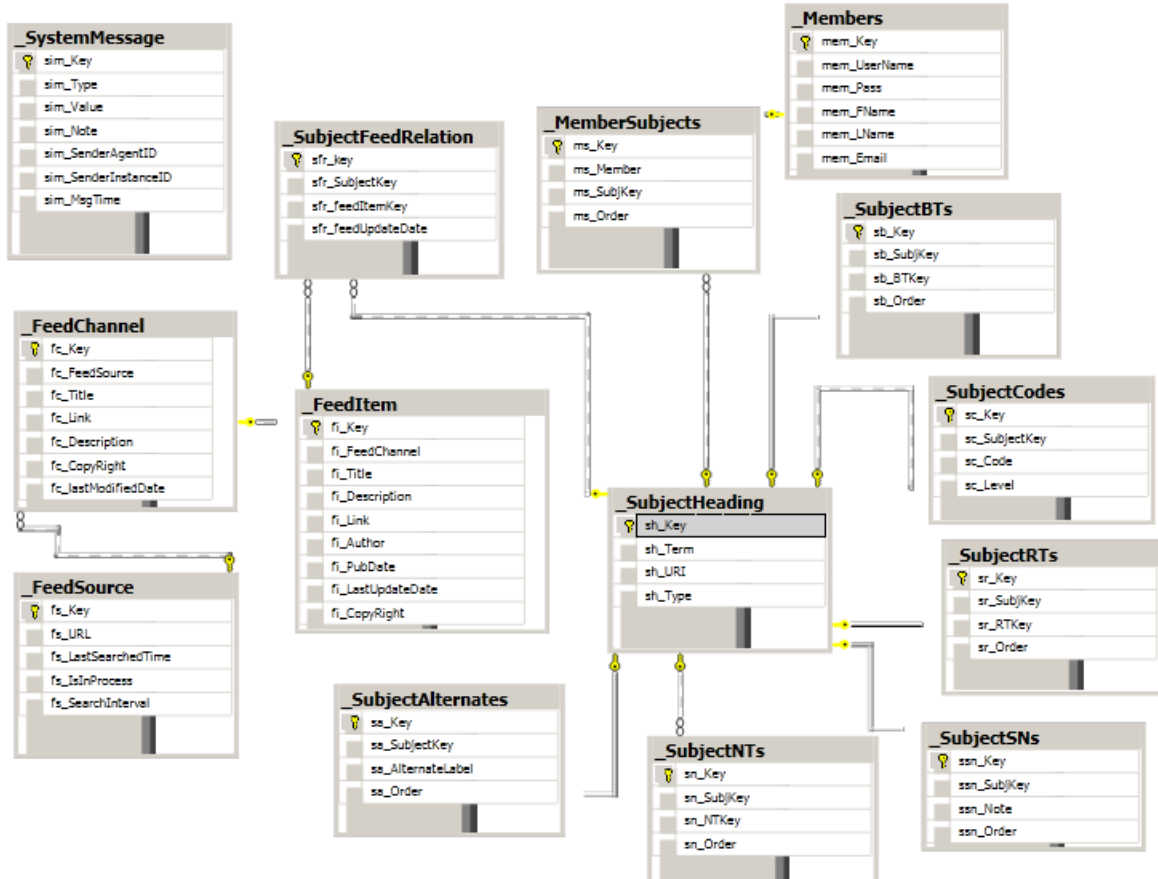


Figure 6.12: Feed Analyzer's Database Diagram.

CHAPTER 7: THE RESULTS

As a result of developing the Feed Analyzer system, the impact of using conceptual classification and sorting to provide an efficient access to large amounts of information, has been examined. The system used a large and powerful subject thesaurus to perform the classification. Since the web feeds are published frequently, the feed dataset grows rapidly. Therefore, the system needed to process two large datasets: subject dataset and web feed dataset.

At the time of this writing, the state of the data in the system was as follows:

- There are almost 100 feed sources in the system.
- The system has downloaded 79949 feed channels and 80115 feed items.
- The subject heading table contains 353,307 subject heading records, 235,150 NT (relationship between a subject and its Narrower Term) relations and 14,402 RT (relationship between a subject and its Related Term) relations.
- The index table has 18,245 records.

In this Chapter, the proposed conceptual searching method and the single keyword searching method are compared. In addition, the ranking and sorting methods in the subject and feed reports are evaluated. Moreover, the efficiency of this system for accessing web feeds by subject will be compared to the efficiency of some other feed readers and aggregators. The remainder of the chapter includes conclusion and future work.

7.1 Comparing the Result of Conceptual Searching with Keyword Searching

As it was mentioned in the previous chapters, searching was one the main functions of conceptual classification. In this section, the number of feed records that may be accessed with

conceptual searching and keyword searching will be compared. To perform this comparison, a group of 78 subjects have been randomly selected from those possess at least one Alternate Label, RT, or NT. These subjects comprise nearly 50% of the subjects in our thesaurus.

For the other half of the subjects in the thesaurus, conceptual search of the subjects is identical to that of performing a keyword search of the root subject, because they don't have any Alternate Label, NT or RT to be searched.

On the other hand, in this evaluation those subjects for whose search results (in both methods) that wind up empty are ignored. Although an empty search result indicates that the searching methods are unable to locate any web feeds related to the subject, it does not signify any deficiency nor advantage in any of these methods, because no tool currently exists to find other possible web feeds that would refer that subject. With the available search methods (including keyword search and proposed conceptual search), the empty search result indicates that there is no web feed in the database that refers to the subject or its related subjects.

From the implementation point of view, the difference between keyword search and conceptual search is that in conceptual search, in addition to the root subject term, its Alternate Labels, Related Terms, and its Narrower Terms (within two levels) have been searched. From this point of view, the keyword search is searching the subject. Therefore, in this section in order to evaluate the proposed method, the result of keyword search (represents as subject only search) and conceptual search in terms of the number of found web feeds are compared. Moreover, in order to show the impact of searching Alternate Labels, RTs, and NTs on the search result, the result of searching the subject (represents the keyword search) is compared to the results of searching subject and Alternate Labels, subject and RTs, and subject and NTs. All the comparisons are performed in terms of the number of found web feeds in the system's feed

dataset. It is important to note that in this evaluation searching a subject term involves searching the term in the web feed table (_FeedItem) with Microsoft CONTAINS method. The charts in figures 7.1 to 7.4 represent these comparisons for the first 20 subjects in the test set. Since the test set is extensive, the complete information and detailed charts are put in the Appendix B.

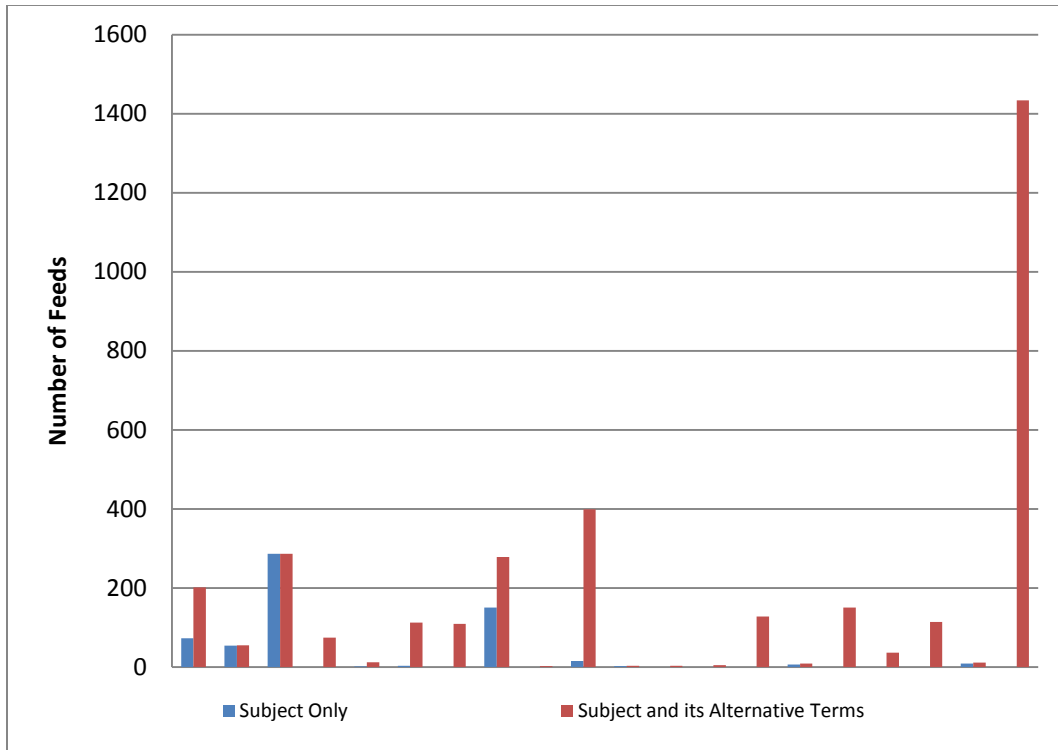


Figure 7.1: Comparison of Search Result (Subject with and without Alternate Labels).

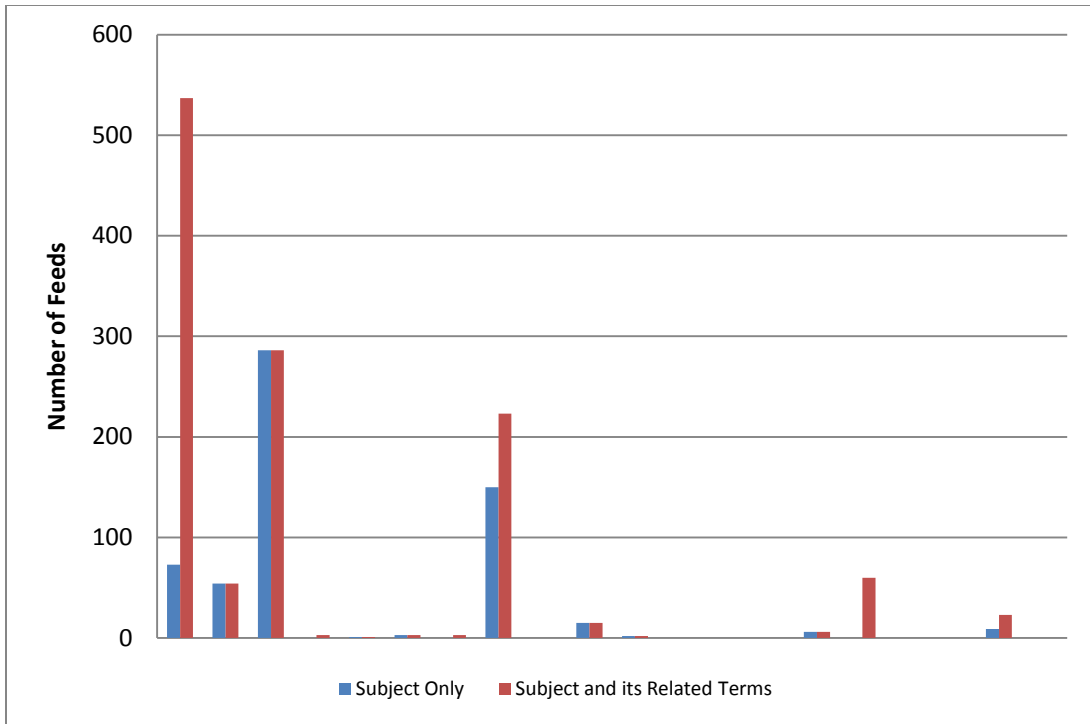


Figure 7.2: Comparison of Search Results (Subject with and without Related Terms).

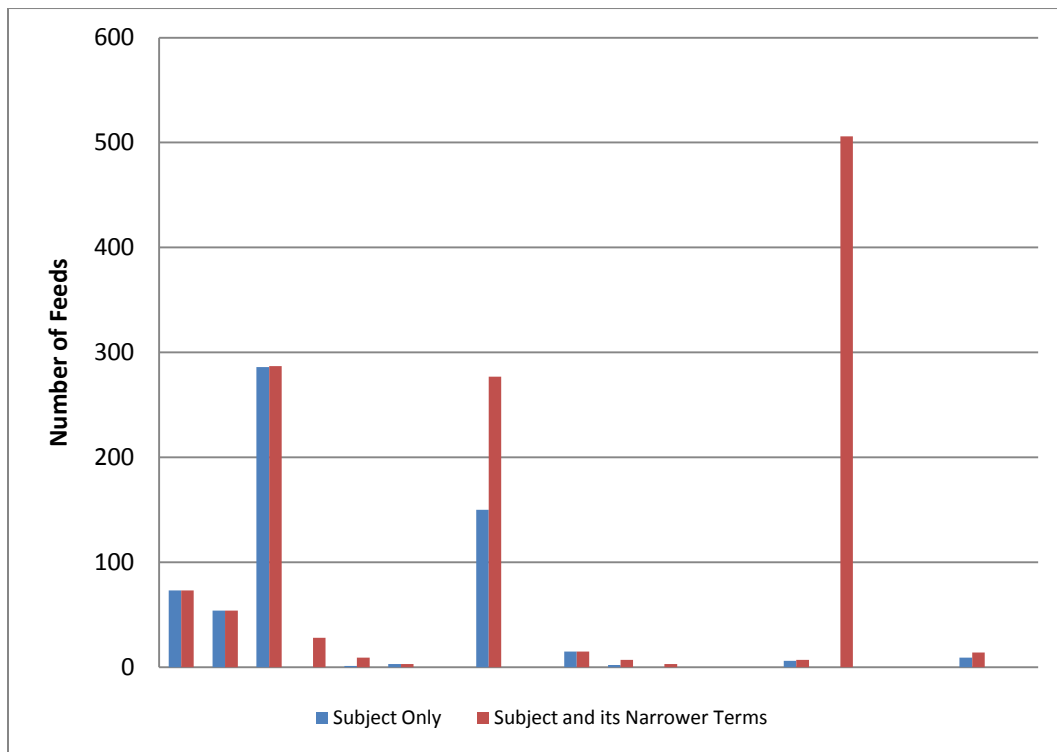


Figure 7.3: Comparison of Search Results (Subject with and without Narrower Terms).

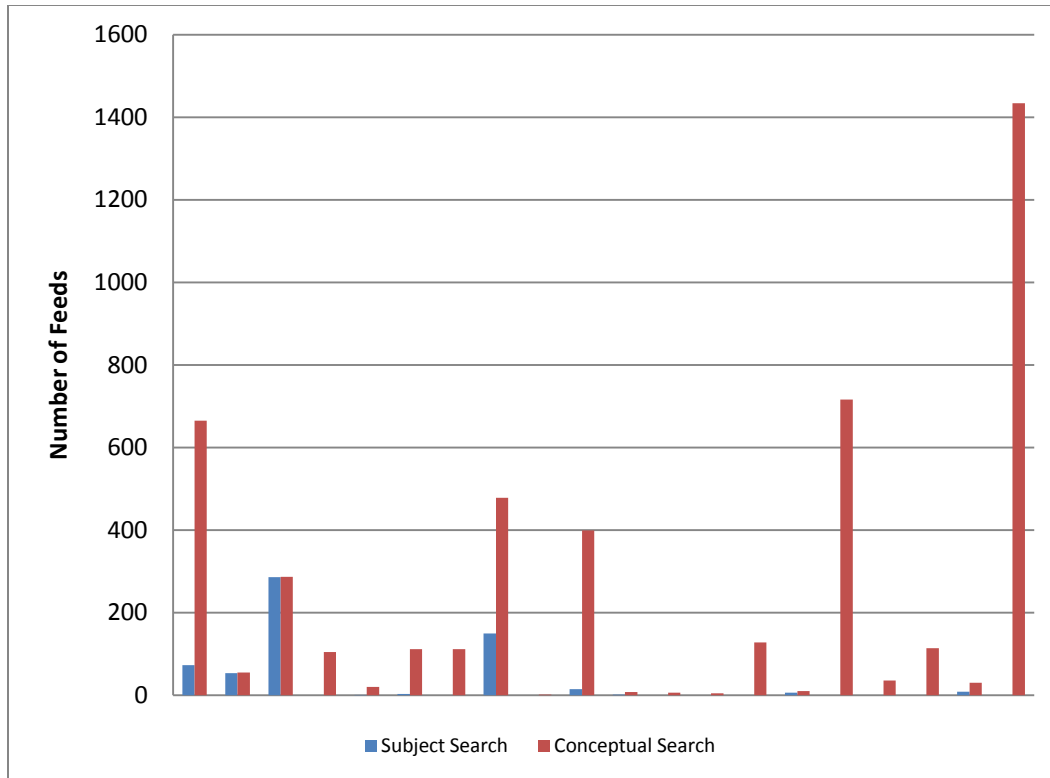


Figure 7.4: Comparison of Subject Search and Conceptual Search.

As the figures show, for some subjects the results are equal. When the search results are equal, one of the following conditions is true:

- Searching Alternate Labels or RTs or NTs were empty.
- Search Result of a subject contains the search results of its Alternate Labels, RTs or NTs.

The Figure 7.4 compares the final result of keyword search and conceptual search for the selected subjects. The chart has been generated based upon the values in Table 4 found in Appendix A. The chart shows that for about 90% of the subjects in this test set, the conceptual

search returns more records than keyword search. However, for about 10% of the records the search results are the same and conceptual search has not impacted the resultset.

It is necessary to consider that the subjects in this test case have been selected from the subjects in the thesaurus that had at least one Alternate Label, RT, or NT, and as a result, the subjects emanate from 50% of the subjects. In addition, the amount of information that a user misses with keyword search depends upon the term that is used in the search. For instance, searching a more common term related to a subject yields many more results than searching for a rarely used term.

One of the other reasons that searching Alternate Labels yields more results in the above testing is that the main term of a subject in the thesaurus is not always the most common term of that subject. The size of the search result depends upon the subject's term that is being utilized from the search. While, conceptually, all of the terms that are used to describe a subject have same value and their result set should be the same. Regarding this fact, the most important advantage of conceptual search is that the search result does not necessarily depend upon the search term; instead, it depends upon the chosen subject.

The fact that for some of the selected subjects, the result of conceptual search is the same as the result of keyword search, doesn't indicate that conceptual searching was not successful to provide more efficient access to the web feeds for these subjects. Since the search has been done on the the content of the available web feeds in the system, , it is possible that by expanding the feed dataset and by retrieving more web feeds about various subjects, the conceptual search for these subjects will yield more results, as opposed to keyword search.

In conclusion, performing conceptual searching may improve the search result of those subjects that possess at least one Alternate Label, RT, or NT. The resultset of this searching

method contains the resultset of keyword search in addition to the resultset of Alternate labels' search and related subjects'.

7.2 Ranking and Sorting the Subject Report

Ranking the subject report has two advantages. As was depicted previously, the subject are ordered by the number of references to these subjects in the web feeds. By increasing the subjects in the user's favorite subject list, it is necessary to sort the subjects by the number of references to them in the web feeds in order to help the users gain access to the most popular subjects much faster. Without this ranking method, the users have to review the a long list of web feeds to identify the top subjects. While with this ranking and sorting method, user can access to the top subjects fast and easy.

On the other hand, the subject report which was described in section 6.2.4 has other considerable advantages in analysis of web feed information. By reviewing the reports from different periods of time, historic trends (eg. Media trends) to the selected subject can be realized. This realization will be quite helpful for companies and businesses, since as was mentioned in Chapter 2, they are interested in understanding the historic trends in order to assist them in making important business decisions. Figure 7.5 to 7.7, represent the subject reports of a user who has added "sport" to his/her favorite subject list for three different months.

Show Feeds From To

[Favorite Subjects](#)

Subject Term	Rank	
Sports	44.5	Show Details

Subject Details :

Subject Term	Renk in This List
Games	32.25
Football	3.5
Baseball	2
Racing	2
Athletics	1.5
Tennis	1
Sports	1
Golf	1
Basketball	0.25

Figure 7.5: A User's Subject Report for April.

Show Feeds From To

[Favorite Subjects](#)

Subject Term	Rank	
Sports	3375.75	Show Details

Subject Details :

Subject Term	Renk in This List
Games	1600.5
Sports	1144
Racing	218.5
Football	97.25
Shooting	86
Baseball	54.5
Basketball	47.5
Athletics	27.75
Lacrosse	16.75
Tennis	8
Golf	3
Olympics	1
Parachuting	0.5

Figure 7.6: A User's Subject Report for May.

Show Feeds From To

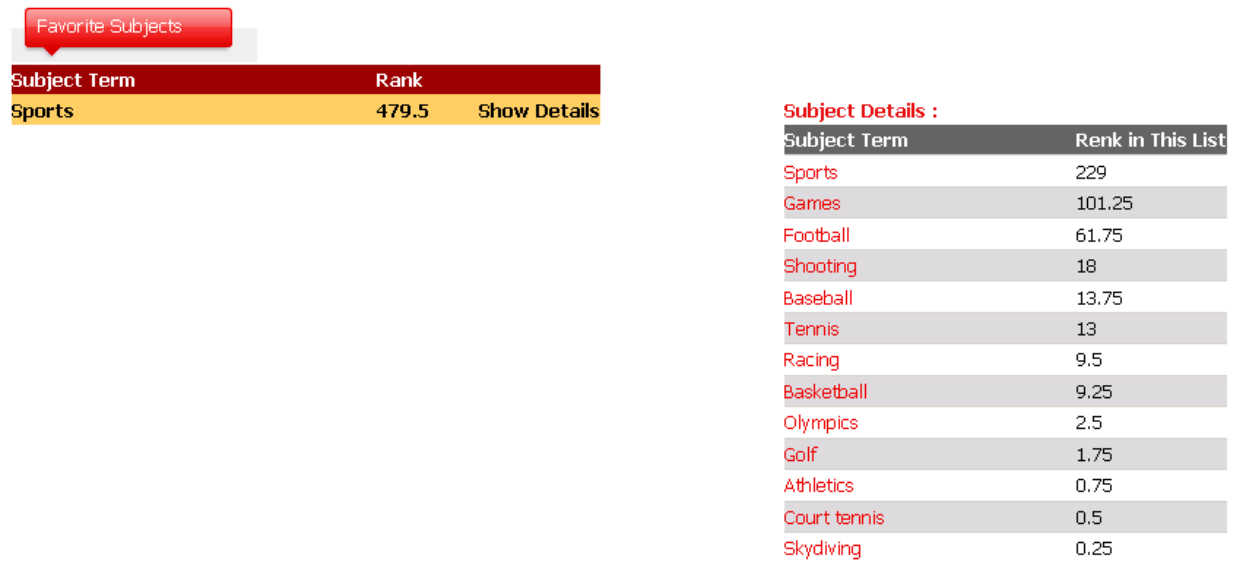


Figure 7.7: A User's Subject Report for June.

Reviewing these reports reveals some interesting facts about the Feeds (news and stories). For example, "Football" was more popular in April and June in comparison to May. In May, "Racing" was more popular than "Football". Another example is "Tennis" and "Court Tennis". These reports show when navigating from April to June, "Tennis" appears more popular. This trend is reasonable, because in the summer, outdoor tennis is more popular. There are more important award tennis matches during this time period; same with golf; with football, April is popular because of the NFL Draft; May has no activities normally, while June is the beginning of training camps and preseason football is close to beginning.

By using these reports, business executives are able to track the social trends for a specific area and make decisions according to these trends. For example, a sport retailers may find this report extremely helpful in gathering information when it is time to order sport supplies. In this way, they may decide to place a higher priority on the supplies for the more popular sports.

Calculating the subject's rank. As was previously detailed, when ranking a subject in the subject report, the subject's power, was used to represent the value of the web feeds which contain that subject from user's point of view. Since in the conceptual search of a root subject (a subject in users' favorite subject lists), the system searches for the subject in an area around that subject in the thesaurus, the most important subject in that area is the root subject, and it may be considered the center of that area. The web feeds which contain this subject has highest value for the users.

The value of the web feeds which contain other subjects in that area is defined by virtue of the distance of the subjects in terms of their semantic from the root subject in the thesaurus. The closer a subject is in terms of the semantic, the higher the power of its corresponding web feeds. Without considering this value, the ranking in a conceptual search would not be accurate, because in the final resultset, all the feeds which contain the subjects related to the root subject have the same value as the web feeds which contain the root subject, though this is not right. For example, consider a user added two subjects to the favorite subject list: subject1 and subject 2 where each has one RT. The number of web feeds which contain subject1 and its RT are n_1 and m_1 . The number of web feeds which contain subject2 and its RT are n_2 and m_2 . If $(n_1+m_1) > (n_2+m_2)$ but $n_1 < n_2$, without considering the subject's power, the total rank of subject1 would be more than total rank of subject2. While the number of web feeds which refer to the subject2 is more than the number of web feeds which refer to subject1. In this way, the conceptual searching and classification ignores the user's priority which was the selected subject. While by using the power, the calculation of the total rank of a root subject considers this semantic priority.

7.3 Ranking and Sorting the Feed Report

Ranking and sorting the web feeds in the report by their relevancy to subjects assists users in accessing more valuable web feeds in faster and easier ways. The value of a web feed from the subject's point of view is defined as the relevancy of that web feed's content to the subject. Microsoft's ranking method for full text search has been used in this report. The produced rank value represents the relevancy of the feeds' description to the searched subject.

Therefore, when a report has many records, users do not need to review all of the report content and evaluate them in order to find the most relevant ones. Instead, the most relevant stories always appear first in the report, saving the users a substantial amount of time.

7.4 Impact of Using an Agent-Oriented Approach and Cloud Computing

Applying an agent-oriented approach in developing the Feed Analyzer has increased cohesion in each agent and has reduced the coupling among agents. In addition, using the agent-oriented approach has led to the increase in scalability of the system. Therefore, changing or adding a new agent has had less impact on the other agents.

Using cloud computing and developing the Feed Analyzer as a cloud service has created many advantages for the system. Along with benefiting from the wide range of on-demand resources that the cloud provides, cloud computing provides the following advantages:

- Using the Microsoft Azure Queue storage for agents' communication has helped maintain low coupling in the system, because agents do not need to communicate directly. In addition, this structure facilitates communication between agents. Without this queue, in order to avoid direct communication among agents, the developer has had to develop a queuing mechanism.

- The Azure Fabric Controller manages and runs the services in the system. Without a controller, the developer has to develop a controlling and managing mechanism because in a agent-based system which is implemented as a group of services, a managing mechanism is a necessity. Otherwise, reaching the system's goal(s) with a group of independent services devoid of the complete knowledge of the entire system would be rendered impossible.
- One of the best advantages of developing the system as a cloud software is increasing the scalability in the system. Changing the number of instances of each service in the system in the cloud is very simple and very fast. However, it should be understood that to benefit from this feature of the cloud, the system design should have the ability to support multiple instances of a service. Otherwise, the system would experience inconsistency in data or would fail to achieve its goal.

7.5 Comparison with Other Systems

There are many feed readers/aggregators and news aggregators available on the Web. Each provides a group of features that facilitates information access and improves the users' experiences in accessing this information. These features include grouping the web feeds by their channel, providing predefined or custom-defined groups in order to organize the web feeds based upon their feed channels, and to provide a normal keyword search.

However, most of them do not provide subject categories to organize web feeds by their subjects. Some news aggregators such as Google News [15] present a limited list of top subjects in a set of predefined groups. By clicking on a subject, the system shows similar stories related to a subject. Although Google News provides many features for users to personalize their pages by

their region or language, there is no known feature for personalizing the news lists by users' selected subjects.

There is also commercial feed aggregator software called the Market News Analyzer [17] which allows users to define a structure of categories and subcategories. For each category, the system only downloads the web feeds that contain the labels of the category and its subcategories. However, users have to define the categories and although they may benefit from customizing the categories, defining the tree structure is a time consuming task. In addition, categorizing the web feeds in this software is only done through single keyword searching. Although it is useful in searching and filtering by specific names, it will not be sufficient enough to simply search the subjects with just a single keyword.

In comparison to these services, the Feed Analyzer provides a conceptual search and a classification of the web feeds. Users may select the subjects of interest to them and the system will only display the web feeds related to those subjects. Therefore, users will not have to struggle with predefined categories of subjects (as is the case in Google News) and do not have to read all of the news in a single feed channel to find their particular stories.

In addition, users of the Feed Analyzer benefit from a powerful and complete subject thesaurus for selecting their favorite subjects. Therefore, instead of defining the tree structure of subject categories and sub categories, users may select the subjects from an authorized subject thesaurus.

Yet, the most important advantage of this system is that in classifying the web feeds based upon the subjects, the system performs a conceptual search instead of a keyword search. Instead of searching a single label that represents a subject, the Feed Analyzer searches all of the

labels of that subject and its related subjects (RTs), along with the specific form of that subject (NTs).

Therefore, when users select a subject, the system then searches for that subject in the thesaurus and the close area around it. Through the conceptual search of a subject in the web feed dataset, users access all the news and stories which refer to the subject with any Alternate Labels that is used to represent that subject. Instead with keyword search, users only access the information that refers to a subject using the searched keyword. In order to achieve the same result as provided by the Feed Analyzer for users in other systems, users must search all of the Alternate Labels of the subject and that requires knowledge of them all. In addition, searching all of the Alternate Labels for some subjects involves sifting through more than 20 different labels and that is an arduous task.

On the other hand, ranking the subjects in this system has two advantages. Much the same as Google News, the system provides the most popular subjects for users, and in addition, the ranking is considering the main subjects that users select.

In Google News only a limited number of top subjects in each category are displayed, and these categories are quite generic and therefore, only the most referenced subjects are represented. Unfortunately, users may be interested in subjects that are not one of the most popular subjects in the news. Instead they might be interested in the subjects which are in a specific area that is attracted by a small group of people. These users do not have a chance to see the subjects in that specific area in Google News unless they employ a regular search in Google News. However, in the Feed Analyzer users may select the areas of interest and the system will display the top subjects in that area and the stories and news about each subject in that area.

Another advantage of the Feed Analyzer is sorting the news and categories related to a subject by their relevancy to that subject. In most feed readers/aggregators, even if users filter the news by a subject, they still must contend with an extensive list of stories and news that even in the best case scenario are in order of their modified or published dates. As a result, users have to read the list to find the most relevant stories, and these lists may be extensive. For example, assume these users have subscribed to 10 feed sources, with the average feed count of 10 in each feed channel. If readers retrieve the feed channels once per day, they have to read 100 web feeds in a day, and worse yet, many websites update their content hourly.

The problem is even more serious in feed aggregators, as they retrieve information from many websites, and even by filtering the news, users still have a huge list of information. For example, in one of the tests of the Feed Analyzer, in one day, 167 web feeds were retrieved containing the term “computers.” Reading a list of 167 short stories simply to find the most relevant news to “computers” is an unpleasant task, to say the very least, whereas with the Feed Analyzer, the most relevant stories to a subject come first in the feed report.

CHAPTER 8: CONCLUSION AND FUTURE WORK

This chapter presents the conclusion and provides the suggestions for future works to improve the proposed classification method and developed application.

8.1. Conclusion

By developing the Feed Analyzer system, the impact of using conceptual classification and sorting the data, in accessing large amounts of information has been examined. Although the proposed conceptual classification method may be used on any text dataset, for this thesis, the web feed dataset has been selected, as it had some very unique features. In comparison to an offline dataset, web feed dataset changes frequently. Therefore, it needs an efficient indexing mechanism to keep track of the changes.

In addition, since the web feeds are published frequently, the feeds dataset expands quickly, so it is easy to test the model against a large dataset. Unfortunately, most of the feed readers/aggregators only focus on retrieving web feeds, while these systems have not seriously considered developing a convenient filtering and classifying method for the web feeds.

The web feeds include short summaries about the actual contents of the websites. Hence, they may contain the main points of Web content; yet, at the same time, they are shorter than the main contents. Therefore, filtering the Web contents based upon their main points by processing their corresponding web feeds is a valuable task that may be completed at a reduced amount of time because the application does not need to process large data from the content to extract the main points of news or a story.

In addition, ranking and sorting the subject categories by the number of references to the subjects not only allows users to identify top subjects but it also helps users keep their fingers on the pulse of global society historic trends through these subjects. On the other side, sorting the

feed report facilitates accessing information, since users may read news and stories that are most relevant to the subject in a much faster and easier way.

From a technology point of view, employing the agent-oriented software engineering has facilitated the development process while increasing the adaptability of the system. In addition, using the cloud platform service supports an agent-oriented approach in a software. The agents in a multi-agent system may be mapped easily to the cloud services and cloud queue services provided by PaaS such as Microsoft Azure and Amazon Web Service and may be used to implement indirect communication among agents in the system.

Moreover, the controller module (Fabric Controller) in Microsoft Azure executes and manages multiple services and multiple instances of each service. Without such a controller, the developer of a system would have to implement a controlling module that executes the agents and test them to ensure that they would function properly. However, even with this feature in the cloud platform services, users may still require a controller module that manages the system's logic.

Besides, the system benefits from the scalability that is brought through cloud computing. The scalability is not just from the resource point of view cloud computing in fact some cloud platform services such as Microsoft Azure, allows the use of the cloud platform to free developers from struggling with hardware resources and network infrastructure, and it also provides more scalability in terms of the number of instances of each service. It would be easy to increase the instances of a service in the cloud by changing the instance count value in the App.config file.

8.2 Future work

Although the system supports RSS as a commonly used web feed standard, in order to cover larger groups of web feeds, supporting other standards, especially the Atom standard is necessary. Fortunately, the system design supports adding this new standard by developing a new type of feed reader to the system. The only challenge with this task is in the implementation of a method for distinguishing the supported standards of the feed sources.

The Feed Analyzer searches all of the Alternate Labels for the subjects in the users' favorite subject lists. This approach has improved the search performance by locating all of the subject's referred feeds. In addition to the subject and its Alternate Labels, the conceptual search algorithm searches subject RTs and NTs (within two levels). Yet, in searching related subjects, only the main labels of these subjects are searched. One of the areas that requires improvement is searching Alternate Labels for the related subjects. This thesis has hopefully shown that searching Alternate Labels of a subject will improve searches.

On the other hand, despite the fact that Library of Congress subject heading is a powerful subject thesaurus, the dataset only includes subjects, though sometimes users require a search of the personal, geographical or chronological names in the news and articles. Therefore, adding a name heading to the system or replacing Library of Congress subject heading with another subject heading that includes these names would certainly help. Unfortunately, a free name heading has not been discovered to import to the system.

Feed Analyzer uses the same interval for retrieving web feeds from all feed sources. While the update rate of different feed sources is not always the same, as websites may vary and may publish new articles or stories hourly, daily, or as each new event occurs. Therefore, using the same update interval for all feed sources often results in duplicate data. The interval may not

be increased too much because the Feed Analyzer could then potentially miss crucial feeds from some websites. Therefore, designing an algorithm to update the feeds based upon the update rate of each website could save resources, because Feed Analyzer will not consume the network and processing resources necessary to normally read a feed channel that is available in the system.

REFERENCES

- [1] H. E. Meyer. (2010, *How to Analyze Information*. Available: http://www.howtoanalyzeinformation.com/How_To_Analyze_Information/How_To_Analyze_Information_by_Herb_Meyer.html.
- [2] Internet Content Syndication Council :<http://internetsyndication.org/>.
- [3] Internet Content Syndication. (2008, Content creation and distribution in an expanding internet universe: A white paper, may 2008. Internet Content Syndication Council. [Accessed: 6/27/2011].
- [4] "Web feed clustering and tagging aggregator using topological tree-based self-organizing maps," in *Intelligent Data Engineering and Automated Learning - IDEAL 2009.*, E. Corchado and H. Yin, Eds. Springer Berlin / Heidelberg, 2009, pp. 368-375. Available: http://dx.doi.org/10.1007/978-3-642-04394-9_45.
- [5] "Global survey: The business impact of big data," Avanade, 2010.[Accessed: 6/20/2011].
- [6] avanade Company web site :<http://www.avanade.com/us/Pages/default.aspx>.
- [7] Web Feed [Accessed: 6/1/2011]:http://en.wikipedia.org/wiki/Web_feed.Wikipedia.
- [8] B. Hammersley, *Developing Feeds with RSS and Atom*. O'Reilly Media, 2005.
- [9] Feed Icon, From Mozilla website. Available: <http://www.mozilla.org/foundation/feed-icon-guidelines>
- [10] RSS 2.0 at Harvard Law 2003, [Accessed:5/20/2011]:
<http://cyber.law.harvard.edu/rss/rss.html#licenseAndAuthorship>.
- [11] Extensible Markup Language (XML) 1.0
2008,[Accessed:6/20/2011]:<http://www.w3.org/TR/REC-xml/>.W3C.
- [12] Atom Syndication Format - Introduction [Accessed:5/20/2011]:
<http://www.atomenabled.org/developers/syndication/>.
- [13] Sam Ruby. RSS and Atom 2008, [Accessed:5/30/2011]:
<http://www.intertwingly.net/wiki/pie/Rss20AndAtom10Compared>.
- [14] News Crawler :<http://www.newzcrawler.com/>.
- [15] Google News :<http://news.google.com/>.
- [16] Sharp Reader :<http://www.sharpreader.net/>.

- [17] Market News Analyzer :
http://www.marketnewsanalyzer.com/Stock_News_Analyzer/rss_news_analyzer_general.htm.
- [18] Google Reader: [http:// google.com/reader](http://google.com/reader).
- [19] Filters for Google Reader, 2008, Available:
<http://googlesystem.blogspot.com/2008/03/filters-for-google-reader.html>, Licence available at:
<http://creativecommons.org/licenses/by/3.0/>.
- [20] Pipes: <http://pipes.yahoo.com/pipes/>.
- [21] Electronic Commerce, From Library of Congress Subject Headings [Accessed:6/1/2011]:
<http://id.loc.gov/authorities/sh96008434#concept>.Library of Congress.
- [22] Library of congress Authorities and Vocabularies : <http://id.loc.gov>.
- [23] "Conceptual classification to improve a web site content," in *Intelligent Data Engineering and Automated Learning*. 2006, pp. 869-877 Available:http://captura.uchile.cl/jspui/bitstream/2250/6158/1/Rios_Sebastian.pdf.
- [24] Marketing, From Library of Congress Subject Headings [Accessed:6/1/2011]:
<http://id.loc.gov/authorities/sh85081333#concept>.Library of Congress.
- [25] Linked Open Data :<http://data.nytimes.com/>.
- [26] R. M. Losee. (2007, 7). Decisions in thesaurus construction and use. *Information Processing & Management* 43(4), pp. 958-968.
- [27] "Complex systems and agent-oriented software engineering," in *Engineering Environment-Mediated Multi-Agent Systems.*, D. Weyns, S. Brueckner and Y. Demazeau, Eds. Springer Berlin / Heidelberg, 2008, pp. 3. Available: http://dx.doi.org/10.1007/978-3-540-85029-8_2.
- [28] R. Kishore, H. Zhang and R. Ramesh. (2006, 10). Enterprise integration using the agent paradigm: Foundations of multi-agent-based integrative business information systems. *Decis. Support Syst.* 42(1), pp. 48-78.
- [29] N. R. Jennings and M. Wooldridge. "Agent-Oriented Software Engineering," vol. 117, pp. 277-277-296, 2000. 6/1/2011.
- [30] "Cloud computing fundamentals," in *Handbook of Cloud Computing.*, B. Furht and A. Escalante, Eds. US: Springer, 2010, pp. 3-19. Available: http://dx.doi.org/10.1007/978-1-4419-6524-0_1.
- [31] A. Tveit. "A survey of Agent-Oriented Software Engineering," 2001. [Accessed: 5/30/2011].

- [32] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang and A. Ghalsasi. (2011, 4). Cloud computing — the business perspective. *Decis. Support Syst.* 51(1), pp. 176-189.
- [33] "Tools and technologies for building clouds," in *Cloud Computing.*, N. Antonopoulos and L. Gillam, Eds. London: Springer, 2010, pp. 3-20 Available: http://dx.doi.org/10.1007/978-1-84996-241-4_1.
- [34] D. Doan. (2009, A developer's survey on different cloud platforms. *ProQuest Dissertations and Theses*. Available: <http://search.proquest.com/docview/304856058?accountid=10639>.
- [35] D. Chappell. "A Short Introduction to the Cloud Platform," 2008. [Accessed: 5/20/2011].
- [36] A. Poggi , M. Tomaiuolo and P. Turci . "An Agent-Based Service Oriented Architecture," .
- [37] D. Chappell. "Introducing Windows Azure," 2010. [Accessed: 6/27/2011].
- [38] Amazon Web Service : <http://aws.amazon.com/>.
- [39] Google App Engine : <http://code.google.com/appengine/>.
- [40] Microsoft Windows Azure : <http://www.microsoft.com/windowsazure/>.
- [41] R. Alonso-Calvo, J. Crespo, M. Garc'ia-Remesal, A. Anguita and V. Maojo. (2010, 5). On distributing load in cloud computing: A real application for very-large image datasets. *Procedia Computer Science* 1(1), pp. 2669-2677.
- [42] J. Celko. (2004, *Joe Celko's Trees and Hierarchies in SQL for Smarties SQL for Smarties*.
- [43] "RDF primer," 2004. [Accessed: 5/10/2011].
- [44] dotNetRDF: <http://www.dotnetrdf.org/content.asp?pageID=Library%20Overview>.
- [45] Indexes, SQL Server 2000 2003, [Accessed:5/25/2011]: [http://msdn.microsoft.com/en-us/library/aa933129\(v=SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa933129(v=SQL.80).aspx).
- [46] RSS Tutorial 2005,[Accessed:6/27/2011]. Available: <http://www.mnot.net/rss/tutorial/>.
- [47] Apache Lucene - Overview: <http://lucene.apache.org/java/docs/>.
- [48] The Apache Software Foundation: <http://www.apache.org/>.
- [49] Microsoft SQL Server 2008: <http://www.microsoft.com/sqlserver/2008/en/us/default.aspx>.
- [50] "Full-Text Search Architecture, SQL Server 2008 R2", [Accessed 5/25/2011]: <http://msdn.microsoft.com/en-us/library/ms142541.aspx>

[51] sys.dm_fts_index_keywords_by_document (Transact-SQL), [Accessed 5/25/2011]: <http://msdn.microsoft.com/en-us/library/cc280607.aspx>.

[52] Full-Text Catalogs and Indexes [Accessed:6/20/2011]: [http://msdn.microsoft.com/en-us/library/aa214429\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa214429(SQL.80).aspx).

[53] Full-Text Search(SQL Server) [Accessed:6/1/2011]:<http://msdn.microsoft.com/en-us/library/ms142571.aspx>.

[54] CONTAINS [Accessed:6/20/2011]: [http://msdn.microsoft.com/en-us/library/aa258227\(v=sql.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258227(v=sql.80).aspx).

[55] FREETEXT [Accessed:6/20/2011]: [http://msdn.microsoft.com/en-us/library/aa258864\(v=sql.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258864(v=sql.80).aspx).

[56] CONTAINSTABLE [Accessed:6/20/2011]:[http://msdn.microsoft.com/en-us/library/aa258229\(v=sql.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258229(v=sql.80).aspx).

[57] FREETEXTTABLE [Accessed:6/20/2011]: [http://msdn.microsoft.com/en-us/library/aa258861\(v=sql.80\).aspx](http://msdn.microsoft.com/en-us/library/aa258861(v=sql.80).aspx)

APPENDIX A: SEARCH RESULTS OF RANDOM SUBJECTS USED
IN EVALUATION OF THE THESIS RESULT

Subject	Result set size (number of found feeds)		
	Subject	Alternate Labels	Subject and Alternate Labels
Architecture	73	128	201
Bankruptcy	54	1	55
Baseball	286	286	286
Belief and doubt	0	74	74
Beverages	1	11	12
Breakfasts	3	109	112
Brunches	0	109	109
Building	150	128	278
Cell interaction	0	2	2
Cell phones	15	389	399
Charities	2	1	3
Child abuse	0	3	3
Chili powder	0	5	5
Church work	0	128	128
Civil rights	6	3	9
Clothing and dress	0	150	150
College sports	0	36	36
Composers	0	114	114
Computer security	9	2	11
Computer software--	0	1434	1434
Computer-aided design	0	2	2
Computers	1715	2	1717
Cooking (Beef)	0	4	4
Crime	149	63	210
Critical care medicine		7	7
Dance	26	36	59
Databases	80	1	80

Decision making	0	16	16
Delta infection	0	3	3
Diabetes	198	71	198
Diabetes in adolescence	0	2	2
Digital video	7	4	11
Disabilities	6	2	8
Drama	311	362	358
East Asia	1	2	3
Ecology	0	3093	3093
Economic development	4	34	38
Energy consumption	2	40	41
Environmental engineering	0	1	1
Fiction	227	710	937
Finance	163	346	499
Golf	23	1	23
Government aid	0	2	2
Graphic arts	0	155	155
Health insurance	7	0	7
Information networks	0	0	0
Insurance policies	2	2	2
Internet marketing	0	7	7
Internet telephony	2	1	3
Internship programs	0	1	1
Judicial power	0	10	10
Labor time	0	8	8
Loans	85	226	307
Marketing research	2	270	270
Microcomputers	0	5	5
Microorganisms	0	5	5
Mining corporations	0	2	2
Motion pictures	0	2067	2067
Motor vehicle drivers	0	2	2
Natural disasters	12	1	13
Neurons	1	6	7
Obesity	11	111	116

Ohio River Valley	0	3	3
Omega-3 fatty acids	4	4	4
Painting	7	1	8
Photographers	78	0	78
Private investigators	4	1621	1621
Radio stations	2	2	2
Restaurants	41	37	78
Space	800	0	800
Space travelers	0	0	0
Tax returns	14	9	14
Telecommunication	0	363	363
Therapeutics	0	3	3
Videoconferencing	7	9	16
Vocal coaches	0	1	1
Webcams	19	2	21
White shark	3	3	3

Table A.1: Result Set Size, Searching Subjects and their Alternate Labels.

Subject	Result Set Size (number of found feeds)		
	Subject	RTs	Subject and RTs
Architecture	73	465	537
Bankruptcy	54	0	54
Baseball	286	0	286
Belief and doubt	0	3	3
Beverages	1	0	1
Breakfasts	3	0	3
Brunches	0	3	3
Building	150	73	223
Cell interaction	0	0	0
Cell phones	15	0	15
Charities	2	0	2
Child abuse	0	0	0
Chili powder	0	0	0
Church work	0	0	0

Civil rights	6	0	6
Clothing and dress	0	60	60
College sports	0	0	0
Composers	0	0	0
Computer security	9	14	23
Computer software--	0	0	0
Computer-aided design	0	0	0
Computers	1715	31	718
Cooking (Beef)	0	0	0
Crime	149	9	158
Critical care medicine		0	0
Dance	26	0	0
Databases	80	0	80
Decision making	0	0	0
Delta infection	0	0	0
Diabetes	198	0	198
Diabetes in adolescence	0	0	0
Digital video	7	0	7
Disabilities	6	0	6
Drama	311	33	344
East Asia	1	0	1
Ecology	0	0	0
Economic development	4	4	8
Energy consumption	2	0	2
Environmental engineering	0	293	293
Fiction	227	0	227
Finance	163	0	163
Golf	23	0	23
Government aid	0	3	3
Graphic arts	0	0	0
Health insurance	7	9	14
Information networks	0	0	0
Insurance policies	2	0	2
Internet marketing	0	0	0
Internet telephony	2	0	2

Internship programs	0	0	0
Judicial power	0	123	123
Labor time	0	0	0
Loans	85	3,372	3,457
Marketing research	2	0	2
Microcomputers	0	0	0
Microorganisms	0	3	3
Mining corporations	0	0	0
Motion pictures	0	0	0
Motor vehicle drivers	0	0	0
Natural disasters	12	0	12
Neurons	1	0	1
Obesity	11	0	11
Ohio River Valley	0	0	0
Omega-3 fatty acids	4	0	4
Painting	7	0	7
Photographers	78	0	78
Private investigators	4	9	13
Radio stations	2	0	2
Restaurants	41	0	41
Space	800	0	800
Space travelers	0	0	0
Tax returns	14	0	14
Telecommunication	0	2	2
Therapeutics	0	0	0
Videoconferencing	7	0	7
Vocal coaches	0	0	0
Webcams	19	0	19
White shark	3	0	3

Table A.2: Result Set Size, Searching Subjects and their Related Terms.

Subject	Result Set Size (number of found feeds)		
	Subject	NTs	Subject and NTs
Architecture	73	0	73
Bankruptcy	54	0	54

Baseball	286	1	287
Belief and doubt	0	28	28
Beverages	1	9	9
Breakfasts	3	0	3
Brunches	0	0	0
Building	150	127	277
Cell interaction	0	0	0
Cell phones	15	0	15
Charities	2	5	7
Child abuse	0	3	3
Chili powder	0	0	0
Church work	0	0	0
Civil rights	6	1	7
Clothing and dress	0	506	506
College sports	0	0	0
Composers	0	0	0
Computer security	9	5	14
Computer software--	0	0	0
Computer-aided design	0	0	0
Computers	1715	1,457	3,187
Cooking (Beef)	0	0	0
Crime	149	79	224
Critical care medicine		0	0
Dance	26	6	28
Databases	80	1	80
Decision making	0	0	0
Delta infection	0	0	0
Diabetes	198	0	198
Diabetes in adolescence	0	0	0
Digital video	7	0	7
Disabilities	6	2	8
Drama	311	60	369
East Asia	1	0	1
Ecology	0	0	0
Economic development	4	0	4

Energy consumption	2	0	2
Environmental engineering	0	1	1
Fiction	227	4	227
Finance	163	6,046	6,053
Golf	23	3	25
Government aid	0	7	7
Graphic arts	0	115	115
Health insurance	7	0	7
Information networks	0	6	6
Insurance policies	2	0	2
Internet marketing	0	0	0
Internet telephony	2	0	2
Internship programs	0	0	0
Judicial power	0	0	0
Labor time	0	0	0
Loans	85	65	87
Marketing research	2	0	2
Microcomputers	0	6	6
Microorganisms	0	0	0
Mining corporations	0	0	0
Motion pictures	0	272	272
Motor vehicle drivers	0	0	0
Natural disasters	12	341	351
Neurons	1	0	1
Obesity	11	0	11
Ohio River Valley	0	0	0
Omega-3 fatty acids	4	0	4
Painting	7	0	7
Photographers	78	3	81
Private investigators	4	0	4
Radio stations	2	0	2
Restaurants	41	5	45
Space	800	0	800
Space travelers	0	21	21
Tax returns	14	0	14

Telecommunication	0	96	96
Therapeutics	0	7	7
Videoconferencing	7	0	7
Vocal coaches	0	0	0
Webcams	19	0	19
White shark	3	0	3

Table A.3: Result Set Size, Searching Subjects and their Narrower Terms.

APPENDIX B: COMPARISON OF SEARCH RESULT FOR ALL SUBJECTS

Comparison of Search Results (Subject and Alternate Labels)

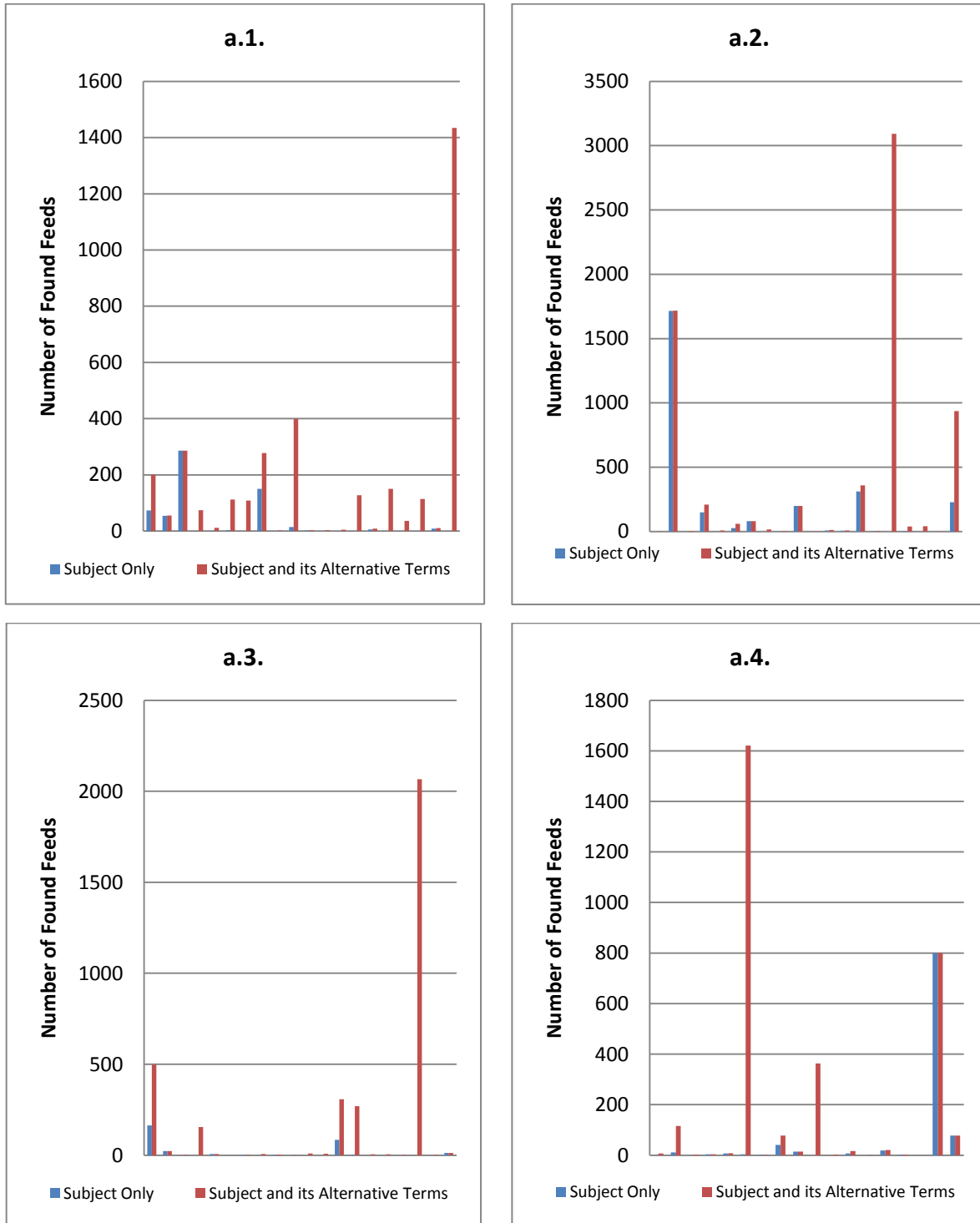


Figure B.1: Comparison of Search Results for All Subjects in Test Set (with and without Alternate Labels).

Comparison of Search Results (Subject and Related Terms)

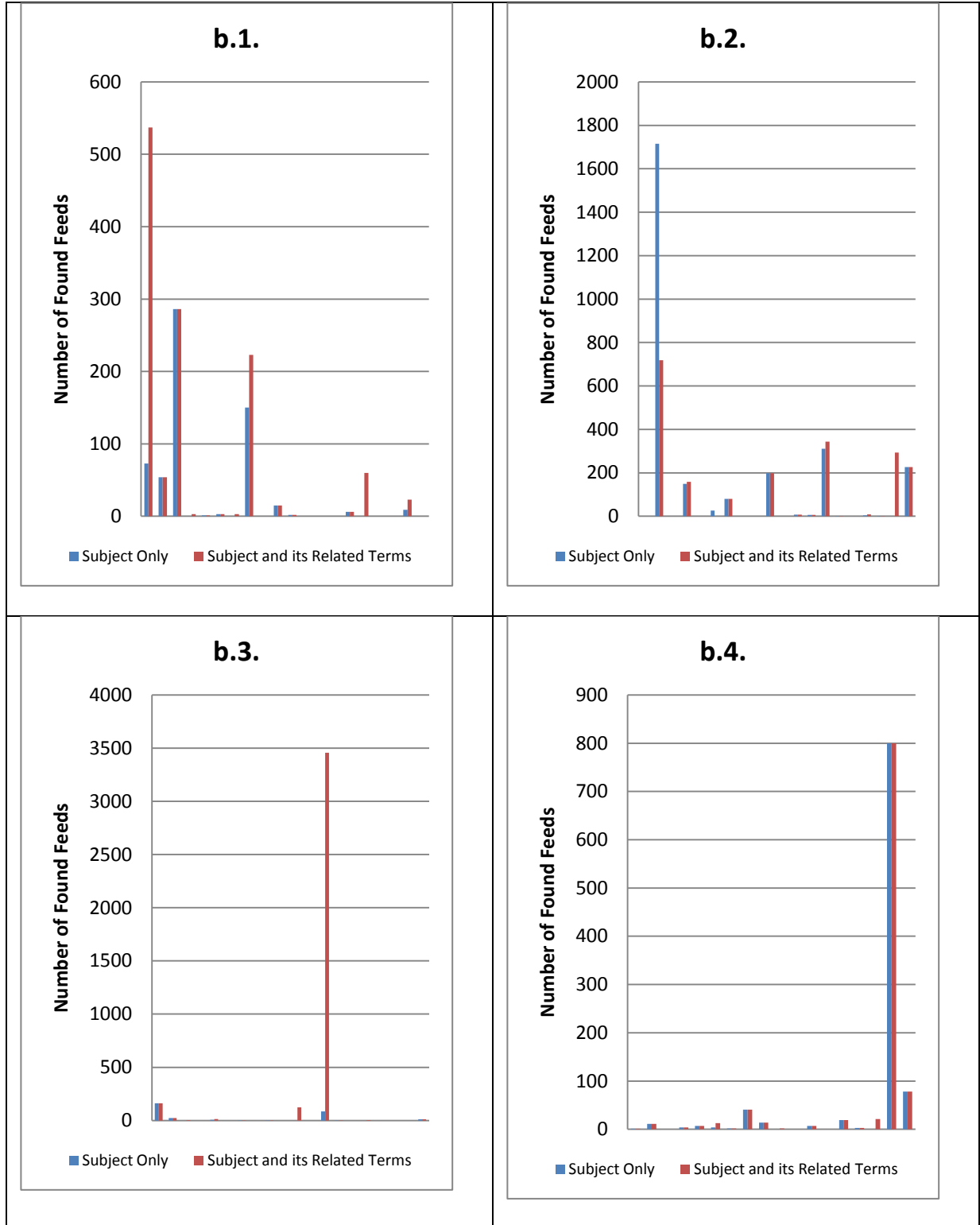


Figure B.2: Comparison of Search Results for All Subjects in Test Set (with and without Related Terms).

Comparison of Search Results (Subject and Narrower Terms)

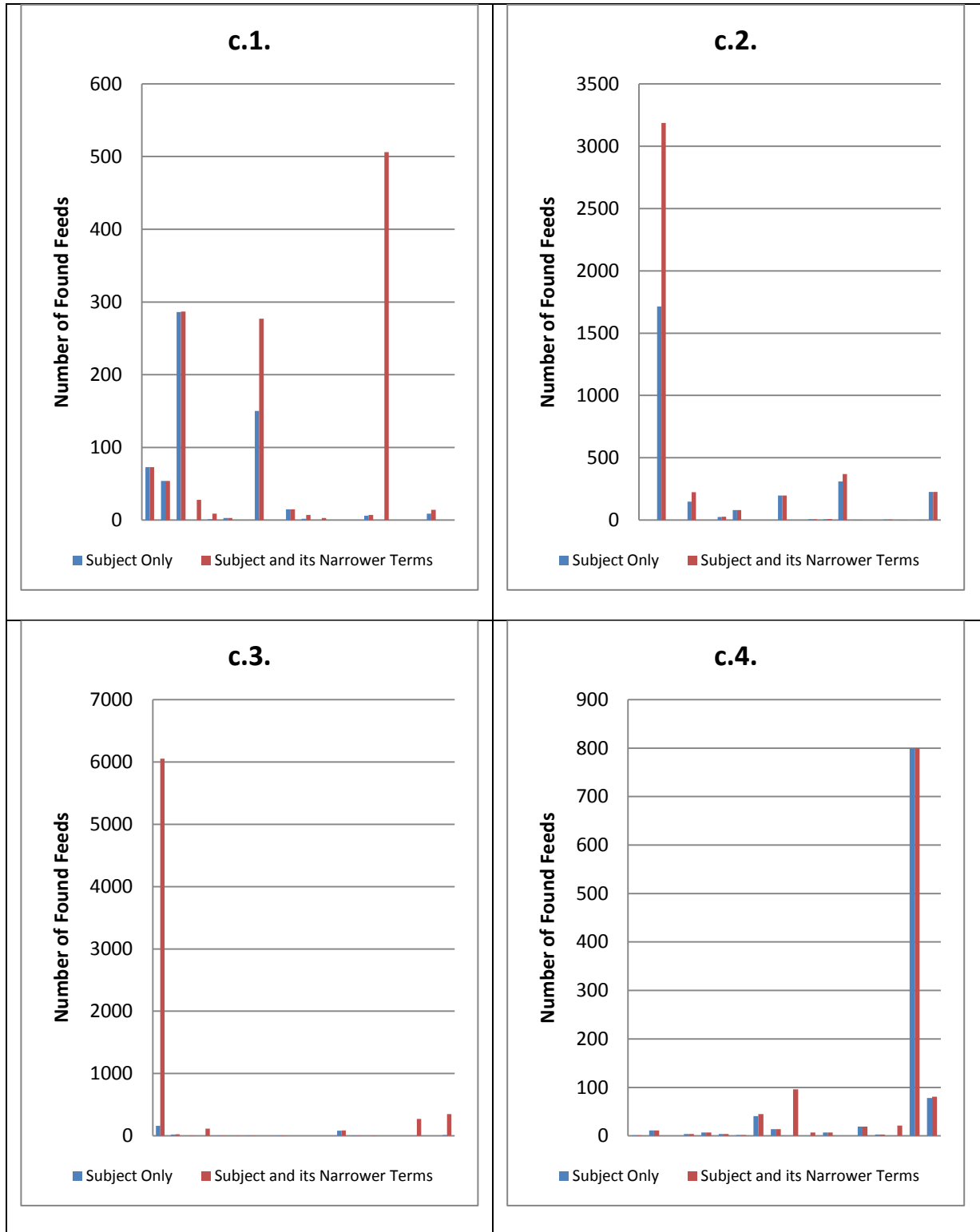


Figure B.3: Comparison of Search Results for All Subjects in Test Set (with and without Narrower Terms).

Comparison of Subject Search and Conceptual Search

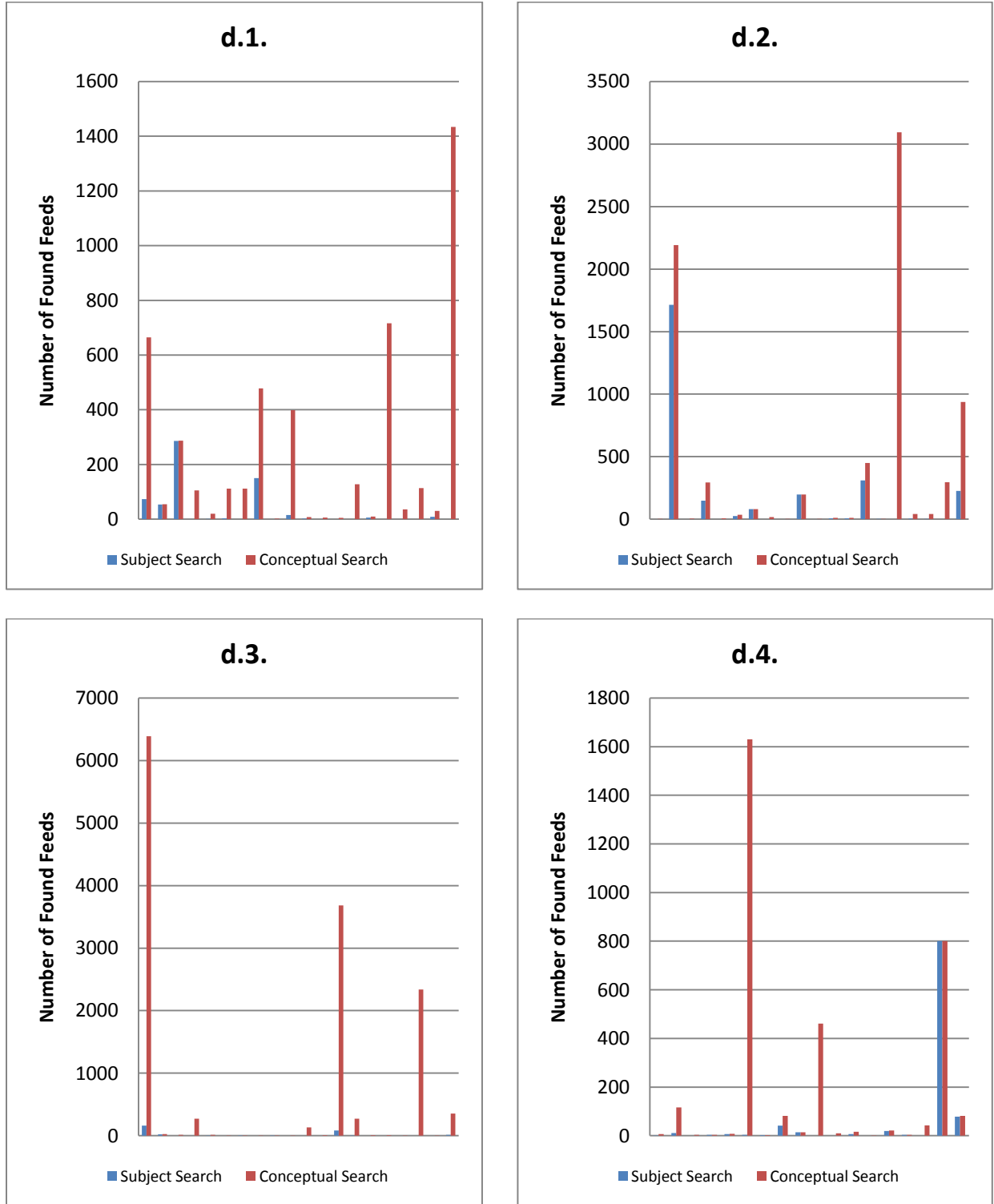


Figure B.4: Comparison of Subject Search and Conceptual Search for All the Subjects in Test Set.

