University of Alberta

BUBBLENET: AN INTERACTIVE INTERFACE FOR RETRIEVING DOCUMENTS

by

Saeed Mohajeri

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2013
Edmonton, Alberta

Abstract

As the size of the World Wide Web and the type of documents it holds grow, the need for tools helping users to find their required information becomes more increasingly important. There are several ways to summarize, navigate through or retrieve documents on the web, such as query-based search and tag clouds; but neither are sufficient for both getting a summary and finding desired documents.

In this thesis, we introduce BubbleNet: a new interface that helps users to get an overview of document collections and explore them by providing an interactive network of topics, their semantic relationships and their related documents. Our experiments show that BubbleNet gives a better overview, is faster and more useful in certain information retrieval tasks.

Acknowledgements

It was impossible to create and develop BubbleNet without the help and support of the kind people around me: to only some of whom it is possible to give a particualr mention here.

First, I would like to express my greatest appreciation to my precious supervisors: Professor Davood Rafiei and Professor Osmar R. Zaïane for their wise and kind guidance, as well as their continious support during my studies. It was a long and challenging road from the idea of BubbleNet to the point of making it happen as a product, and they intensively helped me find my way in this journey.

Also, I would like to thank my colleagues in Database Research Group at the University of Alberta: Hamman Samuel for helping me with ideas, tools and information, Afra Abnar, Afsaneh Esteki, Mohsen Taghaddosi and Samaneh Eskandari for their kind and passionate support in implementation and evaluation of BubbleNet. No one can deny the impressive role of having really good friends in gaining acheivements.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As the size of the World Wide Web grows, and more and more people are using it to find, publish and share information of different types and structures, the need of tools and methods to hold, maintain and retrieve huge amounts of information also becomes more increasingly important.

There are several forms of sharing information on the Web: static websites, news groups, mailing lists, social networks, discussion forums, personal blogs, etc. Recently generated media on the web are more interactive and let people collaborate on sharing related pieces of information. News websites nowadays allow users to comment on news articles, which is a great way for people to discuss about the news. A user can understand how other people think about news as well as how the author of the article thinks. Discussion forums are another group of websites that host several threads of discussions under different topics. Users can post their own comments or reply to others. This makes discussion forums dynamic environments for people to share streams of ideas and thoughts.

When using a website that contains thousands of documents, the user may want to get an abstract/high-level image of the contents of the documents without the need to study all of them. Also, the user may want to easily and quickly find a particular document or topic within all the topics contained in the website.

The first scenario happens when there is a large number of documents that are mentioning different topics or different aspects of a particular topic. Similarly, in a news website where users are encouraged to comment on news articles, there are two main parts of information: the news article itself which is from the authors' or

editors' point of view, and the set of comments that people make on the article that represents the perspective and opinion of the reader on that article. Within a huge amount of comments made on several news articles, it might be difficult to obtain an overall view on the concepts mentioned in the discussions or to find specific comments concerning a particular topic related to an article.

There has been a large body of work on how to summarize texts, news articles and discussions. Some are listed in Chapter 3. Such summaries can be used for helping the user acquire a big picture view of the contents of a website. There are also other approaches for representing a high-level image of major topics discussed in a website, such as word clouds and tag clouds. They represent the most frequent words (or tags) of documents with different sizes together in a page. The size of a word or tag indicates its importance (i.e. how many times it appeared in the documents). We introduce previous works on word clouds and tag clouds in Section 1.1.1.

The second goal is finding desired information within all of the documents in a website, which can be difficult for users considering the growing size of information shared on the web. Even in a single, but long document, the same difficulty may occur. For example, in a discussion forum, when several people take part in a discussion, it is likely to see topic drifts in the thread of comments, which means that new topics that are not originally discussed in the main post of a discussion will appear as people continue the discussion. Thus, a user may be interested in a topic that is mentioned in the middle of a long discussion, which is difficult to find.

There are several methods to help users access their desired information. They can use search tools by entering a query containing keywords that are likely to appear in desired documents. Alternatively, they can use hierarchies of categories and links provided by most of websites to reach a particular document from the first page of a site. Again, word clouds and tag clouds can help users find documents based on frequent words and tags appearing in them.

In this thesis, we introduce another approach to this issue. We aim to obtain an abstract and high-level representation of major concepts discussed in a set of documents (either a set of individual documents, such as news articles, or a set of

parts of a long document, such as individual comments made on the same news article or individual posts in a discussion thread) and to provide an interactive user interface to help users access a desired document. In the following sections, we will describe this approach in more detail.

1.1 Bubbles Network: An Innovative Interface

A human expert (for example, a librarian) can provide some keywords for a document. These terms are a very high-level representation of the major topics in a document. The more keywords we extract, the more detailed information the set of keywords gives about the contents of the document.

A computer program can do a similar task: given a document (and a set of documents as the background or domain knowledge), there are several algorithms for extracting important terms that represent the major topics of the given document. Some of those methods are briefly introduced in Chapter 3. Using keywords, a user can find documents relevant to a particular topic or more effectively query those documents.

Within a corpus of documents, such as a news website with many articles, or a discussion forum with thousands of discussion threads, there are often many documents referring to the same concepts. For example, if one searches Google News for Android, there will be hundreds of documents returned as the results. They all share the same topic, but they are also about different aspects of that topic (such as Android OS, Android Phones, Android Apps, Android Market, etc.) or different events related to it (such as Android Release, Android Update, Android Unveiling).

The collection of these documents represents some information about the topic, for example, the fact that Android is an operating system, it is used on touch screen devices, it is an open-source software, etc. All these topics are related to the topic Android. Also, because of this, as well as many other open-source operating systems such as Linux and OpenBSD, the concepts operating system and open-source are also somehow related. The same relation would exist between terms open-source and application. Such relations can be inferred by looking at the whole set

of documents related to these topics.

When users are trying to find a document, they have a topic in their minds. This might be an abstract concept or sense and the user may have no words in mind to explain it accurately. Using query-based search engines, they have to convert that topic to one or more words and find their desired document within a set of documents containing those words, returned by the search enginge. More advanced search engines help users enhance their queries by suggesting similar words or automatically refining the query to match more related documents.

Our approach, on the other hand, is quite different. In our approach, we aim to provide a high-level representation of topics appearing in the corpus in the form of a network, showing the topics as well as their relationships. These relationships will be defined in the next chapters, but generally, they tend to be an estimation of semantic relationship between topics. For example, if topic A is a major event related to product B, or a product that is often used together with B, or a problem that always rises when people use product B, it is reasonable to expect that a user who is querying for B would like to see A as a related topic to B.

Having such a network, a user can obtain a big picture of all major concepts within a corpus at a very high-level. The user can then navigate through this network: drill down from a topic to see other related concepts in a lower and more detailed level. A user may also want to enter a query (i.e. a term) to start navigation. He or she can then navigate to other related topics and finally find a set of documents related to their desired topics.

This network can also be constructed on a single but long document instead of a corpus of documents. For example, given a news article and hundreds of comments that people added to that article, such a network could be constructed showing different topics and concepts mentioned in the long thread. This way, by looking at this network, a user can find a big picture of the entire discussion at a glance and then navigate to find a specific comment talking about a specific aspect of the article. A similar application can be considered for discussion forums with long discussions.

1.1.1 Comparing to Similar Approaches

As mentioned previously, there are other solutions to these problems. Tag clouds are a method of representation for the statistics of tag usage in a set of documents. Tags are often manually chosen by the authors or other users reading documents and indicate the topics to which documents are related. A tag cloud represents tags according to their frequency, i.e. a tag that is more frequent in the set of given documents will apear with larger font size. Similarly, word clouds represent frequent words appearing in documents. Tags and words are often arranged randomly, however, there are other methods for arranging them in more meaningful ways such as alphabetical order or based on semantic relationships[1] or to fill a geometrical space or to minimize empty spaces.

Figure 1.1 shows a sample text cloud comparing 2002 State of the Union Address by U.S. President Bush and 2011 State of the Union Address by President Obama[2].

Figure 1.1: A sample text cloud comparing 2002 State of the Union Address by U.S. President Bush and 2011 State of the Union Address by President Obama.

Scientists also aggregated tag statistics and the information about semantic relationships between tags to provide a better and more useful representation. Applying tag clustering to visualize tags in clusters was performed by [3]. They show tags in

clusters, combined with a search box to accept user's queries. Their scenario works this way: A user enters a word as a query, and related tags are represented according to the clustering. By clicking on a tag, that tag is added to the query using AND operator. This way, tag clusters are similar to term suggestion methods in search engines.

Another approach was used by The Economist website for representing the most commented topics. They counted how many times the named entities (such as countries, organizations and persons) were mentioned in the comments that users have made on news articles. Entities are represented in bubbles. The more frequent a word appears in the comments, the larger the corresponding bubble. The technology for this bubble representation is provided by Infomous[4], but they have not published the details of their technology as it is a commercial product. Figure 1.2 shows this bubble approach in The Economist website[5].

Figure 1.2: Bubbles approach for representing most commented named entities in The Economist website.

The previously mentioned approaches all have limitations. Tag clouds require manually chosen tags. Word clouds, in contrast, do not choose an appropriate subset of words. For a very large set of documents, we need a system to automatically find good tags to summarize the documents. They also ignore semantic relationships between tags or words, which could be a very powerful assistant for users to find topics. Clustered tag clouds are a solution to this limitation, but they only let users add tags to the query, while a user may want to explore tags and navigate through them without adding the tag to their query.

The bubble representation that is used by The Economist has also its limitations. It only extracts named entities such as names of countries, organizations and persons, and allows the user to find a list of documents mentioning those entities. For a more general application, we need to consider other possible tags (keywords) that properly present the topics. These topics are not equally important, so the user might want to be able to start from a more general or important topic and drill down to reach a more specific and less important one. Another limitation is that the only way that a user can access the comments is by clicking on entities. This prevents users from retrieving comments that include combinations of topics according to relations between pairs of topics.

1.1.2 Thesis Statement

Based on the discussions above, we hypothesize that a user interface that provides an interactive visualization of topics extracted from documents and their semantic relationships will help users in certain information retrieval tasks, including getting an overview of the contents of a set of documents and finding relevant information to a given topic. Such an interface can result in more satisfaction of users in terms of a faster and easier way to find desired information, a better interface for getting an overview of contents of the documents and a more pleasant experience of information retrieval.

Section 7.3 discusses how our evaluations support this hypothesis.

1.1.3 Our Contribution

BubbleNet addresses some of the limitations mentioned in the previous section. Below is the list of our contributions:

- **Keyword Extraction:** We utilize a state-of-the-art keyword extraction and named entity recognition tools to provide an effective method for automatically annotating documents. These keywords will appear in the interface to guide users to find their desired information.
- **Domain-Specific Ontologies:** We open the doors of our system to pre-constructed, domain-specific ontologies to improve keyword extraction and relation extraction. We provide a way to combine domain ontologies and keyword/-named entity extraction tools. Our method is applicable on both document sets and long individual documents, allowing websites to provide a novel method for navigating within a single, but long document.
- **Browsable Interface:** By constructing a network of topics and relationships, we create an interactive interface that allows users to start from a topic, explore the network to reach other topics freely and find the list of related documents without entering a query.
- **Time Trend Visualization:** The way we construct our network lets us provide a high-level image of discussed topics within a specific timespan. This way, users can see the trends of changing topics in a website during the time.
- **Evaluation:** We conducted an effective user experiment to show the usefulness and usability of BubbleNet in certain information retrieval tasks.

1.2 Our methodology at a glance

For building a bubbles network, we have to complete the following stages:

- Keyword extraction/topic assignment: for each document (or each part of discussions), we need to identify one or a few words that represent the main topics discussed in a document.

- Relation extraction/similarity estimation: once some keywords or topics assigned to each document, the nodes of the network can be formed. We then need to extract/estimate relations between those topics based on the contents of documents to form the edges/links in the network.

- User interface: having the network constructed, we need to index all documents based on the network nodes and links and provide a user interface that enables users to navigate to documents using nodes and edges of the network.

We now briefly describe each stage separately.

1.2.1 Keyword Extraction

Keywords are supposed to be representative of the major topics discussed in a document. There are several algorithms for extracting keywords. Some of them are introduced in the next chapter. They use statistical methods, linguistic methods or a combination of both for finding keywords. We combine three different methods: tf-idf, named-entity recognition, and other advanced methods.

1.2.2 Relation Extraction

There are methods for learning ontologies from text that aim to detect relationships: They find relationships such as generalization, specialization, equality, part-of, etc. Some methods are explained in Chapter 3. In our work, in contrast, we do not focus on the type of relationships. The only important point is to capture a relationship (of any kind) between entities. Showing all those relationships can lead users to interpret the type of the relationship.

To extract relationships, we use statistics on the occurrences of keywords in sentences of the documents. The process must be relatively fast so that as the new documents are being added to the corpus, we can update the relationship list on the fly. We also need to store all this information (keywords, their relationships and their ties to documents) in an appropriate data structure to be able to retrieve and update information quickly. Our methodology is described in the following chapters in detail.

1.2.3 User Interface

After building the network in stages 1 and 2, we provide a user interface to help users search/navigate a set of documents using this network. This network enables a user to see major topics and their relationships, drill down to more specific relevant topics, and finally navigate to a particular document or a part of a document.

Chapter 2

Perliminary Definitions and Overview

In this chapter, we define a set of terms and concepts we are using throughout the thesis and provide an overview of the input and the expected output of our proposed system.

2.1 Term Definitions

- **Document:** A document is a piece of information generated by an author or authors. Each document consists of a text (called **body**), a **date** and optionally a text called **title** representing the main subject of the document, and a set of **tags** provided by the author to describe the contents of the document. In different applications, different formats of data can be considered as a document: an individual comment made on an article, an article together with all it comments, a thread in a discussion forum, an individual reply in a discussion or a post in a blog.

- **Entity:** An entity is a phrase that represents a concept, such as a topic, a person, an organization, a location or any other significant concept or object. An entity is said to be **occurring** in a document if that document refers to that entity (either implicitly or explicitly). An entity has a **score** that indicates how "important" an entity is. An entity is said to be important if it occurs in a significant number of documents frequently. In this manuscript, we will use 'term', 'phrase', 'word' and 'topic' all to refer to entities.

- **Relationship:** A relationship is a scored pair of entities, indicating the semantic relation between two entities. Two entities are said to be semantically related if it is likely for a user to prefer to include the second entity in their query to get more accurate results when already using the first one. The score of a relationship indicates how strong a relationship is.

- **Corpus:** A set of all documents in a website or information system.

2.2 The System Overview

Our system is expected to build a network of entities and their relationships from a given set of documents, as well as providing an interactive user interface. The first task can be formulated as follows:

Given a set of documents $D = \{d_i\}$*,*

- *find a set of representetive entities* $E = \{e_j\}$ *such that* $\forall e \in E \quad \exists d \in D : e$ *occurs in* d

- *find a set of relationships* $R = \{r_k\}$ *such that* $\forall e_i, e_j \in E \quad \exists r \in R : r$ *shows the relation between* e_i *and* e_j*.*

The latter part, the user interface, has to provide the following functionality:

- The user, at the first glance, should see the most important entities for a particular period of time. This set must be updated if the user changes the timespan.

- The user should be able to see the entities related to a particular entity by clicking on it. The size of the entity indicates its score. This is done with respect to the indicated timespan.

- The user should see links between the entities indicating relationships. The thickness of the link indicates the score of the relationship. This is done with respect to the indicated timespan.

- The interface is supposed to provide a list of documents when the user points to or right-clicks on an entity or a relationship. This is done with respect to

the indicated timespan. By clicking on an item of this list, the user can access that document.

Chapter 3

Literature Review

As a novel interface for retrieving and navigating through documents, BubbleNet can be considered as a consequence of several attempts to improve methods that users use to access information in a website. The work relates to several areas of data mining and information retrieval. Here we review the related literature.

3.1 Keyword Extraction

Keyphrase extraction or keyword extraction is the process of recognizing representative words within a text. This process can be done manually, as performed by librarians in order to index books and documents. This task can also be performed automatically and there are different methods for finding significant words:

- **Statistical Methods:** In this approach, the statistics of words, such as TF (term frequency: the frequency of appearance of a word in a document), IDF (inverse document frequency: the number of documents in a corpus that contain that word) and the position of a word in the document, are used to estimate how important and representative a word is.

  The idea behind this approach is that representative words usually have special statistics: They tend to appear in earlier paragraphs of a text since the first few paragraphs usually introduce the main topic of the document; they are likely to appear in the document several times as they are important and related to the topic of the document, and at the same time they are unlikely to

appear in many other documents because they are discriminating the topic in which a particular document is specialized.

In [6] the authors introduce a method for extracting important terms and noun phrases in both English and French based on words' statistical characteristics. [7] uses the statistics of compound words for automatic term recognition. Several variations of *tf-idf* were used by [8] for extracting keywords from news articles. [9] provides a survey of these methods.

The advantage of statistical methods is that they are easy to use and do not have much requirements. They can be used without training data and their results are satisfactory.

- **Linguistic Methods:** In these approaches, the linguistic knowledge such as sentence structure analysis, part of speech tags and other linguistic features are combined with statistics to improve the results. This approach is based on the idea that phrases with special linguistic features (such as part of speech) or appearing in some linguistic patterns are more likely to be useful keywords.

  [10] added linguistic knowledge (such as PoS) to term extraction methods from abstracts and reported a significant improvement. [11] combined linguistic information with statistics for multiword term extraction. Wermter and Hahn claimed that based on their experiments, term extraction methods cannot beat simple frequency-based methods unless they utilize linguistic knowledge [12].

- **Machine Learning Methods:** Using training data to learn models of keyword extraction is also a common method. In many applications, there is a training set available, which contains documents with manually extracted keywords. Machine learning based methods use the training set to learn a model to recognise important words. This model can be constructed based on features ranging from statistical and linguistic features to words and n-grams. For example, some words are more likely to be keywords due to their meanings or their grammatical role in text based on samples in training set.

The Keyphrase Extraction Algorithm (kea) is a sample of machine learning based algorithms [13]. Kea extracts keyphrases in two main stages: First, it chooses a subset of phrases in a document as *candidate* phrases. In order to choose candidate phrases, Kea uses some heuristics, such as the number of words in a phrase, stop words, etc. These words are then assessed based on features such as tf-idf and first appearance (position), and keyphrases are selected on the basis of a model learned from the training data.

Machine learning based methods can be domain-specific, as they build a model using a training set. This could be considered as an advantage and as a disadvantage. The main positive point with this characteristic is that in applications with predefined domain that we have a set of training documents available, the algorithm can generate better results. In contrast, if we aim to use the algorithm for a general domain, we need a large-enough and diverse enough training set to cover different topics than can be discussed in documents.

Assignment vs. Extraction: Some methods aim to assign keywords chosen from a controlled vocabulary to documents, while others choose phrases from documents. Algorithms of the first group are appropriate for domain-specific applications where pre-constructed vocabularies are available. In contrast, in general applications (such as our system) we prefer to extract phrases from documents rather than having a large general vocabulary.

Single Document vs. Corpus: From this perspective, keyword extraction methods can be categorized into two main classes: methods that extract keywords from a single document without need to look at a corpus of documents, and methods that require a whole corpus of documents to extract keywords even from an individual one. In [14], for example, the authors propose a method for extracting keywords from a single document, while many other methods (including the methods we cited above) use a corpus to train models before extracting keywords. Using a corpus in a keyword extraction method has the benefit of generating more corpus-specific results, which might be more accurate. But the disadvantage is that it is not

useful for applications for which there is no corpus available.

In our case, the database is supposed to be constructed incrementally. That is, there will be a set of new documents arriving everyday, and those documents will be added to the database. Thus, particularly in the early stages, the database is not populated enough to be considered as a corpus of documents. An ideal algorithm should be able to use individual documents (and any external source of knowledge, such as a trained classifier) only. But there is still the possibility to use corpus-wide statistics using so-far constructed database. Indeed, it is impossible to use algorithms that need a training set since it is not reasonable to train the algorithm everyday on a huge amount of data.

3.2 Named Entity Recognition

While keyword extraction aims to find terms and phrases that summarize the most important points discussed in a document, named entity recognition finds terms and phrases that refer to special entities, such as persons, organizations, locations, facilities, etc.

There are different approaches to this problem:

- **Rule Based:** Having a set of patterns in which named entities are likely to appear, an automated system can recognize named entities. [15] introduces a system that uses a set of rules to find named entities. For example, names of persons usually consist of a first name and a last name, both capitalized, with an optional middle name. These names are also likely to be followed or preceded by professional titles.

- **Machine Learning Methods:** If there is a set of training documents available, machine learning can be used to build models for recognizing named entities. A piece of text can be seen as a sequence of phrases. Each phrase has a label that indicates its type, such as person, location, product, or non-named-entity. Markov chains and conditional random fields are then used to predict labels of new documents based on models created using training data.

The authors in [16] propose a machine learning based method for extracting named entities from advertisement text.

Named entity recognition methods mostly work for well-written and formal texts such as news articles. Liu et al. introduce a method for named entity recognition from tweets, where texts are short and there is no training data available [17].

Named entity recognition can improve the quality of summarization, as mentioned in [18]. Named entities are ideal for our system, as we are interested in extracting names of persons, locations and other important entities from documents. An ideal method for our system should have a trained classifier so that it does not require training data and can be used for our incrementally growing database.

3.3 Topic Modelling

Another set of methods has been developed for assigning topics to documents. This task is different from extracting important terms, but can produce similar results. A topic modelling method gets a document or a set of documents as input and assigns one or more topics to each document. *Topics* are not necessarily well-known terms. That is, a topic is not something like 'sports', 'politics' or 'obesity'. It is an abstract assignment of several documents to a shared subject. Mathematically speaking, a *topic* is a probability distribution over a vocabulary of words. Topics do usually correspond to an actual topic in the real world. For example, a topic can be equivalent to 'University of Alberta Budget Cut', while another topic can represent subjects about 'BC Pipeline'. From a mathematics point of view, the former gives more probability of appearance to words like *student, salary, president* and *cut-off*, while the latter makes words such as *environment, oil, energy* and *ecology*. Words like *Alberta* and province are likely to appear in both topics. A document, then, is a mixture of different topics.

These algorithms generally use some form of generative models where it is assumed that the text is produced from a topic model, and they then try to find that model using singular value decomposition [19] or, more recently, non-negative matrix factorization [20]. More advanced methods are Latent Semantic Analysis [21]

and Latent Dirichlet Allocation [22]. In Latent Dirichlet Allocation (LDA), the generative model is supposed to have a Dirichlet distribution. Each topic is seen as a probability distribution over words and each document is seen as a probability distribution over topics. LDA then tries to find these distributions based on the sets of words (documents) it gets as input. By doing this, LDA assigns a topic to every word in every document with a confidence level, and thus, a mixture of most probable topics to every document.

In LDA, the number of topics to be found is fixed before training. This means that the user needs to know how many topics the documents in a corpus belong to, and then ask LDA to find them. There are extensions to LDA to remove this limitation. One of them is nested Chinese Restaurant Process [23]. This process aims to build a hierarchy of topics and does not require the number of topics to be fixed before running the process.

Since topic modelling methods need a training stage that has to be done on the entire corpus, they are not appropriate for our work, because we assume that our database will be created incrementally (updated as new documents arrive daily). This requires us to use methods for annotating documents that can use only the individual document (and, of course, any other information such as a pre-trained classifier, but not the rest of the corpus). Furthermore, topic modelling algorithms are relatively slow and are not appropriate for applications that require high speed.

3.4 Automatic Summarization

The task of automatic summarization aims to make a summary of a document so that a user can understand the main points discussed in a document by reading the summary instead of the entire document. Thus, a summary is a short, coherent, readable text that briefly covers most important stories mentioned in a document. When dealing with long documents or large number of documents, studying summaries helps users save time while getting an important portion of information.

There are several approaches to automatic summarization. Here we briefly explain each approach:

- **Extraction-based:** In this approach, a summary is generated by extracting important clauses and sentences from text. Therefore, the summary will be a subset of sentences forming the document. Several heuristics can be used to find the most important and informative sentences, such as keywords in a sentence, position of a sentence and its structure. For example, sentences in the first paragraph often carry significant information about the main points of a document, as well as sentences containing several keywords. Important clauses and sentences can be recognized based on such rules or using machine learning when a set of documents and their human-extracted summaries is available. This way, the task of extraction-based summarization can be seen as a classification task. Chuang and Yang, for instance, propose a method for extracting summaries using machine learning. In their method, sentences are divided to segments and some features are calculated for each segment, including position, average term frequencies and number of title words. A model trained on labeled data is then used to find important segments to be extracted [24].

  Considering a thread of comments made by readers on a news article or blog post, researchers have also developed methods for recognizing the most commented parts of an article/post and summarize it based on this characteristic [25][26]. In this approach, finding a sentence or paragraph in an article that corresponds to each comment is used to find the most commented, and therefore, the most important parts of the article to be included in the summary.

- **Abstraction-based:** A human can build a synopsis of contents of a document by reading it. If a machine does the same, a summary can be then generated from that synopsis using natural language generation technologies. In this approach, a model of the story discussed in a document is built and then a coherent and human-readable summary is produced using that model. Thus, the sentences appearing in such a summary are not necessarily part of the document. Some methods also use humans aid to generate summaries, such as getting highlighted candidate paragraphs or requiring post-processing by

human experts.

- **Multiple-Document Summarization:** While many summarization algorithms produce summary of a single document, there are also methods for generating summaries of multiple documents. Having several documents related to similar topics, an automated summarization task should produce a summary that outlines all important points in all documents. This is something more than generating separated summaries for each document and then concatenating them since documents might be sharing topics and discussing different aspects of them. Thus, multi-document summarization should extract information from documents and then bring them together to produce a consistent summary. Therefore, the task of multi-document summarization can be seen as an application of information extraction. [27] discusses the challenges and prospects of multi-document summarization via information extraction.

Summarization vs. Keyword Extraction: A summary is a coherent text that contains a brief description of the stories mentioned in a document. In contrast, keywords only represent the main topics of a document and do not tell the story about them. For example, consider two articles about the University of Alberta budget cuts, one criticizing this issue and the other supporting it. The set of keywords extracted from both might be very similar, as both documents are discussing the same topic, but summaries have to be different as the documents are describing two opposing perspectives.

In BubbleNet, we assume that the user wants to get a picture of all important issues discussed in a corpus and be able to easily find articles related to a particular topic. To this end, keywords and named entities are good representatives, while textual summaries are not appropriate.

3.5 Document Clustering

Putting documents in categories based on their contents is called document clustering. Using document clustering, we can organize documents of a corpus such that they can be accessed through topical categories. This has several advantages:

First, by looking at clusters, a user can infer the main categories that documents of a corpus fall into. In addition, it helps users find their desired documents because similar documents (documents that share similar topics) are placed together. This also helps users explore documents that were not exactly matching the keywords they had in mind at first, but are related to their aimed topic and users might be interested in them.

There have been several researches conducted in the field of document clustering. There are two main approaches to this problem:

- **Hierarchical Clustering:** In this approach, all documents are first considered in a single cluster. This cluster is then divided into several clusters based on the contents of documents. This way, a document falls under one and only one cluster in second level, while all documents belong to the same set in first level. The process of division continues until a hierarchy of documents gets formed. This process can also be done in reverse direction, i.e. every document is an individual cluster at first and then in each step, similar clusters are merged together to form larger ones. A label will be assigned to each cluster based on the contents of its documents.

  For clustering, the similarity of documents has to be measured. Different measures can be used to estimate similarity of documents, including simple text-mining features (bag-of-words and cosine similarity), page titles and meta data and link structures.

- **K-means:** Despite hierarchical clustering, k-means method generates a flat set of clusters. In this method, k documents are selected as centroids to represent k clusters of documents. Other documents are then assigned to those clusters. This method is more efficient, but results in a non-hierarchical cluster structure.

In [28] a comparison of document clustering methods and their efficiency is provided.

Clustering of Comments and Tweets: Having a large volume of comments or short posts (such as tweets), researchers aimed to group comments that are similar

in topic or direction of the opinion they express. [29] [30] In [31], the authors use Latent Dirichlet Allocation (described in section 3.3) to annotate social network blogs and short posts.

Document Clustering vs. BubbleNet: As discussed above, the result of document clustering is a grouped structure of documents, either in a flat or hierarchical form. In a flat clustering, each document belongs to one category, and in a hierarchical structure, documents belong to a set of categories, from the most general to the most specific.

This has some characteristics. First, a document cannot belong to more than one category (or in a hierarchy, a document cannot have more than one parent). This limits the freedom to model the topics appearing in documents when we are not only interested in general topics, but also in more detailed ones discussed in documents.

For example, consider a document related to built-in Google search features in Chrome web browser. This document can be categorized under, for example, 'web search' category, or under a hierarchy like 'technology / world wide web / search engines / google'. This document, however, should belong to categories such as 'chrome', 'information retrieval', 'user interface' and 'web browsing'. It may also refer to Chrome plug-ins, compatible operating systems and comparison to other web browsers. All these topics cannot be captured in a clustering at the same time.

Ideally, what we aim to do in BubbleNet is to capture all these topics mentioned in this document. Furthermore, we want to provide users with links to similar topics. For example, documents related to other built-in features of Chrome, other properties of Chrome, other web browsers such as Firefox and Safari and other search engines such as Yahoo! and AltaVista. To this end, we extract keyphrases representing those topics and entities, remembering which documents mentioned them. We then build a network of these topics based on relationships between those entities. This is quite different from clustering, as a document can be linked to several nodes, and nodes are interconnected based on their similarity or relationship.

Figure 3.1 shows the difference between document clustering and BubbleNet, how the concepts are organized and how the sample document mentioned above is

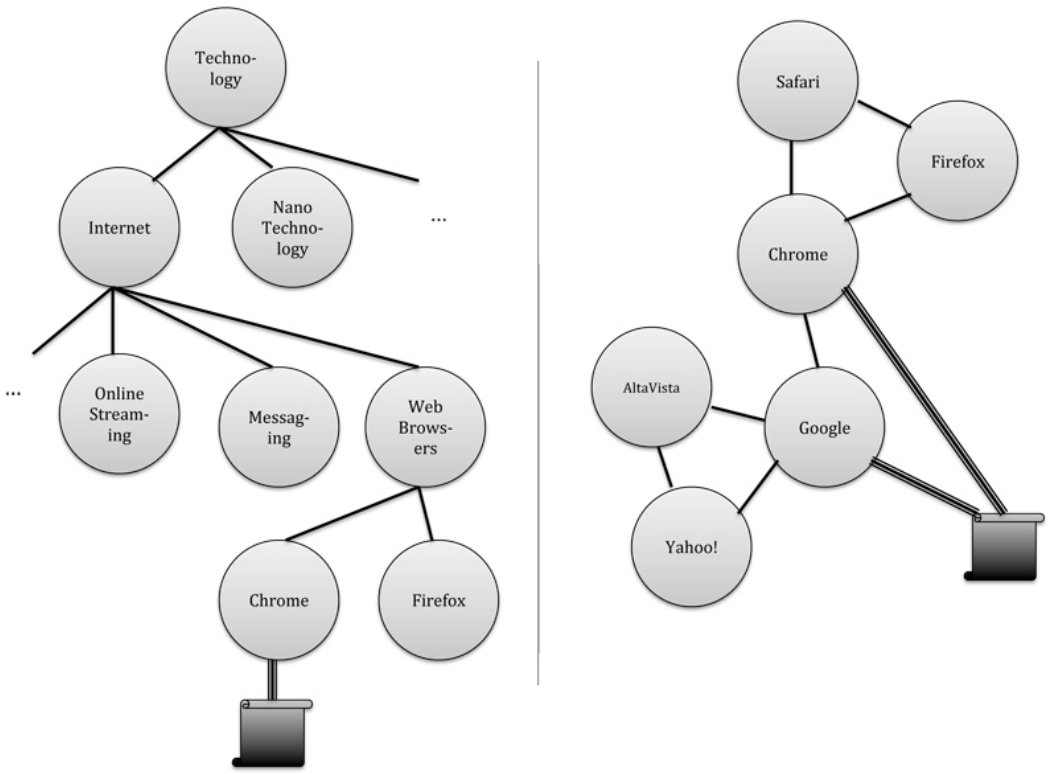

Figure 3.1: Comparison of a sample hierarchical clustering to a sample BubbleNet entity network.

linked to these concepts.

3.6 Query Expansion

In order to help users find exactly what they want, search engines use methods for helping users perform better searches. In most search engines, users have to enter a query (which is a set of words or phrases, optionally combined with some operators) to express what they have in mind. This is not always an easy task for users to do, as they may not be aware of all words used in the literature to refer to the concept they mean, or they may have difficulties finding the words that precisely explain their desired topic.

Search engines basically try to retrieve documents that contain words mentioned in user's query. This obviously requires preprocessing on queries, as users may have entered words incorrectly. Removing stop words (such as 'of', 'the' and 'a'), spell correction and punctuation refinement are some basic preprocessing steps. Search

engines then try to perform more advanced refinements, as we discuss below.

- **Stemming:** Stemming words of a query (such as converting 'negotiating' to 'negotiate') will result in more documents being retrieved and the chance of missing relevant documents to be decreased.

- **Finding Synonyms:** Adding synonyms of query words also improves search results, such as adding the word 'buy' to a query containing the word 'purchase'.

- **Re-weighting Words:** Words in a query are not necessarily having equal importance. Re-weighting methods assign an importance to each word based on relevance feedbacks to improve information retrieval [32].

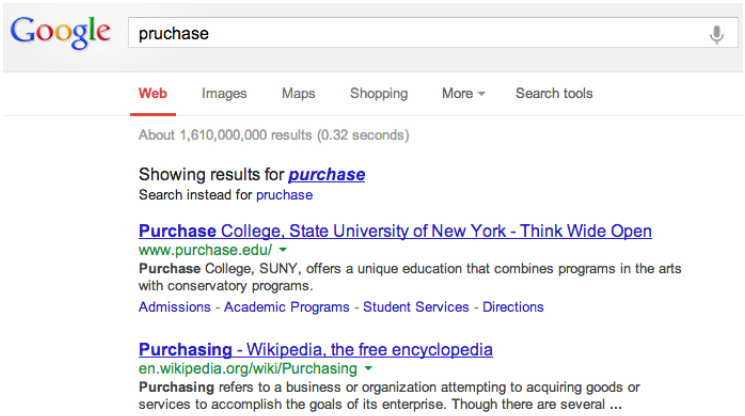

Figure 3.2: A sample of spell correction in Google search. (Captured on June 16, 2013.)

Researchers in this field applied ontology-based query expansion to improve search results by adding words extracted from ontologies such as WordNet [33]. There is an important issue to be considered when using query expansion. As the query is manipulated by an information retrieval system, the recall is expected to increase since more documents are now matching the refined query, but the precision can drop because the refined query is different from what the user entered.

3.7 Term Suggestion

In addition to query expansion, search engines try to help users refine their query at the time they are entering the original query by suggesting terms. This helps users find words that best describe what they mean. Search engines suggest words that are relevant to the rest of the query (i.e. the part of query that user has entered so far). There are several ways to find these relevant words:

- **Query Logs:** By analyzing a large number of queries that users have entered before, a search engine can find the words that are likely to appear together in queries. For example, there are thousands of queries including words 'Edmonton', 'weather' and 'forecast'. Thus, it is easy to guess that if a user enters 'Edmonton weather', he or she would like to add the word 'forecast' as well. Query logs are great sources of information about how users use keywords to find documents and some research has been done on using them [34].

- **Users' Feedback:** In addition to co-occurrences of words in queries, query logs can be analyzed to find how users change their queries by adding, removing or reordering words in their queries after seeing search results. If many users add word B to a query containing word A after seeing the search results, it is very likely that others also would like to do the same. Thus, a search engine can use this information to suggest word B when a user is entering word A as a query [35].

- **Semantic Relations:** Words that are semantically related to query words are also good candidates for suggestion. Having a network of semantically related words, a search engine can provide users with a list of suggestions that are related to the words appearing in the query. There are several ways to infer semantic relationships between words. Shieh et al. propose a method for building a social network of words where nodes represent terms and edges represent semantic relations between them. To build this network, they use Wikipedia articles and lists of their authors, assuming that if an author writes

several articles, their topics tend to be relevant. They compare the results of their proposed system to another method that uses WordNet, a public ontology of English language to extract relevant words [36].

- **User Specific Suggestion:** By profiling recent queries of a user, as well as information about user's location, institute or company and so on, search engines can suggest more personalized terms to a specific user. This can improve user satisfaction, specially when the query is ambiguous [37].

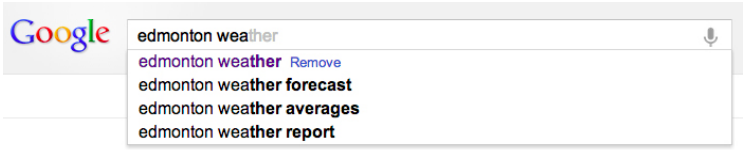

Figure 3.3: A sample of term suggestion in Google search. (Captured on June 16, 2013.)

Term Suggestion and BubbleNet: Term suggestion has some aspects in common with BubbleNet. In BubbleNet, we aim to provide the user with a set of related topics to every particular topic or entity. This is similar to what term suggestion algorithms do in terms of finding relevant terms and phrases to another word. But the difference is the way we extract relationships between entities. In term suggestion, query logs and statistics from a huge amount of crawled web pages by search engines can be used to infer relationships. In contrast, in BubbleNet, we do not have access to such information. Although it is possible to embed knowledge bases extracted from such resources in BubbleNet, it is incompatible with the mission of BubbleNet. The reason is that in BubbleNet we aim to extract entities that occurred in documents of a specific corpus (for example, within the domain of a website). Similarly, we want to extract relationships based on the contents of those documents, not based on the contents of the entire web. For example, we know that in the real world, Google and Yahoo! are related entities since both provide web search and other similar services. Yet in a corpus that contains two different sets of documents, one talking about Google Maps and the other talking about Yahoo! Weather, it is possible that no document talks about both Google and Yahoo!. Thus,

BubbleNet should not consider a relationship between those two entities as they are not relevant according to the contents of the corpus.

However, other methods of relation extraction, such as statistical methods and ontology-based methods can be used to extract relationships in BubbleNet. More discussion about relation extraction methods is provided in Section 3.9.

3.8 Word Sense Disambiguation

When a user enters the word 'apple' as a query, it is difficult for search engines to find out if the user means apple, the fruit, or apple, the company. There are several words that have more than one meaning and therefore, they can fit in several totally different contexts. This is a very common, and consequently, widely studied problem in information retrieval.

There are two main approaches to this problem. The first, called *deep approach*, is based on a deep knowledge of facts in the real world. For example, consider the following two sentences:

- John eats an apple every day.
- Apple is trying to win the competition of cellphone production with Google.

Knowing the facts that 'a human can not eat a company or a computer' and 'a fruit is not often interested in competing with technology companies' can help search engines disambiguate the sense of word 'apple' in these sentences. This approach can generate more accurate results, but is often impossible or difficult as it requires a huge machine-readable knowledge base.

Despite the first approach, one can guess the sense of a word without understanding the text by looking at clues such as surrounding words. In the above example, the word 'eats' is an evidence that 'apple' is in the fruit sense, but the words 'cellphone', 'production' and 'Google' suggest that 'apple' is in the company sense.

In [38], the authors provides an introduction to word sense disambiguation methods. The authors of [39] provide a method that forms a network of keywords to detect word sense communities and use it to cluster web search results.

Although ambiguous words cause ambiguity in BubbleNet entity network, we do not consider this problem in this thesis to keep the problem simple. As a future work, entity disambiguation should be added to BubbleNet system.

3.9 Relation Extraction

The problem of finding relations between entities from unstructured text is widely studied. In general, there are various kinds of relationships that can be considered for different applications, such as typed relationships (*is-part-of, is-member-of, belongs-to, is-owner-of, born-in, etc.*), synonymity, generalization/specialization and topical association. Researchers have proposed several methods for extracting such relationships. Below we describe different approaches to this problem. We limit our discussion to binary relationships (i.e. relationships between two entities).

- **Co-occurrence:** Word co-occurrence is a traditional approach to find semantic associations between words. Spence and Owens in [40] study theoretical aspects of this approach, as well as its practical applicability. They propose hypothesizes about the correlation of the number of co-occurrences of pairs of terms in specific windows of text and semantic association between terms. Their findings show that the words that are semantically associated are more likely to appear close to each other in a text. They state that the number of co-occurrences of associated pairs of terms forms a Poisson distribution, peaked when terms are one right after the other (less than 10 characters distant) and then decreasing as the distance increases. Up to a window size of 250 characters, co-occurrence statistics are highly correlated with semantic association of term pairs and this effect remains observable until the window size reaches 1000 characters. This indicates that even words appearing in separate sentences can be associated if they frequently co-occur.

- **Feature-based Classification:** The task of relation extraction can be seen as a classification task: given a pair of entities, assign a class to the pair that indicates whether they are related or not, and if they are relevant, what type of relationship exists between them. As this method extracts typed entities, it

mostly considers terms appearing in the same sentence because there are rare cases that separate sentences indicate typed relationships. These methods are used for learning ontologies from unstructured text [41].

The classification task described here needs feature engineering to choose appropriate features, as well as enough training data to build models. Some features that are commonly used for this task include entities themselves, lexical contexts and syntactical features. For example, if two entities are both persons and the surrounding words are 'is' and 'father', it is likely that the relationship is of type *is-father-of*. As another example, if the first entity is a person and the other one is a club or party, it is likely that they are related in type of membership.

- **Pattern-based:** Along with feature-based classification methods, patterns are widely used for relation extraction. The idea behind this approach is that sentences that indicate typed relationships between entities tend to follow particular patterns. For example, patterns such as *X was born in Y* or *In Y, X was born* are common forms of indicating birth dates of persons. Similarly, ownership can be found in sentences like *X belongs to Y*, *X's Y* and *X of Y*, although the last one may indicate other relation types as well.

  Patterns can be written manually by human experts, or learned from annotated data, or learned from text itself. The last method is called *bootstrapping*. It starts with a set of seeds (i.e. pairs of terms with known relationships). These pairs are then thrown into the corpus to catch sentences that contain them. By looking at these sentences, patterns that indicate those relationships can be inferred. These patterns are then applied to the corpus to find new pairs of related terms and then new patterns again. By repeating this cycle, the set of patterns grows and helps the system extract related entities.

Several researchers applied relation extraction methods on specific domains, such as medical text. The authors in [42] extracted temporal relations from clinical text using temporal information of entities, and [43] extracted drug-drug interactions by processing medical texts. Chapter 2 of [44] provides a survey of supervised

relation extraction methods.

Relationship Extraction for BubbleNet: In BubbleNet, we primarily limited the scope of our system to untyped semantic associations. This means that we do not distinguish between different types of relationships, such as synonymy, generalization/specialization and etc. We consider the notion of 'relationship' as a general concept that indicates a relationship (of any kind) between two entities. To extract these kinds of relationships, we use traditional co-occurrence method to extract relationships between pairs of entities appearing in the same document. However, different relationship extractor modules can be plugged into the system to improve its performance, especially in more specific domains. For example, pre-constructed medical ontologies can be used to score relationships between medical entities in addition to the traditional co-occurrence method.

3.10 User Interfaces for Information Retrieval Systems

As we discussed in Chapter 1, BubbleNet is a system for helping users acquire an image of contents of a website and navigate through its documents more efficiently. This is what the user interface of an information retrieval system does. There have been various ways of representing information proposed. Here we discuss some of them and explain their differences with our work.

3.10.1 Query-based Search

A very common way for retrieving documents is using query-based search. The user enters a query, which consists of words, phrases and operators, and the IR system retrieves documents matching entered search criteria. Almost all websites or text databases that contain a large number of documents provide search features. They often allow users to specify dates, types of documents and fields to be searched to help users get better results.

In addition to customizable search criteria, techniques discussed in sections 3.6 and 3.7 can be used to improve search results. While search engines are general

purpose tools for retrieving documents from the World Wide Web, search features embedded in websites provide search facilities limited to the domain of that particular website.

Figure 3.4 shows BBC search page when using the word ‘obama’ as a query.

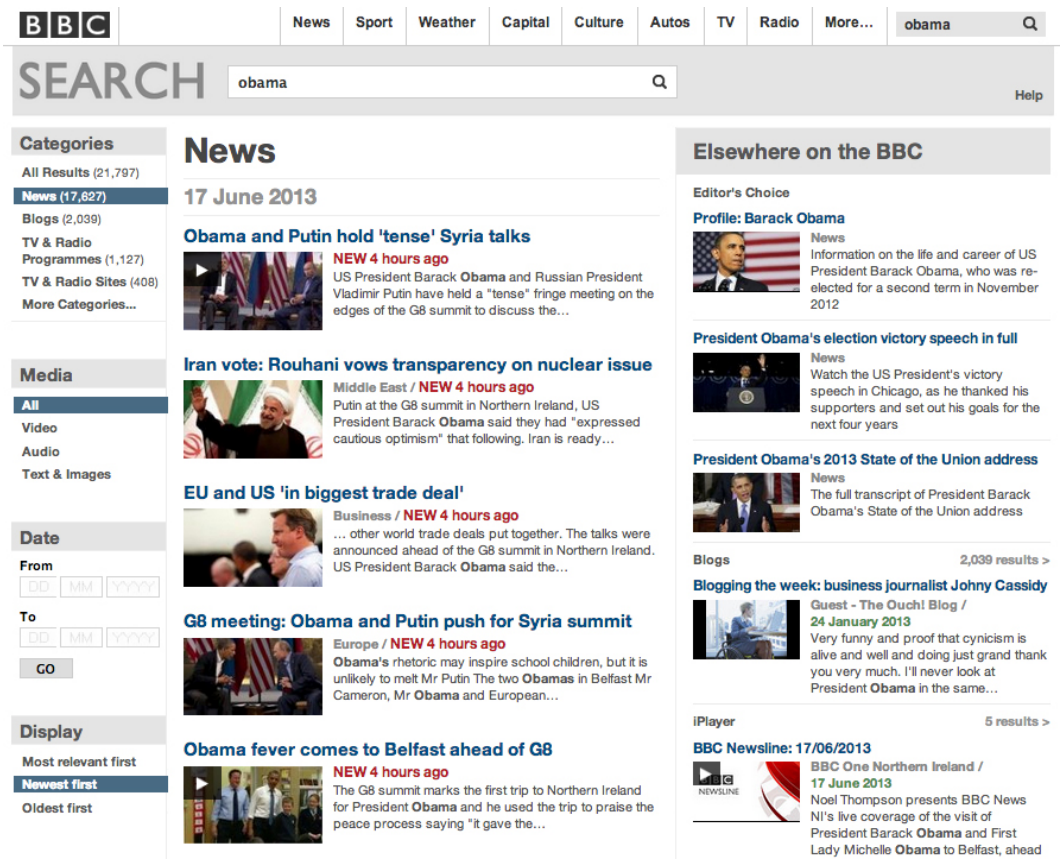

Figure 3.4: BBC search results for query ‘obama’. Features for customizing search results such as dates, ranking order and search scope are provided on the left. (Captured on June 17, 2013.)

Limitations: Query-based search is only a way to retrieve documents. It is useful when the user has a good intuition of what he or she is looking for, and can explain it in terms of a textual query. Although a good search engine can help the user find better words to form more accurate queries, this method is not useful when the user does not have exact words in mind to express his or her desired topic. This is also true when the user is not aware of the exact content of the document collection such as in a forum. This method also does not allow users to interact with the system except by changing and resubmitting queries based on returned results.

Another limitation is that this method does not give users a picture of main topics mentioned in the corpus, so they have to spend a lot of time exploring documents to get a summary of the contents of the website.

3.10.2 Hierarchical Directories

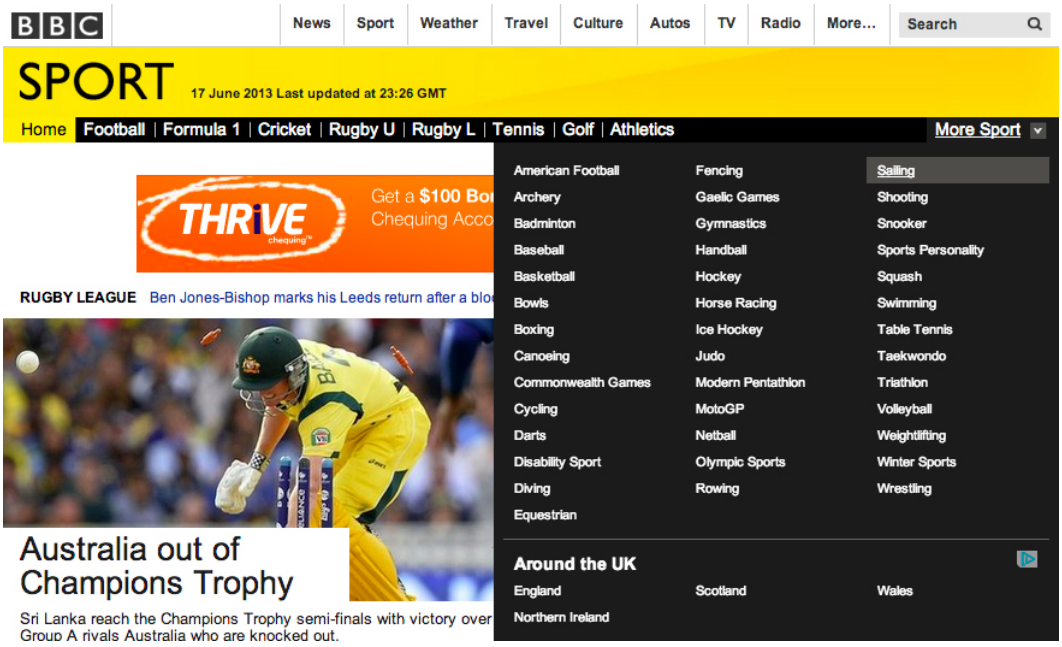

Figure 3.5: BBC categorized menu for sports. (Captured on June 17, 2013.)

Another way to represent and retrieve documents in a corpus is using hierarchical directories. In this approach, documents are clustered hierarchically and a directory of topics is shown on the website. Users can browse this hierarchy to reach a particular document. Many websites partially provide this features. For example, BBC shows a menu bar with several topics that users can choose and filter documents based on their topics.

Advantages: This method has many advantages. First, it represents a big picture of topics in the corpus in an abstract form. Users can get a picture of topics by looking at categories, and they can interact with the system by expanding categories and going down from more general to more specific levels. They also can find desired documents without the need of expressing their meaning in terms of a query, as they can follow the hierarchy to find what they want.

Figure 3.6: A screenshot of Open Directory Project under Regional/Middle East/Iran. (Captured on June 17, 2013.)

Figure 3.5 shows BBC categories for sports. A screenshot of a general directory of websites, called 'open directory project'[1] is also shown in Figure 3.6.

Limitations: To build a hierarchy of documents, IR systems use document clustering. Thus, hierarchical directories do suffer from the same problems that document clusters do (see Section 3.5). First, each document falls into one category, and therefore, it is difficult to cover different aspects of documents. Furthermore, relations between topics discussed in documents are not provided to the user in a browsable way, except the generalization-specialization links of the hierarchy. This prevents users from freely exploring relevant topics regardless of the structure of the hierarchy.

3.10.3 Word Clouds

Word clouds have been growing in popularity and are often used by websites to provide a means for summarizing and navigating through website contents. A word cloud visualizes statistics of term usage in a text or a corpus. The more frequently

[1] http://www.dmoz.org/

a word is used in a corpus, the larger that word appears in the cloud. Figure 3.7 shows a word cloud generated from the first part of Chapter 1 of this dissertation using ToCloud[2], a free tag cloud generator tool.

Figure 3.7: A word cloud generated from the first part of Chapter 1 of this dissertation using ToCloud.

Word clouds can also be constructed using folksonomies: tags that users assign to documents. By using tags instead of words, a tag cloud can represent the most important topics discussed in a corpus.

Scientists tried to improve tag cloud representation. In [45] the authors provide a method for selecting appropriate tags, clustering them and arranging them in alphabetic order. An assessment of different methods of tag cloud visualization was performed in [46].

Clustering tags was performed by [3] to help users find related tags around a particular tag. In their interface, tags are shown in clusters and users can add relevant tags to their queries. New tags are combined with the current query using AND operator. This way, users can build their queries incrementally by choosing relevant tags to what they have entered so far.

[2]http://www.tocloud.com/

Advantages: An advantage of word cloud is that it gives the user a broad perspective of important topics discussed in a document or a corpus of documents in just one glance, as a way of summarizing its contents. Another advantage is that user can decide which word to click by looking at the list of words, not by having query words beforehand. By clicking a word, an IR system can provide the user with a list of documents containing that topic.

Limitations: Primarily, a word cloud does not say anything about relationships between important topics. Words are ordered alphabetically or other esthetic-based ordering and there is no clue about possible relationships between entities. There are variations of word cloud that tried to solve this problem by arranging relevant words together.

Another limitation is that a word cloud generates a flat view of topics and does not provide any means for interacting with users. Thus, a user can only look at the list of words and choose a topic to retrieve relevant documents. As the space is limited in a flat representation, this interface can not provide an efficient way of information presentation.

Furthermore, it is important to select appropriate terms to be counted to build a word cloud. If using all words in a document, there will be several unimportant words listed in word clouds. On the other hand, if using manually-assigned tags, there will be a need for human experts to tag documents. Word cloud, by itself, is not able to extract important topics from documents.

3.10.4 Topic Graphs

A similar approach to BubbleNet is topic graphs. As introduced in Section 1.1.1, The Economist uses this approach to represent most commented entities. They extract named entities and keyphrases from user comments and provide a graphical view of them according to their frequency. They also draw links between entities representing their co-occurrence rate (see Figure 1.2).

Advantages: Representing the most commented topics in the form of bubbles is a great way to give a picture of topics discussed in a corpus to the user. Links are also shown to represent relationships between entities, which makes this way of

representation very effective. The user is also able to click on entities to get a list of comments referring to that topic.

Limitations: This interface lacks some features. First, it does not capture all topics in documents. A more general approach should extract keywords and keyphrases from documents. Furthermore, this interface does not allow users to retrieve documents based on combinations of topics, since the only way of accessing documents is by clicking on entities.

3.10.5 SKIMMR: Machine-Aided Skim-Reading

Published in March 2013, SKIMMR is a tool for summarizing and retrieving documents that uses methods very similar to ours [47]. It automatically extracts entities from a set of documents using natural language tools and provides a web-based interface to an interconnected graph of the extracted entities.

Specially for medical texts, they use a text-mining tool for extracting named entities. They use a co-occurrence method for extracting relationships, as well as corpus-wide analysis of similarities. This is very similar to our approach to summarizing entities and forming the network of BubbleNet.

It was interesting to us that other people are also working on the same idea as it means this is a hot topic in information retrieval. However, there are differences between our work and SKIMMR. First, BubbleNet is able to provide an overview of the entire corpus by showing the most important entities and their significant relationships at the first page and let the user navigate through the entities, while users can access to SKIMMR's graph only by entering a query. Second, they only extract named entities using domain-specific tools while we use a general-domain approach and extract keyphrases and topical words as well to better represent contents of documents. Since we extract relationships based on individual documents and then aggregate relationships, our system is capable of being incrementally updated, while SKIMMR needs the entire corpus to evaluate corpus-wide relationships. And finally, our method provides a feature to limit the search to specific time intervals.

3.11 BubbleNet: Keeping Advantages and Removing Limitations

In BubbleNet, we aim to provide an interface that has the advantages of all the different interfaces mentioned above as much as possible, while not suffering most of their limitations.

- BubbleNet aims to represent the most important topics and their relationships in the form of interconnected bubbles. This allows users to get an overview of the contents of a corpus at first glance.
- Topics are extracted from documents automatically. Thus, there is no need for training data or human-assigned tags.
- BubbleNet is general and can be used for all kinds of documents. However, it can easily be specialized for a particular domain by plugging pre-constructed ontologies to the system for entity extraction and relationship extraction.
- BubbleNet shows bubbles in different levels, i.e. users can start from the more important topics and drill down to more specific ones. This looks like a hierarchy, but it is different since topics are not subsumed by others, but they are semantically related in different levels of importance.
- BubbleNet allows users to interact with the system and browse topics. This way, a user can start from a topic and end up browsing with a totally different topic.
- Users can access documents by clicking on entities or links between them. This way, they can retrieve documents related to an individual entity or combinations of entities.

Chapter 4

A Small Experiment: Navigation Through Health Discussion Forums

4.1 Navigating Health Discussion Forums

The idea of this thesis was originated from a course project we have done before: An Innovative Way for Exploring Health Discussion Forums using Medical Ontologies. This project was done with cooperation of Afsaneh Esteki, a gradute student at the University of Alberta. In this chapter, we explain this project and how it helped us develop the idea of this thesis.

Online health discussion forums contain a large amount of valuable user-generated content, which can be used as a useful resource to collect information from. There are different entities and relations in forum discussions. For example several diseases might share similar symptoms or several names and expressions may refer to a particular disease, symptom or treatment.

Finding a particular discussion that is relevant to a user's query might be difficult in large discussion forums. We introduce an innovative method for navigating through discussions: A visual version of the ontology of medical entities used in discussions will help users to see a high-level summary of information in discussions and find the topic they are interested in.

There are pre-constructed ontologies for medical domain, such as MeSH (Medical Subject Headings) [48]. MeSH is the National Library of Medicine's controlled vocabulary thesaurus that is used for indexing articles for PubMed, and consists of sets of descriptors in a hierarchical structure.

For our application, we need a specialized version of ontology for a particular discussion forum. To this end, we first use MeSH to recognize medical entities. The relations are then extracted using different methods to form the ontology. At the end, we explain how to use this ontology to provide an efficient interface for navigating through discussions.

Researchers have used health discussion forums for information extraction. [49] used sentence-level shallow information extraction to extract medical case descriptions from forums. For this purpose two important sets of features are used: semantically generalized terms and forum structure features. [50] use patterns of keywords to detect semantic relations in National Library of Medicine (MEDLINE) article titles. In order to generate relation extraction rules, they use MeSH descriptors associated with articles and the co-occurrence of terms. [51] try to find question answer pairs from online forums. They use a sequential patterns based classification method in order to detect questions in a forum thread, and also a graph based propagation method for detecting answers for questions in the same thread. A combination of feature based and graph based approaches has been used in [52] for automatically generating medical ontologies from structured Wikipedia. [53] use a decision tree based classifier for information extraction from medical records in resources such as WordNet and UMLS. They try to extract past medical history and social behaviour from the records. [54] use online health forums such as Yahoo Health and Wellness Group for drugs safety investigation purposes. They try to identify those drugs which are likely similar to watchlist and withdrawn drugs by using multiple machine learning classifiers.

4.2 Dataset

Our dataset consists of two main parts: health discussions and medical entities. Here we explained how we collected these two sets.

Health Discussions. We used three health discussion forums with populated

Table 4.1: Number of discussions from each forum

Forum	# files	Percent
eHealthForum	10,096	20.4 %
Healthboards A	7,141	14.5 %
MedHelp B	32,143	65.1 %
total	49,380	100 %

communities: Healthboards[1], eHealthForum[2] and MedHelp[3]. A crawler program was used to download discussions from those forums. Storing one complete discussion in a file, we gathered total of 49,390 files (almost 200 MB). Table 4.1 shows number of files extracted from each forum. We did our best to keep our dataset biased in terms of categories, so we collected discussions from all categories within each forum.

Medical Entities. In order to identify a significant and suitable set of medical entities, we have used the MeSH We first extracted the set of MeSH main headings from the MeSH descriptors and in order to reduce the number of terms, we have only used those that are in the MeSH categories [A] Anatomy, [C] Diseases, and [E] Analytical, Diagnostic and Therapeutic Techniques and Equipment [55]. This set of terms constitutes a bag of words of about 6,000 terms. For the purpose of our experiment we have only used singleton terms, terms formed by one only word.

4.3 The Experiment Method

4.3.1 Pattern Matching

A common method for extracting relations between entities is using pattern matching. A pattern is a sequence of words and wildcards (which we call them *slots*) that represent a certain relationship. For example, *X is a type of Y* is a pattern representing the generalization relationship between X and Y.

We first reviewed some of discussions to find some frequent patterns people use to express relations between entities in health forums. Then, we wrote some patterns manually. They can be categorized in 4 groups. Table 4.2 shows some of

[1] http://www.healthboards.com/
[2] http://www.http://ehealthforum.com/
[3] http://www.medhelp.org/forums/list/

Table 4.2: Examples of manually-written patterns

Category	Pattern
Treatments	X is cured by Y X is used in treatment of Y
Causes	X is a result of Y X causes Y
Symptoms	X symptoms include Y signs of X are Y
Association	X is a type of Y X is associated with Y

those manually-written patterns.

We tried to design a flexible pattern matching algorithm in order to catch as many related sentences as possible. At the time, we considered a scoring method to be able to rank match instances so as to increase precision. Here we explained pattern matching algorithm in details.

Pattern Matching Algorithm. Having a set of patterns in hand, we iterate on sentences in sentence repository and try to match every sentence with all patterns. If a pair (pattern, sentence) has a matching score higher than some threshold, we keep it as a **match instance**.

Matching a pattern to a sentence consists of two steps: First, we map tokens of the pattern to tokens of the sentence. This mapping is then represented in the form of two words that each character corresponds to a token (if not matched) or a pair of matched tokens. The second step is to calculate matching score using edit distance of the two words.

Consider the following example:

- **Pattern:** X causes Y
- **Sentence:** Depression can cause headache

First we want to map tokens of the pattern to tokens of the sentence. For this purpose, we use Levenshtein edit distance[56] to estimate how similar two words are. Then we map each word of the pattern to the most similar word of the sentence. Slots, on the other hand, can match any word, but they prefer medical entities that are listed in our dataset. For the above example, we have the following mapping:

$\{X \rightarrow Depression, null \rightarrow can, causes \rightarrow cause, Y \rightarrow headache\}$

A character is assigned to each character (or pair of characters) so that the mapping will look like $\{acd \rightarrow abcd\}$. The better the sentence matches the pattern, the more these two words look similar. Thus, using edit distance, we can estimate the matching score for the given pattern-sentence pair.

In many cases, the sentence is longer than provided patterns: People usually write long sentences to explain what they mean. Those sentences may contain several information chunks that can be caught by different patterns. To address this problem, we slide a window of the same length of the pattern over the sentence to find the position in which the pattern fits best.

A match instance consists of a pattern, a sentence, their mapping and a matching score. Using a threshold, we can keep only match instances that are highly scored. Match instances are stored in a repository. Every time we look for related entities to a particular entity, we can find match instances in which the entity we are interested in is mapped to a wildcard. The word matching the other slot is then considered as target of the relation.

4.3.2 Co-occurrence Matrix

Another way to find relations between entities is to look for their co-occurrences. The idea is that if two entities are strongly related, they are likely to occur in many sentences together. Thus, we can estimate relations using a measure of how many times two entities co-occur in sentences of our dataset. For this purpose, we form a co-occurrence matrix using the following strategy:

We iterate over the sentence repository. For each sentence, we look for all entities occurred in that sentence. Then, for each pair of entities appeared in the sentence (which is corresponding to an entry of co-occurrence matrix), we create a *co-occurrence instance* and calculate a co-occurrence score. We then accumulate all those scores and also keep up to 5 most highly-scored co-occurrence instances for each entry.

In order to calculate co-occurrence score, we use the following heuristics:

- The closer two entities, the higher the score the co-occurrence instance gets.

The idea is that if two entities appear in two ends of a long sentence far away from each other, they are less likely to be related rather than the case they are close.

- The shorter the sentence, the higher the score it gets. This is because short sentences are more probable to represent coherent relationships between entities than longer ones.

- We collected a set of words that are frequently used for representing relations. Some examples are *associated, related, causes, ...* We call them *relational words*. The more relational words a sentence have, the more score it gets.

To find entities related to a particular word, we simply scan corresponding row in the matrix and find the column in which their intersection gets maximized. Note that this relation is not symmetric. That is, the co-occurrence matrix is not symmetric because the order in which the two entities appear in a sentence is important. For example, *X causes Y* and *Y causes X* are different relationships.

Figure 4.1 shows a simplified version of similarity matrix.

4.3.3 Distribution-Based Method

The idea of this method is that if two words are similar (or relevant), they tend to have similar distribution of co-occurrences with other words. Thus, for each entity, we looked for a word that has the most similar distribution of co-occurrence scores.

Despite simple co-occurrence method, distribution-based similarity is a symmetric relationship. Therefore, we first add a transposed version of co-occurrence matrix to it to make it symmetric. We then normalize values of each row with respect to its maximum so that in each row values vary between 0 and 1.

The next step is to subtract a row from another. By calculating sum of squares of the components of resulting vector, we can measure how similar or different those two rows are. This way, it is possible to find the most similar row for each row of the matrix, which is equivalent to finding the most similar word to another one.

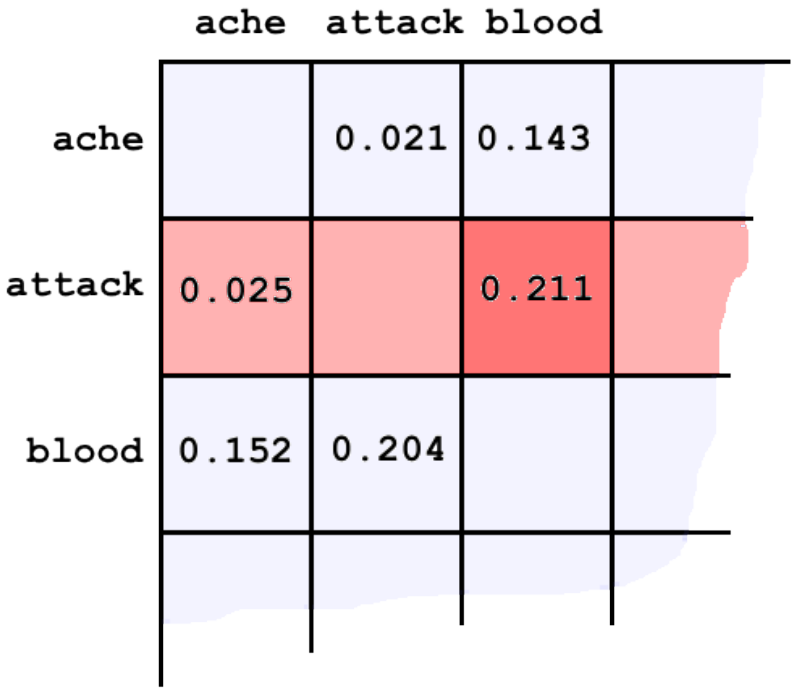

Figure 4.1: A simplified version of similarity matrix.

4.4 User Interface

We have implemented a user interface for our extracted medical ontology. For this purpose we have used Swing, the Java GUI widget toolkit. A user can enter a medical keyword, for example the name of a disease or a medicine, and by clicking on a button we suggest to him/her a set of related words. Since we consider all discussions from 3 forums, a large set of terms is returned as the related words. However we only show ten first related words, which have higher scores. Figure 4.2 depicts an example of the interface. As figure shows when user enters "vomiting" keyword, our system returns a set of related words. These words might be some body organs such as "stomach", a drug like "promethazine", a symptom such as "dizziness" or another similar word for the keyword, like "anorexia". Also, the user can see the relationship between two words. Relations are returned based on any of the three methods; co-occurrence, patterns or similarity in distribution. For example, the figure shows that "vomiting causes dizziness", and at the same time it is in relation with "promethazine" based on the similarity in distance. There is also

an option of reviewing a random medical entity and its related words.

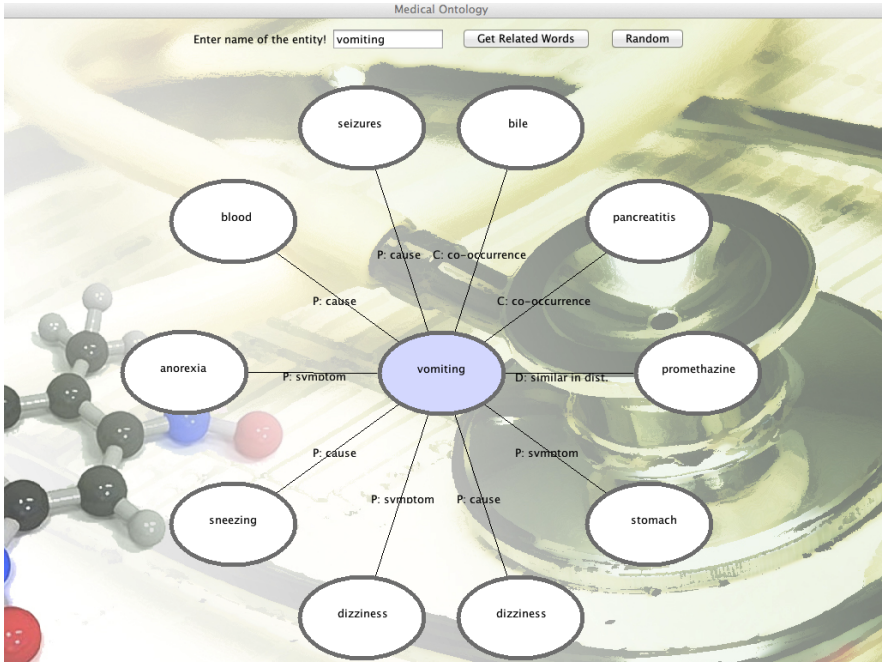

Figure 4.2: User Interface

4.5 Results and Discussions

Running our algorithms on almost 50,000 discussions and 6,000 medical entities, it took an hour to calculate similarity matrix and half an hour to extract pattern matches. Keeping only matches with scores higher than 0.5, we got 4456 match instances. Table 4.3 lists some of match instances extracted from our dataset. Entities that have been mapped to slots are bold.

Applying the other two methods (co-occurrence matrix and distribution-based) also gave us pairs of entities. Table 4 lists some instances of results of co-occurrence and distribution- based methods.

Evaluating our method was a serious challenge. The problem is we do not have any ground truth. We are not looking for scientifically true relationships between entities; thus, we can not use medical information to evaluate results. In addition, we do not have a set of discussions with extracted ontology to calculate a recall.

Table 4.3: Examples of pattern match instances (results of running pattern-matching algorithm)

Pattern	Sentence
X causes **Y**	Could a 4mm **prolactinoma** cause **tinnitus** and nausea
X causes **Y**	can drinking soy **milk** cause your **hair** to fall out
X causes **Y**	Severe **hypotension** caused by **shock** is a medical em...
X is related to **Y**	...cervical **lordosis** possibly related to muscle **spasm**
X is related to **Y**	I believe the back **pain** is related to **prostate** conditions
symptoms of **X** are **Y**	If you are having symptoms of **meningitis** (stiff **neck**...
X symptoms include **Y**	**kidney** stone symptoms, abdominal **pain**, ...
symptoms of **X** are **Y**	The most significant symptom of **fibromyalgia** is **pain**

Table 4.4: Examples of results of co-occurrence (left) and distribution-based (right) algorithms

Entity 1	Entity 2	Entity 1	Entity 2
calcium	zinc	gastritis	esophagus
blood	plasma	eczema	scabies
mercury	poisoning	uveitis	sclera
biopsy	fibrosis	mercury	arsenic
abdomen	pelvis	foot	knee

Even if we had such an ontology, it was difficult to evaluate our work because the notion of relations between entities is not a true/false or presence/absence matter.

In order to estimate how efficient and accurate our method works, we used the following method: Given the set of found pattern matches and the set of pairs generated by co- occurrence and distributed-based algorithms, we picked 50 instances of each set of results randomly. Then, we asked a friend (who is neither computer scientist or physician) to evaluate them.

For pattern matches, there are two notions to be evaluated: First, we need to see if our method catches the correct relations. For example, if an *X causes Y* pattern is matched with an *X cures Y*, the pattern is not consistent with the real nature of relationship that the sentence is representing. Second, the algorithm is supposed to match right entities with slots. Thus, we have two percentages reported for pattern matching algorithm.

Co-occurrence and distribution-based methods are evaluated in a different manner: we roughly marked a pair of words as 'correct' if the two words were related

Table 4.5: Evaluation Results

Method	Evaluation Result
Pattern Matching	80%, 63%
Co-occurrence	70%
Distribution-based	68%

(ex. symptoms of the same disease, organs of human body, drugs of the same category, etc.) and 'incorrect' otherwise. Thus, we have only one percentage reported for each algorithm.

Table 4.5 lists the evaluation results.

Pattern matching algorithm is a simple method in terms of ideas, although its implementation was not totally straightforward. Using pattern matching, one can extract specific relations with known meanings and directions. But it needs an appropriate level of domain knowledge. The results of evaluating our method show that pattern matching has a good performance on the task of finding similar relationships. However, we do not know the recall, that is, how much of existing relations are extracted using this method. The other weakness is its low performance in finding entities that are related. In one third of evaluated instances, our algorithm mapped slots to incorrect entities. But this problem can be addressed effectively by adding more natural language processing ideas to the algorithm.

In co-occurrence method, the main difference is that we do not know the exact semantic of relations anymore. We can only say that 'there is a relationship, but we do not know what it is'. This is because several different relations can result in the same effect: high frequency of co-occurrence. Therefore, this method may not work well when we need to know the nature of relations or we are looking for specific relations between entities. But it can be combined with other methods, such as pattern matching, to find out the type of relationship once it was found.

Co-occurrence method is also helpful when we do not care what type the relationships are. An example is query expansion for improving performance of information retrieval systems, where we do not worry about the semantic of relationships and we only need to know words that often appear close to a particular entity.

Since co-occurrence method considers all co-occurrences of a pair of entities,

the only thing we need to do is to define a wise and accurate scoring strategy to rank co-occurrence instances. Once we came up with such a strategy, we can easily find related words to a particular word by scanning the matrix.

This method is also ideal for real time systems in which we want to update the system every time new data arrives. The idea is to simply scan new discussions for probable co- occurrences of existing entities and for each co-occurrence, update the corresponding entry in co-occurrence matrix.

Distribution-based method is similar to co-occurrence to some extent in terms of problem set up and implementation issues. In our experiments, it performed worse than other methods since it is not as precise as the others, but it was still much better that what we expected. But its performance grows significantly as the size of data set grows because the greater data set we have, the more accurate and meaningful co-occurrence distributions are. Thus, it can be kept as a useful tool besides a co-occurrence based ontology extraction system.

4.6 Towards BubbleNet

After finishing this project, we realized that this interface, regardless of domain-specific entity and relation extraction methods, can be useful for navigating web-sites. By generalizing the idea of the project explained in this chapter to all domains and document formats, we ended up in BubbleNet: a tool for extracting key entities and relationships from all kinds of texts and providing a graph-based user interface for helping users navigate websites and retrieve documents. Thus, we decided to use general keyword extraction and named entity recognition methods, as well as general relation extraction so that the system can be used for any website in any domain. At the time, we kept the idea of using ontologies in mind so that our system is designed to be able to use domain-specific ontologies, when available, to improve results in specific domains.

Chapter 5

The System Model and Design

In this chapter we give an implementation-based overview of BubbleNet.

5.1 Objects

The objects that our system deals with are *documents*, *entities*, *relationships*.

5.1.1 Document

A document is a piece of text that is given as input. This can be a blog post, a news article, a user-made comment, a post in a discussion forum, an entire discussion in a discussion forum or any other kind of text grabbed from a corpus.

The notion of "document" is important because it is defined according to the needs of an application. For example, if BubbleNet is going to summarize and visualize contents of a news website, each news article could be considered as a document. In contrast, if we are interested in building a BubbleNet from the set of comments that users made on an individual article, a document should be equivalent to an individual comment. Similarly, we can call each post in a discussion a *document* to build a BubbleNet summarizing an entire individual discussion, or we can divide discussions to single posts and build the BubbleNet from the set of posts coming from a discussion. In summary, a document is the smallest individual piece of text that we deal with.

Each document can possess several data fields, such as title, author's name, document type, category, page number and volume, etc. In BubbleNet, we limit

these fields to contents (body), date, address (URL or file path where a document is available at) and an automatically assigned document ID.

Figure 5.1: An abstract model of a document

5.1.2 Entity

An entity is a word or phrase extracted from or assigned to a document, representing a topic, concept, object, person or any other real world entity that is mentioned in that document.

Entities are used in summarizing documents, i.e. we use entities as representatives of topics discussed in a document. For example, a document concerning the budget cuts of the University of Alberta might be represented by extracting entities like U of A, budget cut, financials, tuition fees, research funds and so on.

Entities can be single words or multi-word phrases. They can be extracted from the text itself, or assigned to a text using a topic modeling or tagging algorithm.

As entities are derived from documents, they can be assigned the fields of a document they are derived from, including document ID, date and score. An entity may have a *score* that shows how important that entity is in a document or a corpus. Importance here refers to the amount of information that an entity carries about the topics discussed in the document. If the entity is extracted from a document, its score means its importance in that document. If an entity is representing a topic for a set of documents, its score means an overall importance of that entity across all documents in that set.

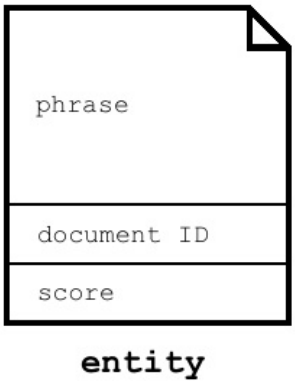

Figure 5.2: An abstract model of an entity

5.1.3 Relationship

Relationships indicate how semantically relevant two entities are. This relevance has to be defined according to the application and is estimated by relationship extraction modules. Generally, relationships may engage more than two entities, but in BubbleNet, we limit relationships to two entities.

Like entities, relationships also possess scores. The score of a relationship indicates how strong the two entities are related (according to the definition of relevancy in that application). Relationships may also be assigned a date and document ID fields similar to entities.

In general, relationships are not necessarily symmetric, that is, if A is related to B, B is not necessarily related to A in the same way and with the same strength (or score). In BubbleNet, however, we consider relationships symmetric. This way, if there is a relationship with a particular strength between A and B, the order of A and B is not important.

5.2 System Components

Now we provide a high-level model of BubbleNet system, indicating the system components and their functionalities.

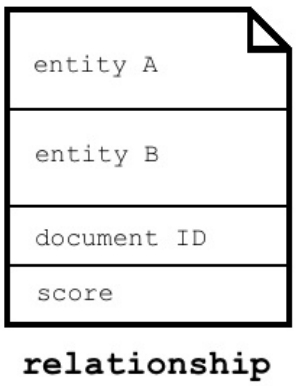

Figure 5.3: An abstract model of a relationship

5.2.1 Document Loader

The journey of documents through BubbleNet starts in the Document Loader. In a practical setting, we may have a number of new documents arriving daily (or even hourly). The Document Loader is responsible for dealing with different document formats, as well as providing an interface to the back end user (e.g. the system administrator) to add documents to BubbleNet system.

The input for Document Loader is a set of documents and a signal from the system maintainer to start adding new documents, and the output is a set of document objects. Note that the entity objects generated by this component are representatives of a single document and are not representing topics among all documents in the corpus.

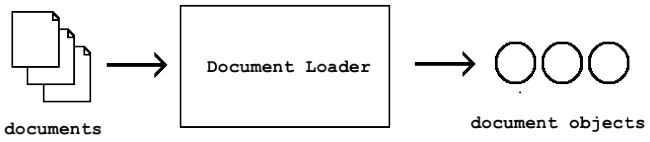

Figure 5.4: Components: Document Loader

5.2.2 Entity Extractor

BubbleNet is flexible to use different entity extraction and relationship extraction methods. The purpose of this component at a high level is to extract a set of scored entities from a given document using an appropriate algorithm. We implemented different concrete modules extending this abstract interface for different extraction methods.

The input of this component is a document object, and the output is a list of entity objects extracted from (or assigned to) that document. The entities are scored according to their importance, and the extraction algorithm thus should provide a way for scoring entities.

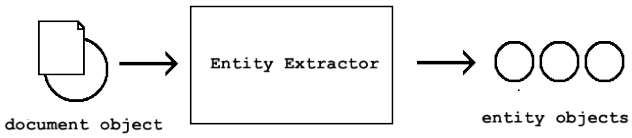

Figure 5.5: Components: Entity Extractor

5.2.3 Relationship Extractor

Similar to the Entity Extractor, this module has responsibility to extract relationships between pairs of entities in a document. The 'extraction' of relationships means 'estimating their strengths' as we assume that the entities are already extracted and given. Thus, this module scores relationships between all the pairs of entities for a given document.

The input for Relationship Extractor is a document and the set of entity objects extracted from that document. The output is a set of scored relationship objects. Note that the relationship objects generated by this component are representatives of a single document and are not representing topic similarities or relationships among all documents in the corpus.

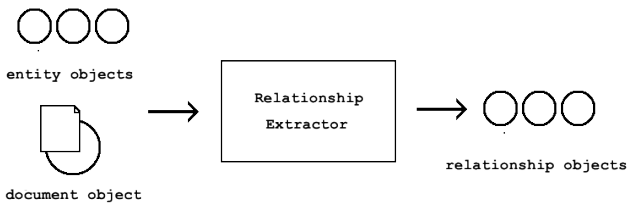

Figure 5.6: Components: Relationship Extractor

5.2.4 Database

Documents, entities and relationships are all stored in a database. The database stores entity and relationship objects extracted from different documents as separate objects even if their words or phrases are the same, so that we can query the database for entities and relationships belonging to a particular time span. More detail of the database and its schema can be found in Section D.4.

5.2.5 User Experience Scenario

BubbleNet is designed to provide an effective user experience for exploring entities and relationships extracted from documents. Below we describe some user experience scenarios:

- **Overview:** A big picture of the contents of a corpus is given by showing the most important entities and relationships extracted from all of the documents within a particular time interval.

- **Exploring Bubbles:** The user is able to explore a given BubbleNet. Clicking on a bubble expands the entity, revealing related entities around the clicked one. The user can click other bubbles to make them centre and further expand them. The user can also enter a word or phrase to look for the matching bubble immediately. The result can be restricted to documents within a particular time interval.

- **Retrieving Documents:** When a user is interested in studying documents related to a bubble or a link, he or she can simply hover the mouse over the bubble (when it is centered and expanded) or the relationship, and see a list

of related documents and click the one that looks interesting. This action can also be restricted to documents within a particular time interval.

5.2.6 User Interface

This module plays the role of an interface between the end user and the database to make the user experience explained in the previous section happen.

This module can be divided into two main parts: database interface and visualization . The database interface has to submit the following queries into the database and prepare well-formatted results for the visualization part:

- Given a time interval, return the k most important entities and the most strong relationships between them.
- Given an entity and a time interval, return the k entities that are most strongly related to this entity.
- Given an entity or a pair of entities and a time interval, return a list of documents that are most relevant to that entity or relationship. (This means that the given entity or relationship has the highest score in those documents within the specified time interval).
- Given a document ID, retrieve the document address and contents.

The visualization component sends requests to the database interface and visualizes the obtained results for the user.

5.3 System Architecture

Figure 5.7 shows the connection between the components. As a new document arrives, it is loaded by Document Loader and is stored in a document object. Document objects are then passed to the Entity Extractor component, which produces entity objects extracted from documents. A document object is then passed to the Relationship Extractor component, followed by a set of extracted entity objects. The result is a set of relationship objects. Then, all the objects (documents, entities

and relationships) are stored in the database. The User Interface component helps user access the database and get a visualized presentation.

Different implementations of those components can be plugged into this architecture. For example, different document loaders for different document formats are implemented. Also, if there is a pre-constructed ontology for a particular domain, it can be used in an implementation of entity and relationship extraction components.

In the next chapter, we explain how we implemented BubbleNet system based on the model we provided in this chapter.

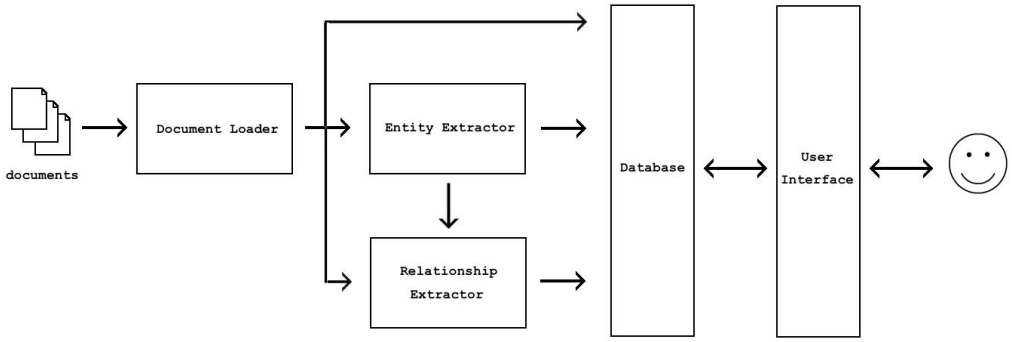

Figure 5.7: The Architecture of BubbleNet System

Chapter 6

Constructing BubbleNet

6.1 Extracting Entities and Relationships

BubbleNet is constructed based on entities that represent topics of documents and their similarity relationships. Since we want BubbleNet to be useful for a general domain, we consider domain-independent entity and relationship extractors. The method used for extracting entities and relationships for a general domain is explained in sections 6.1.1 and 6.1.2.

In addition, we wanted to investigate if using a domain-specific ontology helps BubbleNet . To this end, we report some of our experiments with an entity extractor that uses MeSH to extract medical entities and a relationship extractor that uses WordNet to better estimate the similarities of extracted entities.

6.1.1 Entity Extraction Using Alchemy

In order to extract entities from a document, we use keyword extraction as well as named entity recognition. There exist several tools for extracting keywords and named entities, such as Yahoo! Content Analysis[1], Kea[2], Calais[3], Stanford Named Entity Recognizer[4], OpenNLP[5] and AlchemyAPI[6].

We tried all the tools listed previously to see which one is more appropriate for

[1] http://developer.yahoo.com/contentanalysis/
[2] http://www.nzdl.org/Kea/
[3] http://www.opencalais.com/
[4] http://nlp.stanford.edu/software/CRF- NER.shtml
[5] http://opennlp.apache.org/
[6] http://www.alchemyapi.com/

our application. An ideal tool for BubbleNet should:

- Be fast, since we want BubbleNet to be updated quickly as new documents arrive,
- Extract good keywords and named entities that best represent the contents of documents,
- Not require training documents as BubbleNet has to be independent of dataset characteristics,
- Assign scores to extracted entities so that BubbleNet can recognize the most important ones.

Among the tools that we investigated, AlchemyAPI came on top in our case-based evaluations. AlchemyAPI is a commercial online service for text analysis, including keyphrase extraction, named entity recognition. (Since it is commercial, they have not published the technology behind it.) The free version of this system has a limitation of 1000 transactions per day. We managed to get access to Alchemy for this research for free.

We used both keyphrase extraction and named entity recognition services provided by Alchemy. For each document, we sent its content to the Alchemy server and obtained a list of scored keyphrases and a list of scored named entities. We combined the two lists and normalized the scores so that they are between 0 and 1.

6.1.2 Relationship Extraction based on Co-Occurrence

Following Spence and Owens' work in [40], we estimate the relatedness of two entities based on how frequent they co-occur in documents, as well as how close they appear in a particular document. In other words, every coincidence of two entities in a single document increases the total score of the relationship between the two entities, and the amount of this increase depends on their distance.

To take the distance of the occurrences of two entities into account when scoring their relationship, we define a function called $DistanceBasedScore(DBS)$ that

gives the score of a relationship based on the distance of the first characters of two entities (in characters). Figure 6.1 shows this function:

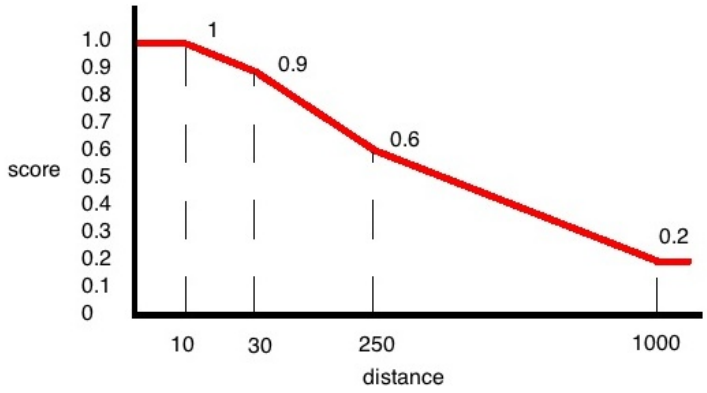

Figure 6.1: Distance-based score function (score based on distance in characters)

The values in this function are set so that it estimates the score similar to what we explained in section 3.9. This function gives the score for a pair of entities. Since entities may appear in a document several times, we accumulate distance-based scores for all pairs of occurrences of two entities to get the total score.

Now, suppose that entity A appears 10 times in document D and entity B appears 4 times in the same document. There will be $4 * 10 = 40$ pairs of occurrences that produce partial scores for the relationship between A and B. But intuitively, we can say that the score of the relationship between a pair of entities should not grow linearly as the number of occurrences of those entities grows. For example, if entity A appears just once and entity B appears 10 times, it should not produce a score two times higher than the score of a relationship in which B appears 5 times. In order to reduce the growth rate of the scoring function, we divide the total score by the square roots of the frequencies of A and B in document D. Given a pair of entities A and B and a document D, the score of their relationship is computed as follows:

- For every occurrence of A and every occurrence of B, calculate a score based on their distance (as described below). Thus, if A appears m times and B appears n times, there will be $m \times n$ scores calculated.

- Sum up all of the scores calculated in the previous step to get the total score for the pair of entities A and B.

- Divide the resulting total score by the geometric mean of the number of occurrences of A and B in D.

This can be summarized as follows: Assume that $O(A) = \{a_i\}$ and $O(B) = \{b_j\}$ are sets of positions of occurrences of entities A and B (measured in number of characters from the beginning of the text) and $dbs(a_i, b_j)$ is their distance-based score. Then the relationship score between A and B in D is calculated as follows:

$$score(A, B) = \frac{\sum_{i,j} dbs(a_i, b_j)}{\sqrt{|O(A)||O(B)|}} \tag{6.1}$$

6.1.3 Entity Extraction Using MeSH

MeSH is a controlled medical vocabulary that contains a large number of categorized medical headings such as body anatomy, disease names, medications, etc. To investigate if such resources can help us improve BubbleNet, we extracted entity names from MeSH, and forced the entity extractor module to use entities in MeSH as a lookup-table to extract entities from documents. That is, if an entry of this vocabulary appears in a document, this module extracts it as an entity.

Since this method recognizes the entities but does not score them, we need to find a way to score extracted entities. For this purpose, we use the intuition that term frequency in a document is a clue to estimate its importance. Thus, we assign the square root of term frequencies to entities as their scores.

6.1.4 Relationship Extraction Using WordNet

Similarly, relationships may be scored using resources such as WordNet. We wanted to investigate how BubbleNet performs if we use knowledge from other resources to estimate how similar two entities are instead of evaluating their similarity based on their co-occurrence in the corpus.

WordNet provides a *distance* measure between two given phrases, which is between 0 and 1 or equals -1 if the phrases do not exist in the ontology. Assuming

that $w(A, B)$ gives this distance, we calculate a relationship score between A and B as follows:

$$score(A, B) = \begin{cases} 0.001 & \text{if } w(A, B) = -1 \\ \frac{1}{w(A,B)+0.001} & \text{otherwise} \end{cases} \tag{6.2}$$

Given a document and a set of extracted entities, we score the relationships between the extracted entities using the function explained above. It is important to mention that this score is the score of the relationship between A and B extracted from document D. The total score of the relationship between A and B is a function of all scores from all documents. Thus, a notion of co-occurrence is taken into account since every co-occurrence of two entities introduces a non-zero partial score in their overall relationship score.

6.2 Constructing the Network

BubbleNet network provides nodes corresponding to entities and edges corresponding to relationships, say within a given particular time span. For this purpose, it has to combine extracted entities with the same phrase from all individual documents. For example, having the entity 'apple' extracted from 10 documents published on 2013-01-12 and 15 documents published on 2013-01-13 (each with a different score), we need to aggregate these scores to end up with a score for the entity 'apple' for the interval 2013-01-12 till 2013-01-13, since they all will be represented as a single node.

There are several functions that we can use in order to aggregate the scores of extracted entities. We experimented with the functions listed in table 6.1.

To compare the functions, we created 5 sets of documents: Each set contained 50 documents randomly chosen from the 5 datasets that we prepared for our experiments (see Section 6.4.1 for complete description of our datasets). For each set, we built four BubbleNet networks corresponding to the functions in Table 6.1. We visualized the networks and captured a snapshot of the first page, showing top-ranked 15 important entities and their significant links (the user interface is explained in Section D.5 in detail). We asked three judges to look at those snapshots and choose

Table 6.1: Different aggregation functions for entity scores

#	Function	Description
1	$\sum s_i$	Scores are summed up.
2	$\sum (s_i \times \log (tf_i))$	Scores are multiplied by log of number of times the entity appears in a document. Then they are summed up.
3	$(\sum s_i) / \log (df_i)$	Scores are summed up and then divided by log of number of documents in which the entity appears (document frequency).
4	$\sum (s_i \times \log (tf_i)) / \log (df_i)$	Scores are multiplied by log of number of times the entity appears in a document. Then they are summed up and divided by log of document frequency.

Table 6.2: Results of comparing aggregation functions by three evaluators

#	Function	Score
1	$\sum s_i$	6
2	$\sum (s_i \times \log (tf_i))$	1
3	$(\sum s_i) / \log (df_i)$	4
4	$\sum (s_i \times \log (tf_i)) / \log (df_i)$	4

one based on the quality of the extracted entities and their connections.

Table 6.2 shows the results. The third column shows how many times the snapshot corresponding to a particular aggregation function was selected as the best one among other functions, summed up for all 5 datasets. As we can see, functions 1, 3 and 4 performed very similarly. However, we chose function 1 as it received the highest score. In all our experiments, the score of a relationship between two entities is the sum of the scores of all relationships between those two entities within the specified time interval.

6.3 Visualization

Our goal in this visualization is to represent the bubbles in a way that the user can understand retrieved information, including entities and their importance, as well as relationships between entities and their strength, in a glance. To this end, we used the idea of simulating masses, springs and forces as a visualization layout for our network.

Using the physics of masses and springs is a great way for visualizing graphs as it has been widely used by others under the name of spring embedder algorithms and force directed graphs[57][58]. We followed this layout with some changes. In this section, we describe how we create, place, link and move the bubbles.

6.3.1 Creating Bubbles

A bubble is a simple circle with a label on it, showing the entity caption. To represent the importance of the entities, we use different radiuses and fill colors. The radius of bubble i is specified as:

$$r_i = 55 \times s'_i + 15 + 1.5 \times len_i \tag{6.3}$$

where s'_i is the normalized score of bubble i among other retrieved bubbles and len_i is the length of the phrase of entity i in characters. The parameters in this formula are chosen by experiments in a way that entity captions fit in the circles.

The colour of a bubble is then selected according to its calculated radius based on a scale that maps radiuses to colors. We used 5 different colors for drawing the bubbles. The font sizes for the bubbles texts are also chosen based on both the bubbles' radiuses and scores, so that the more important an entity is, the larger its font size is.

From a masses-springs perspective, a bubble has a mass, which is proportional to its size. Considering bubbles as discs with equal thicknesses, the mass of a bubble is proportional to r^2.

6.3.2 Links Between Bubbles

A link is a spring between two entities and is shown as a line between the bubbles. The strength of a relationship is represented using both line thickness and length. Stronger relationships are thicker. They are also shorter, resulting in the two bubbles standing closer to each other following the intuition of their relatedness.

The intial length of a spring is specified as:

$$len_{i,j} = 50 \times s'_{i,j} + 20 \tag{6.4}$$

where $len_{i,j}$ is the length of the spring between entities i and j and $s'_{i,j}$ is the normalized score of the corresponding relationship among retrieved relationships.

The initial positions of bubbles are random. The springs always connect the closest points of the two bubbles, thus, their lengths are not necessarily equal to their initial lengths until they reach an equilibrium.

6.3.3 Masses, Springs and Forces Simulation

We simulate physical laws governing masses and springs by calculating forces exerted on the bubbles, their accelerations, velocities and position updates as the time goes by. There are four forces exerted on bubbles:

- **Spring Forces:** Every spring that is connected to a bubble exerts a force on it. According to Hooke's law[7], the magnitude and sign of this force depends on the distance the spring is extended or compressed. The direction of this force is specified by the relative position of the other tail of the spring, which is the other bubble.

$$F_k \propto \Delta x \tag{6.5}$$

- **Repulsion Between Bubbles:** To avoid bubbles having overlap, we consider a repulsion force between bubbles. This force is proportional to the masses of the bubbles and is inversely proportional to their distance.

$$F_r \propto \frac{m_i \times m_j}{d_{i,j}} \tag{6.6}$$

- **Boundary Repulsion:** To keep the bubbles inside a bounding box, we consider four springs connected to the bubbles and the borders; but these springs can freely slide on the borders so that they always exert forces in horizontal or vertical direction on bubbles. Boundary forces are considered to be proportional to bubbles' distances from the borders and their masses:

$$F_b \propto m_i \times d_{i,border} \tag{6.7}$$

[7] http://en.wikipedia.org/wiki/Hooke's_law

- **Friction Force:** If two bubbles are connected by a spring that is not in its initial length, they will fluctuate around an equilibrium point forever. To avoid this, we simulate a friction point that causes the bubbles to gradually lose their velocities. The friction force is proportional to the mass of a bubble as well as its velocity and is always in reverse direction of its velocity:

$$F_f \propto -m_i \times v_i \tag{6.8}$$

Summing up all the forces in a two dimensional space, the interface repeatedly calculates the bubbles' accelerations and updates their velocities and positions to simulate their movements. Figure 6.2 summarizes the forces exerted on bubbles. The parameters in force equations have been set by experiment to get a realistic behaviour.

6.4 Experimental Setup

In this section, we explain our experimental setup, including our dataset preparation and cleaning.

6.4.1 Datasets

We prepared three datasets consisting of three different types of documents:

Reuters News Articles

This dataset consists of 15,000 news articles extracted from the Reuters-21578 dataset[8]. The original dataset consists of 21,578 categorized news articles from Reuters and is widely used for text categorization tasks. We extracted the first 15,000 articles, ignoring all categorization metadata. For each article, we kept the contents as well as its publishing date.

Health Discussion Forums

We introduced our health discussions dataset in section 4.2. We grabbed 5,000 discussions from each of the three discussion forums (HealthBoards, eHealthForum

[8]Available at: http://www.daviddlewis.com/resources/testcollections/reuters21578

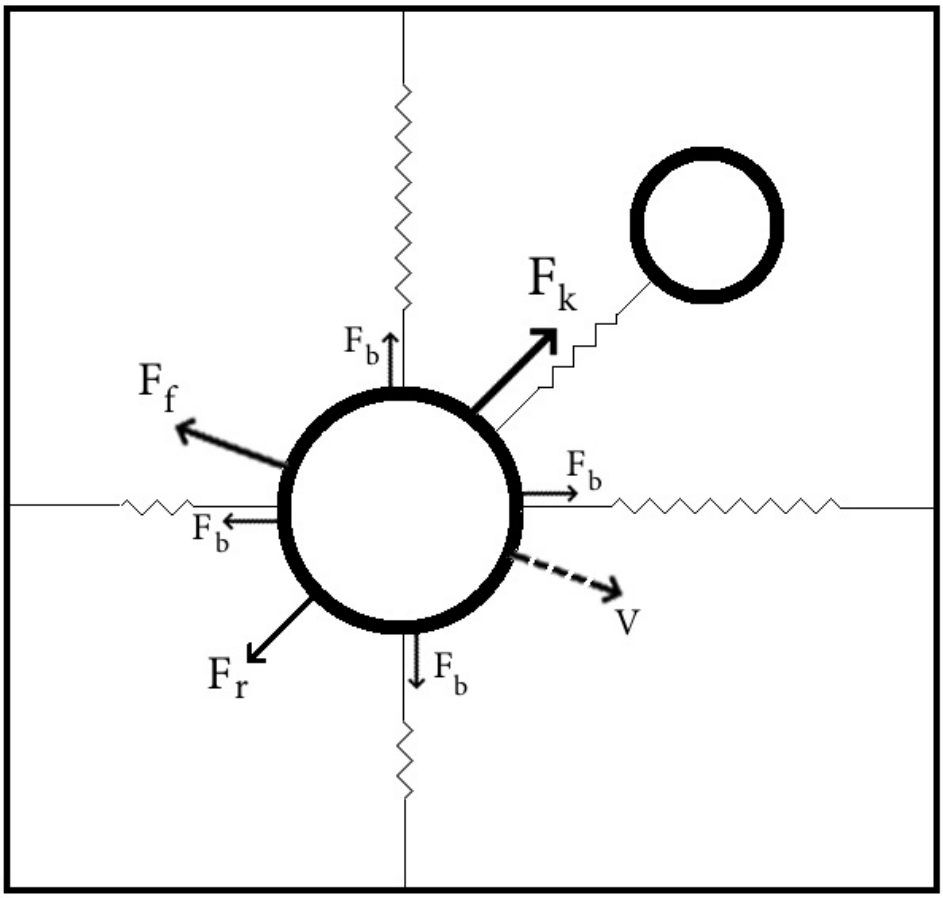

Figure 6.2: Forces exerted on a bubble: Spring force (F_k), repulsion force (F_r), border repulsion forces (F_b) and friction force (F_f). V indicates current velocity vector of the bubble.

and MedHelp), resulting in a dataset with a total of 15,000 discussions. We removed all metadata and considered each discussion (in its entirety) as an individual document.

Dr. Arya's Obesity Blog

Our third dataset is a set of 900 blog posts from a professional blog about obesity. From each post, we kept the body text as well as publishing date.

6.4.2 BubbleNet Configurations

For the Reuters news articles dataset we used Alchemy Entity Extractor and Co-occurrence Relationship Extractor for constructing BubbleNet. For the two other

Table 6.3: Different BubbleNet configurations

Config. Name	Dataset	Entity Extractor	Relation Extractor
Reuters-AC	Reuters News	Alchemy	Co-occurrence
Health-AC	Health Forums	Alchemy	Co-occurrence
Health-MW	Health Forums	MeSH	WordNet
Arya-AC	Obesity Blog Posts	Alchemy	Co-occurrence
Arya-MW	Obesity Blog Posts	MeSH	WordNet

datasets (i.e. health discussion forums and obesity blog posts) we constructed BubbleNet with two different configurations: The first configuration is similar to Reuters dataset using Alchemy Entity Extractor and Co-occurrence Relationship Extractor. But the second configuration uses MeSH Entity Extractor and WordNet Relationship Extractor as we wanted to investigate if using an ontology will help BubbleNet produce better results on medical texts. Table 6.3 summarizes these configurations. We used these five configurations in evaluating our system (see Chapter 7).

6.4.3 Data Cleaning

Since entity extraction methods are not ideal, we have to perform a cleaning step to remove inappropriate entities and relationships.

The first thing to do is to unify different forms of words. We limited this to unifying singular and plural forms. To this end, we used a dictionary of singular English words[9] as a look up table. For every extracted entity, we checked if the phrase ends with s, es or ies. If so, we remove the suffix ($fruits \rightarrow fruit, boxes \rightarrow box, berries \rightarrow berry$) and looked in the dictionary for the resulting phrase. If the entry was in that dictionary, we treated the modified phrase as a singular form of the initial entity.

This process can result in duplicate entities and/or self-referring relationships if both singular and plural forms of a word were originally extracted. We merged duplicates and removed self-referring relationships.

Finally, there are sometimes entities that appear in many documents while they are not very informative, such as the name of the news website or the blog author.

[9]http://www.puzzlers.org/dokuwiki/doku.php? id=solving:wordlists:about:start

Table 6.4: Examples of filtered entities

Config.	**Filtered entities**
Reuters-AC	`pct, dlrs, mln dlrs, reuter`
Health-AC	`doctor, thing, problem, case`
Health-MW	`will, all, problem`

Similar to removing stop words, we can filter these entities by choosing them from the top most frequent entities. The process of choosing these entities consists of: (1) listing all entities orderd by frequency (or score), (2) looking at the most frequent entities, (3) recognizing useless or meaningless entities. Table 6.4 shows some entities that are filtered.

Chapter 7

Evaluation

We indicated two goals for BubbleNet: First, to provide an overview of the content of a website or a set of documents, and second, to help users navigate through documents and find their desired information. In this chapter, we evaluate BubbleNet and report how successful it is in reaching these goals. We also provide an analysis of the backend and the practicality of the system in a real world setting.

7.1 Evaluation Method

We designed a user study to evaluate BubbleNet. In our study, users were invited to complete an online survey, consisting of two information retrieval tasks and a questionnaire. In this section, we explain this experiment in detail.

7.1.1 Task 1: Getting an Overview

Task Description

In this task, we provide the users with a set of documents and ask them to skim through the documents to get an idea of the topics mentioned in them. To help users organize their thoughts and summarize the contents of the documents, we asked them to choose a set of 3 to 5 keywords that best explain the documents in each case. Keywords could be extracted from documents or assigned from outside the document vocabulary. We expected this will force users to ignore details and concentrate on important parts of documents. We also expected that having a set of words in mind, representing the contents of provided texts, can help them in the

next step.

After they typed their selected keywords, the survey showed them two different 'summaries' of the documents: a tag cloud (see Section 3.10.3) and a BubbleNet, both created using the entities extracted from the same set of documents. We created tag clouds using the same set of entities extracted by the Entity Extractor module. Tag cloud shows the *most frequent* tags (entities) with sizes corresponding to their frequencies, while BubbleNet shows the entities together with their relationships in the form of bubbles interactive network.

After the user was shown both interfaces, he or she was asked to choose one that gave a better overview of the documents. They were also welcome to make a comment on their selection.

Experiment Setup

As listed in table 6.3, we developed five different configurations of BubbleNet with respect to different data sets and extraction methods. Of each configuration, we randomly picked three sets of documents, each of size 10, resulting in 15 different document sets (a total of 150 documents). Each time a user starts Task 1, one of these 15 sets is assigned to the user randomly. A user can repeat Task 1 up to 15 times.

In Task 1, a tag cloud shows the top 50 most-frequent entities and a BubbleNet shows 20 entities in the first view and 15 entities when expanding an entity.

7.1.2 Task 2: Finding Information in Relevant Documents

Task Description

In Task 2, we want to evaluate how useful BubbleNet is when users want to look for their desired information. There is a wide variety of information retrieval tasks that one can imagine, but here we chose a task as an example of users' information needs: Given the name of a disease, the users are asked to find the names of 3 to 5 symptoms, medications, diagnosis or treatments related to that disease. They are asked to find those names from the documents provided by our system, not outside or based on their own knowledge.

For this task we only used our health discussion forums dataset. Task 2 consists of two steps: First, we provide the users with the name of a disease and ask them to find related entities using a query-based search. A conventional search interface is developed so that users can enter queries and retrieve documents that match their query. In the second step, we provide the name of another disease and ask the users to find related entities, this time using BubbleNet. Users can navigate through bubbles and hover on the bubbles and their links to access relevant documents.

After doing both steps, we ask the users to choose a method that they think is more appropriate for this task. As an optional question, we asked them why they made that choice. We also measure the time they spend on each step to compare which interface provided a faster way for doing this task.

Experiment Setup

As we described above, we limited Task 2 to our health discussion forums. Therefore, we have had two BubbleNet configurations: Health-AC and Health-MW (see Section 6.3). Each time a user starts Task 2, a configuration is randomly assigned to him.

For selecting disease names, we used a list of disease names in English from Wikipedia[1] and looked up each disease name in our data set to see if there are enough documents related to that disease. This way we picked ten different diseases: *cancer, flu, migraine, asthma, diabetes, anemia, lupus, mumps, hepatitis, tumor.* Each time a user starts Task 2, a pair of different diseases are assigned to him from the above list; one for the query-based search and one for BubbleNet search. For BubbleNet, we also provided a short visual guide to show the user how to use bubbles and their links to find topics and access relevant documents. A user can repeat Task 2 up to 10 times.

7.1.3 The Final Questionnaire

After performing tasks 1 and 2, users were asked to fill a questionnaire. The questionnaire asked users to choose one of the three provided interfaces (tag cloud,

[1] http://simple.wikipedia.org/wiki/List_of_diseases

query- based search and BubbleNet) as the most useful interface and one as the most joyful. This is to see how users evaluate our interface in general after they have used it for some information retrieval tasks. They were also welcome to make a comment on their selection.

7.1.4 Implementation Issues

The survey is implemented as a web-based application so that people can easily access the survey by following a link. Users had the option to start Task 1 or 2 or repeat any of those tasks until they do a task for all possible datasets. They are then forwarded to the questionnaire. The results were saved into our database as the users proceeded with the tasks or the questionnaire.

We removed the name of BubbleNet from the survey in order to avoid biasing the users as far as we could. We asked people from academia (graduate students and professors) to complete the survey.

In the next section we will list the results of this experiment.

7.2 Results

In total, we had 50 participants in Task 1, 40 participants in Task 2 and 41 answers to the questionnaire. In this section we list the results for each part of the experiment in detail. Complete listings of collected data are provided in appendices A, B and C.

7.2.1 Task 1

Out of 56 completions of Task 1, BubbleNet was selected 49 times (87%) as the best interface for getting an overview of the contents, while tag cloud was selected 7 times (13%). We have broken down these results for different data sets and configurations (see Table 7.1). Star signs means that the results are aggregated. For example, *- AC means aggregated results for all datasets with Alchemy-Co-occurrence configuration. Similarly, Reuters-* means aggregated results for the Reuters dataset including both Alchemy-Co-occurrence and MeSH-WordNet configurations.

Table 7.1: Results for Task 1 (Getting Overview of Documents)

Config/Dataset	BubbleNet	TagCloud	Total
*-AC	30 (88%)	4 (12%)	34
*-MW	19 (86%)	3 (14%)	22
Reuters-*	11 (92%)	1 (8%)	12
Health-*	20 (80%)	5 (20%)y	25
Arya-*	18 (95%)	1 (5%)	19
Total	49 (87%)	7 (13%)	56

Results listed in Table 7.1 show that almost the same ratio of preference is obtained in all data sets and configurations in Task 1. Figures 7.1 and 7.2 also show these results.

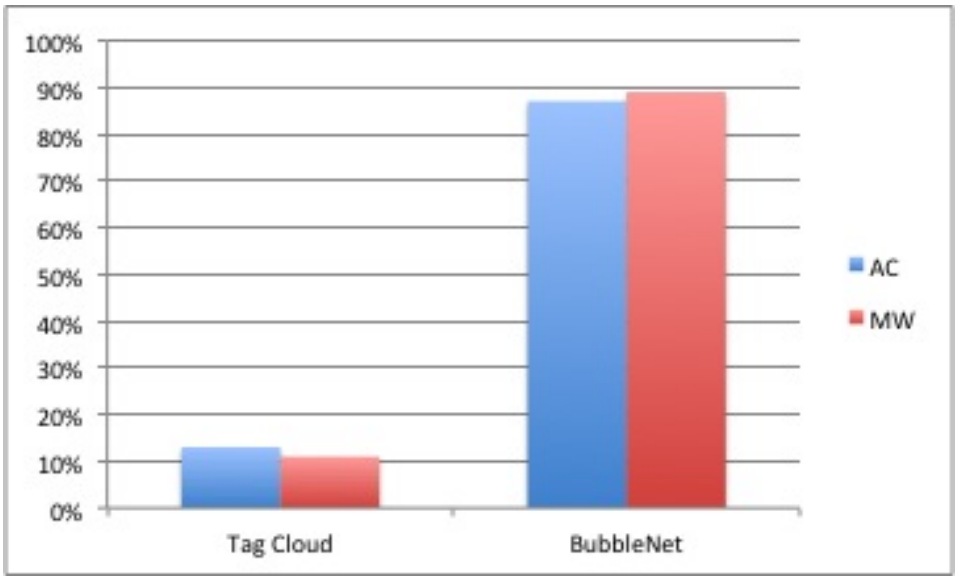

Figure 7.1: Results of Task 1 among different configurations: Users prefer BubbleNet against TagCloud

7.2.2 Task 2

Out of 42 participations in Task 2, BubbleNet was selected 27 times (64%) as the best interface for completing task 2, while query-based search was selected 15 times (36%). Using query-based search, the users spent on average 348 seconds to complete Task 2, while using BubbleNet it took on average 273 seconds (22% decrease). Note that the average search time using BubbleNet also includes the time users spent on learning how to use BubbleNet, while most of the users are familiar with

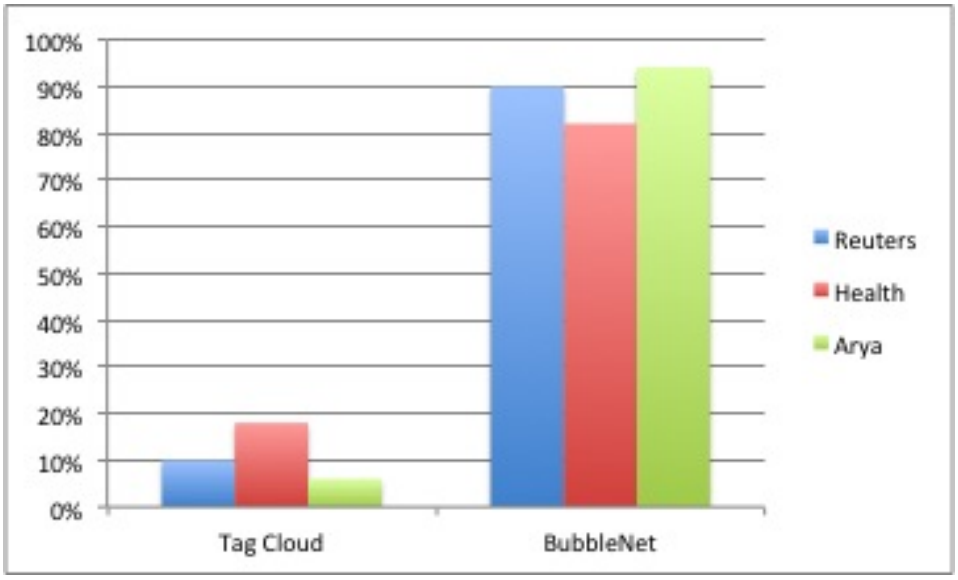

Figure 7.2: Results of Task 1 among different datasets: Users prefer BubbleNet against TagCloud

Table 7.2: Results for Task 2 (Finding Entities Related to a Disease Name)

Config	BubbleNet	Query Search	Total
AC	20 (80%)	5 (20%)	25
MW	7 (39%)	11 (61%)	18
Total	27 (63%)	16 (37%)	43

query-based search.

Table 7.2 and Figure 7.3 show the break-down of the results for AC and MW configurations.

Table 7.3 lists the average search times for Task 2. These results are also shown in Figure 7.4 and Figure 7.5.

Table 7.3: Average Search Times for Task 2 (Finding Entities Related to a Disease Name)

Config	Avg. Time for BubbleNet	Avg. Time for Query Search
AC	213 s	285 s
MW	336 s	447 s
Total	268 s	353 s

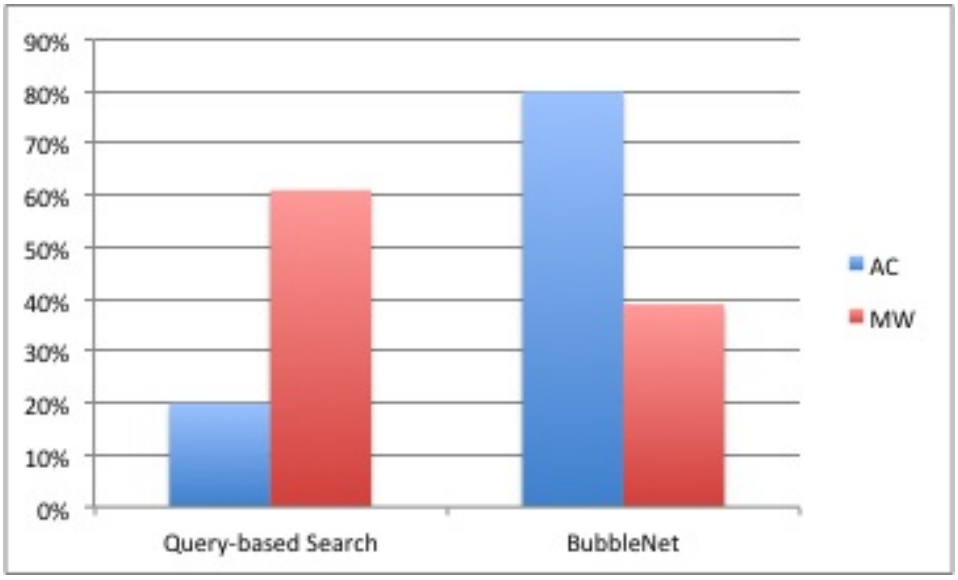

Figure 7.3: Results of Task 2 among different configurations: Users compare query-based search and BubbleNet

Table 7.4: Results of the Final Questionnaire

Config	Question	Tag Cloud	Query Search	BubbleNet	Total
AC	Most Useful	0 (0%)	5 (21%)	19 (79%)	24
	Most Joyful	0 (0%)	2 (8%)	22 (92%)	24
MW	Most Useful	3 (15%)	10 (50%)	7 (35%)	20
	Most Joyful	1 (5%)	5 (25%)	14 (70%)	20

7.2.3 Final Questionnaire

Finally, we report the results of the final questionnaire that users answered when completing the survey. This questionnaire contains two questions: Among the three interfaces that users have observed during completing the survey (tag cloud, query-based search and the BubbleNet), which interface is the most useful and which interface is the most joyful. Similar to Task 2, users to which the AC configuration was assigned for Task 2 were more satisfied with the usefulness of BubbleNet. Table 7.4 lists the results. Note that the first row indicates the configuration assigned to the user for Task 2, regardless of Task 1 configuration.

7.3 Discussion

In our thesis statement, we hypothesized that BubbleNet, as an interactive interface providing an explorable network of interconnected topics extracted from documents, will make users more satisfied in certain information retrieval tasks.

We considered two tasks to investigate if BubbleNet helps users completing those tasks: Getting an overview of the content of a set of documents and finding relevant entities to a given topic.

Task 1 aimed to evaluate BubbleNet on providing a good overview. First, we asked users to choose keywords that they think best represent a summary of the documents. This has two advantages: First, this encourages users to study documents (and make sure they really studied them) before they want to compare BubbleNet vs. tag cloud. Second, having keywords extracted/assigned by users, we can discuss about the words that users tend to choose vs. the words extracted by BubbleNet.

The results of Task 1 (see Table 7.1) show that for all datasets and configurations, the users preferred BubbleNet with a significant difference. This supports a part of our hypothesis, that is, BubbleNet will help users get a better overview of the content of the documents.

We looked at users' comments to see why people preferred tag cloud or BubbleNet (see Appendix A). Among 6 persons who chose tag cloud, only one made a comment, regarding an error they faced in loading documents. Summarizing the comments of users who preferred BubbleNet, they mentioned the following advantages:

- The relationships between topics are also shown
- It is dynamic and interactive, more systematic, intuitive and easier to track
- Topics are simply understood and remembered (better complexity management and attention economy)
- It is more systematic, intuitive and easier to track
- The information is nested in different levels, not all at one level

- It looks nicer and more attractive

In general, people liked BubbleNet mostly because it provided the overview using a structured (interconnected) and explorable (multi-level) visualization.

In Appendix A, we list the keywords users suggested for each dataset/configuration. We mark them if they also appear in the corresponding tag cloud (T) and the first and second levels of BubbleNet (B1, B2).

Task 2 was intended to measure the usefulness of BubbleNet in a search task in comparison to a traditional query-based search. Figure 7.3 compares the number of users that chose BubbleNet vs. tag cloud for different configurations. It shows that for the AC configuration, most of the users preferred BubbleNet, supporting our hypothesis that the BubbleNet helps users find relevant entities to a given disease name more effectively. However, for the MW configuration, users didn't like BubbleNet as much as query-based search. This was unexpected, but showed that the entities and relationships extracted in the MW configuration were not as precise and informative as the AC configuration.

Does this mean that using an ontology won't help BubbleNet produce better results? What these results say is that the combination of MeSH entity extraction and WordNet-based relationship extraction did not produce good results. However, other ontologies may produce better entities and relationships for specific domains. Especially, if we combine an ontology with a domain-independent extraction tool (such as AC), we may observe an improvement as the ontology could help with entities and relationships that are not caught by a domain-independent method.

In the AC configuration the average search time for BubbleNet was less than traditional search. Running a T-test on the search times show that their difference is statistically significant at $\alpha = 0.05$. This difference is not significant for the MW configuration. It can be inferred that the users spent less time to complete Task 2 using BubbleNet (in AC configuration) even when they prefered traditional search over BubbleNet. This supports another part of our statement, that is, BubbleNet helps users perform searching tasks explained in Chapter 7 faster.

Appendix B lists the results for Task 2. Summarizing the users' comments, people who preferred BubbleNet mainly mentioned the following points:

- Easier to follow and much faster to find relevant information
- Suggests more keywords that users can consider when looking for their desired information
- Well structured, complexity management and focusability

On the other hand, people who preferred a traditional search mentioned the following issues:

- BubbleNet cannot combine several keywords and does not accept users' own words to search
- BubbleNet contains misleading information and irrelevant keywords
- Not all documents shown in BubbleNet were relevant to the topic (see the discussion below)
- Search is more straightforward

We also observed some of the mistakes made by users: Some thought that when BubbleNet shows entities related to a particular topic, they are subcategories of the central topic. For example, if entity *symptom* is connected to the central entity *migraine*, some thought that it refers to *migraine's symptoms*. That is, the user expects retrieving topics and documents related to both *symptom* and *migraine* when they click on *symptom*. But in BubbleNet, topics are emerged separately and they are not hierarchical although they are connected. Users have to point the link between *migraine* and *symptom* to retrieve documents related to both topics. This observation shows that people may have different intuitions when using BubbleNet and therefore they may have expectations that are different from our intuition. A possible improvement is to change BubbleNet so that it fits users intuition about the hierarchy of topics.

Finally, after completing the survey, we asked users which interface was the most useful one, as well as the most joyful. This is to measure the users' overall satisfaction about their experience with BubbleNet. As figures 7.6 and 7.7 show,

users preferred BubbleNet with AC configuration over the other interfaces in terms of usefulness in performed tasks. They also significantly preferred BubbleNet as the most joyful interface. This supports another part of our statement that BubbleNet is more useful in certain information retrieval tasks and also provides users with a more pleasant experience.

Appendix C lists the final comments that users made while answering the final questionnaire.

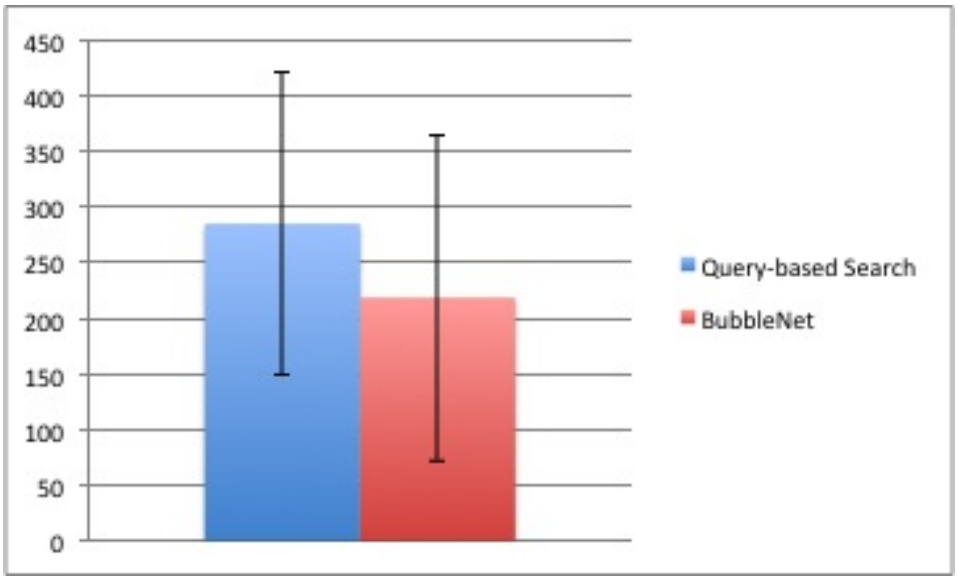

Figure 7.4: Comparing average search times for Task 2 (AC configuration)

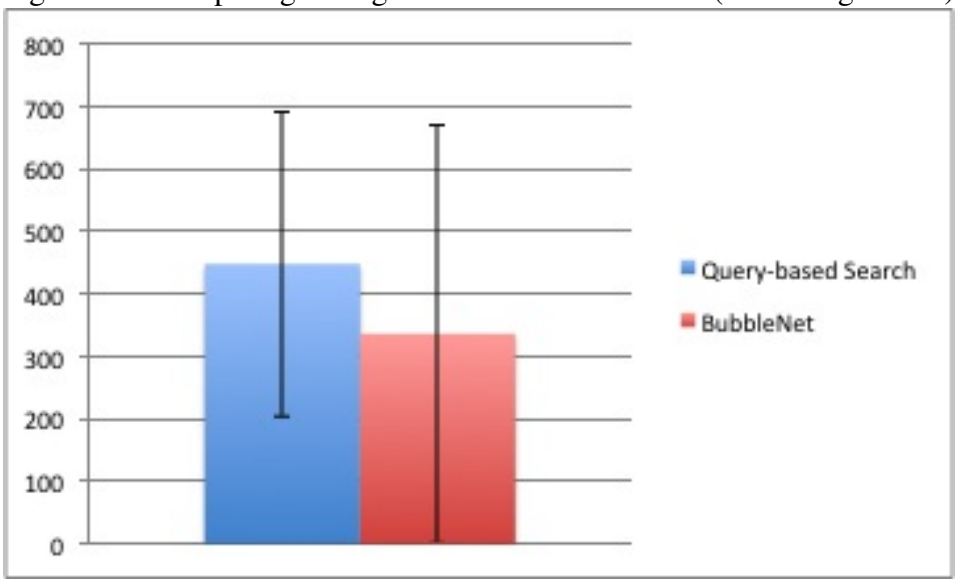

Figure 7.5: Comparing average search times for Task 2 (MW configuration)

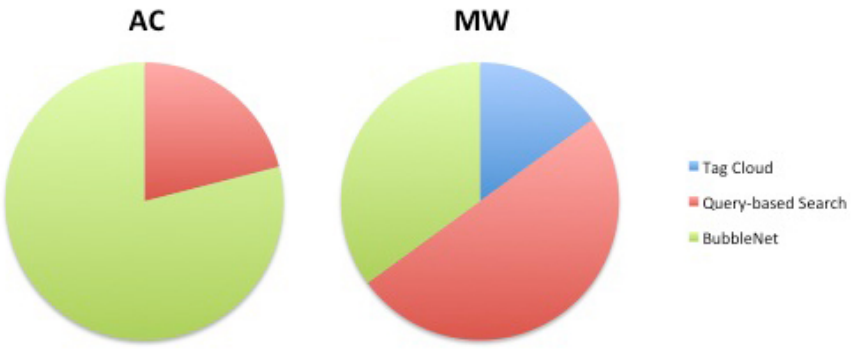

Figure 7.6: Questionnaire results: The most useful interface

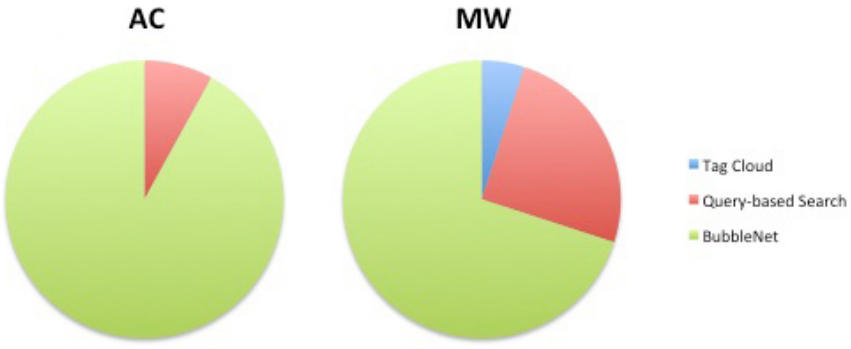

Figure 7.7: Questionnaire results: The most joyful interface

Chapter 8

Conclusion and Research Directions

There have been several interfaces developed for information retrieval systems, such as traditional query-based search and tag clouds. Each has advantages and limitations. Traditional search is very common and effective in retrieving documents, but does not help users get an overview of the documents and requires them to have informative keywords in mind to explain their information need. Tag cloud, on the other hand, provides a summary of contents of the documents, but is not explorable and does not say anything about the relationships between topics mentioned in the documents. It also needs tagged documents as it does not extract important topics automatically.

We introduced an interactive user interface, called BubbleNet, that provides users with an overview of the content of a corpus at a glance in the form of an interconnected network of topics represented as bubbles. It automatically extracts important topics, names, places, etc. and evaluates their semantic similarity to form a network of related entities. Users can navigate through this network, expand bubbles and explore related documents to find their desired information.

We evaluated BubbleNet by asking users to complete an online survey consisting two tasks: Comparing BubbleNet and tag cloud in providing a good overview of documents, and finding entities relevant to a given topic. The results show that users are more satisfied when using BubbleNet for the mentioned tasks when using a domain-independent configuration: They mostly prefer BubbleNet over tag cloud in providing an overview. They also find BubbleNet more useful and faster for finding entities related to a topic, and vote for BubbleNet as the most useful and

pleasant tool among the three interfaces.

Using an ontology-based and domain-specific method for extracting entities and relationships didn't help BubbleNet produce better results. But this does not mean that using domain-specific knowledge can not be helpful. More experiments are needed to evaluate the success of our system when using domain specific knowledge or combining it with domain independent methods.

The current BubbleNet has limitations: It currently doesn't distinguish between different senses of entities (such as *apple: the fruit* and *apple: the company*), doesn't unify similar entities (such as *over-weight* and *overweight*) and doesn't integrate different forms of words (such as *negotiate* and *negotiation*). More advanced natural language processing methods can be used to improve BubbleNet in these aspects.

BubbleNet is a very new interface, meaning that it is not still adapted to users' intuitions. For example, many people complained that they didn't know that by pointing a link between two bubbles they could retrieve documents related to both topics, or the bubbles related to an expanded topic are not subcategories of that topic. Conducting more user experiments will help us improve BubbleNet interface and can make it more intuitive and user friendly.

A very important direction of improvement is the idea of combining the bubbles with the traditional search. People are mostly familiar with traditional search, and it is a very flexible way of retrieving documents. Combining BubbleNet (as an interactive graphical interface) with traditional search can result in a very effective and flexible interface where users can take advantage of both interfaces.

In a frequently visited website, BubbleNet can improve by using users' feedback. Logging the paths that users follow when exploring the network can give us valuable information about importance of the entities and strength of their relationships. For example, if the entity *Canada* is clicked significantly more frequently than others, it means that it is more important than other extracted entities. Similarly, if users always click the bubble *Budget Cut* after expanding the bubble *University of Alberta*, it can be inferred that the link between those two entities is strong. This information can be used to continuously update BubbleNet to better fit

the users' information needs.

Another important issue is studying the use of ontologies in constructing BubbleNet. Although our small experiment here indicated that our configuration of domain specific entity and relation extraction was not improving BubbleNet, using better knowledge resources might be more helpful and has to be investigated.

Finally, a precise evaluation of BubbleNet needs further user experiments, consisting of more information retrieval tasks and a larger population of users.

Bibliography

[1] O. Kaser and D. Lemire, "Tag-cloud drawing: Algorithms for cloud visualization," *CoRR*, vol. abs/cs/0703109, 2007.

[2] "Tagcrowd visualization: State of the union," June 2013.

[3] K. Knautz, S. Soubusta, and W. G. Stock, "Tag clusters as information retrieval interfaces," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pp. 1–10, IEEE, 2010.

[4] Infomous, "3 ways to visualize text with infomous."

[5] "The economist: Topics most commented on."

[6] B. Daille, E. Gaussier, and J.-M. Langé, "Towards automatic extraction of monolingual and bilingual terminology," 1994.

[7] H. Nakagawa, "Automatic term recognition based on statistics of compound nouns," *Terminology*, vol. 6, no. 2, pp. 195–210, 2001.

[8] S. Lee and H. joon Kim, "News keyword extraction for topic tracking," in *Networked Computing and Advanced Information Management, 2008. NCM '08. Fourth International Conference on*, vol. 2, pp. 554–559, 2008.

[9] P. Knoth, M. Schmidt, P. Smrz, and Z. Zdrahal, "Towards a framework for comparing automatic term recognition methods," in *Conference Znalosti 2009*, 2009.

[10] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 216–223, Association for Computational Linguistics, 2003.

[11] G. Dias, S. Guilloré, J.-c. Bassano, J. Gabriel, and J. G. P. Lopes, "Combining linguistics with statistics for multiword term extraction: A fruitful association?," 2000.

[12] J. Wermter and U. Hahn, "You can't beat frequency (unless you use linguistic knowledge): a qualitative evaluation of association measures for collocation and term extraction," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, (Stroudsburg, PA, USA), pp. 785–792, Association for Computational Linguistics, 2006.

[13] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: Practical automatic keyphrase extraction," in *Proceedings of the fourth ACM conference on Digital libraries*, pp. 254–255, ACM, 1999.

[14] Y. Matsuo and M. Ishizuka, "Keyword extraction from a single document using word co-occurrence statistical information," *International Journal on Artificial Intelligence Tools*, vol. 13, no. 01, pp. 157–169, 2004.

[15] G. R. Krupka and K. Hausman, "Isoquest inc.: Description of the netowl (tm) extractor system as used for muc-7," in *Proceedings of MUC*, vol. 7, 1998.

[16] S. Singh, D. Hillard, and C. Leggetter, "Minimally-supervised extraction of entities from text advertisements," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, (Stroudsburg, PA, USA), pp. 73–81, Association for Computational Linguistics, 2010.

[17] X. Liu, F. Wei, S. Zhang, and M. Zhou, "Named entity recognition for tweets," *ACM Trans. Intell. Syst. Technol.*, vol. 4, pp. 3:1–3:15, Feb. 2013.

[18] C. Nobata, S. Sekine, H. Isahara, and R. Grishman, "Summarization system integrated with named entity tagging and ie pattern discovery.," in *LREC*, 2002.

[19] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum, "Information retrieval using a singular value decomposition model of latent semantic structure," in *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 465–480, ACM, 1988.

[20] S. Arora, R. Ge, and A. Moitra, "Learning topic models - going beyond svd," *CoRR*, vol. abs/1204.1956, 2012.

[21] S. T. Dumais, "Latent semantic analysis," *Annual Review of Information Science and Technology*, vol. 38, no. 1, pp. 188–230, 2004.

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[23] T. Griffiths, M. Jordan, and J. Tenenbaum, "Hierarchical topic models and the nested chinese restaurant process," *Advances in neural information processing systems*, vol. 16, pp. 106–114, 2004.

[24] W. T. Chuang and J. Yang, "Extracting sentence segments for text summarization: a machine learning approach," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, (New York, NY, USA), pp. 152–159, ACM, 2000.

[25] M. Hu, A. Sun, and E.-P. Lim, "Comments-oriented blog summarization by sentence extraction," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 901–904, ACM, 2007.

[26] J. Park, T. Fukuhara, I. Ohmukai, H. Takeda, and S.-g. Lee, "Web content summarization using social bookmarks: A new approach for social summarization," in *Proceedings of the 10th ACM workshop on Web information and data management*, pp. 103–110, ACM, 2008.

[27] H. Ji, B. Favre, W.-P. Lin, D. Gillick, D. Hakkani-Tur, and R. Grishman, "Open-domain multi-document summarization via information extraction: Challenges and prospects," in *Multi-source, Multilingual Information Extraction and Summarization*, pp. 177–201, Springer, 2013.

[28] M. Steinbach, G. Karypis, V. Kumar, *et al.*, "A comparison of document clustering techniques," in *KDD workshop on text mining*, vol. 400, pp. 525–526, Boston, 2000.

[29] G. Raveendran and C. L. Clarke, "Lightweight contrastive summarization for news comment mining," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, (New York, NY, USA), pp. 1103–1104, ACM, 2012.

[30] B. Zhao, Z. Zhang, Y. Gu, X. Gong, W. Qian, and A. Zhou, "Discovering collective viewpoints on micro-blogging events based on community and temporal aspects," in *Proceedings of the 7th international conference on Advanced Data Mining and Applications - Volume Part I*, ADMA'11, (Berlin, Heidelberg), pp. 270–284, Springer-Verlag, 2011.

[31] A. H. Razavi, D. Inkpen, D. Brusilovsky, and L. Bogouslavski, "General topic annotation in social networks: A latent dirichlet allocation approach," in *Advances in Artificial Intelligence*, pp. 293–300, Springer, 2013.

[32] Y.-C. Chang and S.-M. Chen, "A new query reweighting method for document retrieval based on genetic algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 5, pp. 617–622, 2006.

[33] R. Navigli and P. Velardi, "An analysis of ontology-based query expansion strategies," in *Proceedings of the 14th European Conference on Machine Learning, Workshop on Adaptive Text Extraction and Mining, Cavtat-Dubrovnik, Croatia*, pp. 42–49, 2003.

[34] M. Agosti, F. Crivellari, and G. M. Di Nunzio, "Web log analysis: a review of a decade of studies about information acquisition, inspection and interpretation of user interaction," *Data Mining and Knowledge Discovery*, vol. 24, no. 3, pp. 663–696, 2012.

[35] Y. Song, D. Zhou, and L.-w. He, "Query suggestion by constructing term-transition graphs," in *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 353–362, ACM, 2012.

[36] J.-R. Shieh, Y.-H. Hsieh, Y.-T. Yeh, T.-C. Su, C.-Y. Lin, and J.-L. Wu, "Building term suggestion relational graphs from collective intelligence," in *Proceedings of the 18th international conference on World wide web*, pp. 1091–1092, ACM, 2009.

[37] P. A. Chirita, C. S. Firan, and W. Nejdl, "Personalized query expansion for the web," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, (New York, NY, USA), pp. 7–14, ACM, 2007.

[38] P. Bhattacharyya and M. Khapra, "Word sense disambiguation," *Emerging Applications of Natural Language Processing: Concepts and New Research*, p. 22, 2012.

[39] J. Chen, O. R. Zaïane, and R. Goebel, "An unsupervised approach to cluster web search results based on word sense communities," in *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, vol. 1, pp. 725–729, IEEE, 2008.

[40] D. P. Spence and K. C. Owens, "Lexical co-occurrence and association strength," *Journal of Psycholinguistic Research*, vol. 19, no. 5, pp. 317–330, 1990.

[41] P. Buitelaar, P. Cimiano, and B. Magnini, *Ontology learning from text: methods, evaluation and applications*, vol. 123. IOS press, 2005.

[42] B. Tang, Y. Wu, M. Jiang, Y. Chen, J. C. Denny, and H. Xu, "A hybrid system for temporal information extraction from clinical text," *Journal of the American Medical Informatics Association*, 2013.

[43] B. Percha, Y. Garten, R. B. Altman, *et al.*, "Discovery and explanation of drug-drug interactions via text mining," in *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, p. 410, World Scientific, 2012.

[44] C. C. Aggarwal, *Mining text data*. Springer Science+ Business Media, 2012.

[45] Y. Hassan-Montero and V. Herrero-Solana, "Improving tag-clouds as visual information retrieval interfaces," in *International Conference on Multidisciplinary Information Sciences and Technologies*, pp. 25–28, Citeseer, 2006.

[46] M. J. Halvey and M. T. Keane, "An assessment of tag presentation techniques," in *Proceedings of the 16th international conference on World Wide Web*, pp. 1313–1314, ACM, 2007.

[47] V. Nováček and G. Burns, "Skimmr: machine-aided skim-reading," in *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion*, IUI '13 Companion, (New York, NY, USA), pp. 59–60, ACM, 2013.

[48] "Mesh (medical subject headings) (http://www.nlm.nih.gov/mesh/meshhome.html)," 2012.

[49] P. Sondhi, "Shallow information extraction from medical forum data," in *In Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, vol. Poster Volumes, pp. pages 1158–1166, Beijing, China, 2010.

[50] B. G. Cimino JJ, "Automatic knowledge acquisition from medline.," in *Methods of Information in Medicine*, vol. 32(2);120-130, 1993.

[51] G. Cong, L. Wang, C. Lin, Y. Song, and Y. Sun, "Finding question-answer pairs from online forums," in *In Proceedings of SIGIR.*, vol. 2, (dsc), 2008.

[52] V. Pedro, S. Niculescu, and L. Lita, "Okinet: Automatic extraction of a medical ontology from wikipedia," *In WiKiAI08: a workshop of AAAI2008*, 2008.

[53] Zhou, Xiaohua, H. Han, I. Chankai, A. Prestrud, and A. Brooks, "Approaches to text mining for clinical medical records," in *In SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, (New York, NY, USA. ACM.), pp. 235–239, 2006.

[54] B. Chee, R. Berlin, and B. Schatz, "Predicting adverse drug events from personal health messages," in *In AMIA Annual Symposium Proceedings*, pp. 217–226, 2011.

[55] S. Radhouani, J. Lim, J.-P. Chevallet, and G. Falquet, "Combining textual and visual ontologies to solve medical multimodal queries," *In in Proc. IEEE ICME*, 2006.

[56] D. Sankoff and J. B. Kruskal, "Time warps, string edits, and macromolecules: the theory and practice of sequence comparison," *Reading: Addison-Wesley Publication, 1983, edited by Sankoff, David; Kruskal, Joseph B.*, vol. 1, 1983.

[57] S. G. Kobourov, "Spring embedders and force directed graph drawing algorithms," *CoRR*, vol. abs/1201.3011, 2012.

[58] J. Fagnan, O. R. Zaïane, and R. Goebel, "Visualizing community centric network layouts.," in *IV*, pp. 321–330, 2012.

Appendix A

Results Listing for Task 1

Table A.1 lists the results of Task 1. Each row corresponds to a participation in this task, including user's field of study (major), assigned dataset and configuration, user's suggested keywords, user's choice (between BubbleNet and tag cloud) and user's comments, if any.

#	Major	Dataset	Config	keywords	Choice	Comment
1	Computer Science	Health	mw	medicine, bp, hospital(T)	tagcloud	
2	Computer Science	Health	mw	thyroid(TB1), test(TB1), blood(T), pressure(TB2), results, alcohol, alcoholics, health, support, medical, x-ray(T), health, healthy, metabolic	bubblenet	Neither are really good.
3	Computer Science	Health	mw	thyroid, tsh(TB1), hormone(TB2)	bubblenet	
4	Chemical Engineering	Health	ac	medical help, patient, seeking medical advice, long questions	tagcloud	
5	MBA	Aria	mw	health(TB1), study(TB1), theory	bubblenet	because they are separated and beautiful and duyamic :) I liked it
6	information technology engineering	Aria	ac	obesity(TB1), weight(TB1), food(T)	bubblenet	more attractive simply understood simply remembered and analysed
7	Civil Engineering	Health	ac	health medical forum	bubblenet	1- It looks better 2- It shows the relation between the keywords
8	engineering	Reuters	ac	bussiness, money, management	bubblenet	
9	Computer	Reuters	ac	Computer(B1), network, economy, share(B2)	bubblenet	
10	mathematics	Health	ac	whatever, toolong, blah	bubblenet	
11	Computer Science	Reuters	ac	trade finance bank(T)	bubblenet	Describes how the entities are connected with each other.
12	Computer Science	Aria	ac	obesity(TB1), weight loss(TB1), good diet	bubblenet	because it shows the relation of the words that gives more information. And also the information is nested so you could find more information in different levels.
13	Computing Science	Health	ac	lump(TB1), pain(TB1), normal, suggestion, help(T), health, hernia(TB1)	bubblenet	There are several advantages: 1) Connection of each subject to the relevant one 2) It is interactive, so search, and work with it would be easier 3) It is more attractive 4) There is a search box making search so easy
14	computer science	Aria	mw	exercise(TB2), canadian, activity(T)	bubblenet	
15	Civil	Reuters	ac	fiscal(B2), orders , shipment(T)	bubblenet	
16	Massage therapy	Health	ac	running, pain, marathon	tagcloud	

#	Major	Dataset	Config	keywords	Choice	Comment
17	Computer Science	Aria	ac	health, obesity, research, case study	bubblenet	
18	Computing Science	Health	mw	diagnosis, blood pressure(TB1), hospital(T)	bubblenet	
19	Computer Science	Aria	ac	food(T), adipocyte-derived protein leptin, health(T), eating(B2)	bubblenet	It is more systematic and easier to track
20	civil engineering	Health	mw	pain, knee, gym	bubblenet	
21	Computing Science	Reuters	ac	acquisition of brooks drug, report, agricultural commodity(TB2), federal limits, bolivia(TB1), economic stabilisation program, public spending, the soviet union(TB1), motor industry(T), shipment, export, financial structure, cash flow(B1)	bubblenet	intractable shows topics relations
22	Electrical Engineering	Aria	mw	obesity(TB1), implications, diseases, prediction, health(TB1), treatment	bubblenet	It makes relations more understandable and it is more organaized. It is more like what brain works.
23	Civil Engineering	Aria	ac	obesity, health, medical research	bubblenet	Visualization 2 provides better overview, but using it is not easy enough. Using Visualization 1 in nice and easy
24	Computer Science	Aria	mw	obesity(TB1), cause, treatment(TB1)	bubblenet	
25	software engineering	Aria	mw	overweight(T), height, weight(TB1), self-reported obesity	bubblenet	in terms of precision and recall and also the better complexity management and attention economy in visualization(= better HCI-related Characteristics).
26	cs	Reuters	ac	shares(T), stock(T), finance(TB1), trade	bubblenet	more information but not all at one time
27	computer science	Health	ac	health symptoms question diagnosis	bubblenet	
28	CS	Health	ac	disease(T), problem, question	bubblenet	

#	Major	Dataset	Config	keywords	Choice	Comment
29	Electromagnetics	Aria	ac	obesity(TB1), patient(TB2), treatment(TB2), program(TB2), weight loss(TB1)	bubblenet	
30	Environmental engineering	Health	ac	biopsy(T), thyroid(TB1), heart(TB2)	bubblenet	
31	Computer Science	Health	ac	fever, sore throat, mumps	bubblenet	looks nicer!
32	Electrical Engineering	Aria	ac	weight loss(TB1), mortality risk(T), obesity(TB1), chronic diseases(B2), hypothalamus	tagcloud	
33	computer science	Aria	ac	bariatric-obesity-chronic disease-dgat1- food industry	bubblenet	
34	Computing Science	Aria	mw	obesity(TB1), diet, bmi	bubblenet	
35	computing science	Health	ac	mump, sore throat(B1), rash(B2)	bubblenet	
36	Electrical Engineering (Communication)	Aria	mw	Obesity(TB1), Public Health, Weight Management	bubblenet	It contains a much more sophisticated information content along with relations and structural connections between the concepts which is a much more complete model for the text content. Yet it is also interactive, which makes it possible to customize the representation to ones appropriate way.
37	Business & IT	Aria	mw	fitness, obesity(TB1), health(B1)	bubblenet	It looks more organized
38	Physics	Health	mw	Pain(B2), disease(TB1), cure, illness, help(B2), problem	bubblenet	
39	Chemical Engineering	Reuters	ac	business, coffee market(TB2), export(TB2), import, stock market(TB2)	bubblenet	
40	health	Health	mw	health, patient(B2), questions	bubblenet	
41	computer science	Health	ac	diseas, cure, medical	bubblenet	
42	Computer Science	Health	mw	health, medicine, disease	tagcloud	
43	cs	Health	ac	sport, reflux, symptom, depression	bubblenet	more information

#	Major	Dataset	Config	keywords	Choice	Comment
44	Computer Science	Reuters	ac	business, news, financials	bubblenet	More intuitive
45	Health	Reuters	ac	economy, finances, money	tagcloud	
46	medicine	Health	mw	exercie, knee, pain(TB2)	bubblenet	faster emphasis of the search word
47	Physical Education	Aria	mw	obesity, comorbidity, determinants	bubblenet	
48	Medicine	Aria	ac	meeting	bubblenet	
49	none	Reuters	ac	loss(T), agriculture commodity(TB2)	bubblenet	I like bubbles
50	Music	Health	mw	Advice, health, overwhelmed	bubblenet	
51	Nursing	Health	mw	Personal, hypochondria, advice	bubblenet	
52	Diabetes Education and Management	Health	ac	Post op experience, sports injury, pain, first-person descriptions of personal experience	bubblenet	
53	Kinesiology	Aria	mw	obesity(TB1), weight-management(TB2), treatment strategy	bubblenet	Visualization 2 allows you to break down the topic from bigger ideas to smaller ideas and sub-categories. Although I find visualization 2 less impactful in terms of its presentation.
54	chemistry-medicine	Reuters	ac	profit, trade markets	bubblenet	
55	medicine	Reuters	ac	financial commodity markets(TB2)	bubblenet	it shows visually which term is important and how it is linked to other terms and we can interactively explore each term, satellite terms and even the links
56	Nursing	Health	mw	Thyroid(TB1), abnormal levels, questions	tagcloud	There was an error loading the documents I was unable to view the second document.

Table A.1: Results Listing for Task 1

Appendix B

Results Listing for Task 2

Table B.1 lists the results of Task 2. Each row corresponds to a participation in this task, including user's field of study (major), assigned configuration and disease names, user's suggested entities for the two diseases (using two different methods), average search times, user's choice (between BubbleNet and traditional query-based search) and user's comments, if any.

In this table, *QST* and *BST* stand for *query-based search average search time* and *BubbleNet average search time* respectively.

#	Major	Config	Diseases	QST	Query Search: Entities	BST	BubbleNet: Entities	Choice	Comments
1	Computer Science	Health-AC	tumor,cancer	372	mitomycin, cancer, mri, adenoma, chemo	245	lekema, chemo, swelling, radiation, surgery	bubblenet	
2	information technology engineering	Health-AC	anemia,diabetes	311	low circulation, not sleeping well, anigma	187	constipation, dizziness, headache	bubblenet	it is faster
3	Civil Engineering	Health-AC	flu,hepatitis	322	sore throat, fatigue, stomachace, protonix	360	lack of appetite, weight loss,catalase	query	because I wanted to search a combination of the words which were not available in the second one
4	Computer Science	Health-MW	tumor,cancer	47	sorafenib sunitinib bevacizumab	100	chemo jaundice liver	bubblenet	Although second method has more potential, it is not without its faults for the purpose of this experiment. Specifically, the name of documents is trimmed so it is not possible to identify what a document pertains to.
5	Computer Science	Health-MW	cancer,tumor	896	biopsy, chemo, radiation, pain, nose bleeds	682	ice pick headaches,loss of appetite,tingling, prickling, dizziness	bubblenet	You can be more sure of the documents to choose when using bubble network. You need less search through a loaded document to see if it is related to what you need or not.
6	Computing Science	Health-AC	migraine,mumps	180	headache, anxiety, dizziness	201	swelling, fever, pain	bubblenet	Easier to follow
7	Computer science	Health-AC	lupus,asthma	276	joint pain, accutane, fatigue	499	pain in toe nails, brittle muscles, chest pain	bubblenet	It gave me more related words, so I had an expectation about the words that I should look for in the document.
8	Civil	Health-AC	mumps,migraine	53	swollen testes	30	throbbing	bubblenet	
9	Massage therapy	Health-MW	hepatitis,flu	76	jaundice, cirrhosis, diclofenac	94	constipation, imuran, random body aches	query	
10	Computing Science	Health-MW	mumps,migraine	98	inoculation pain swollen head hurting herpes	119	pain headache hemicrania	bubblenet	Much faster to find relevant information about the topic
11	Computer Science	Health-AC	flu,hepatitis	132	atches, nexium, stomach pain, burping	144	blood test, liver disease	query	Bubbles did not contain much information
12	Computing Science	Health-AC	lupus,asthma	454	joint pain, hair loss, rash	291	coughing, chest pain, inhaler, prednisone, allergy	bubblenet	easier and faster
13	Electrical Engineering	Health-AC	mumps,migraine	222	throat pain, clogged ears feeling, swelling,	215	headaches, spinach juice, beet, carrot juice, cucumber juices	bubblenet	easier to find out!
14	Civil Engineering	Health-AC	mumps,migraine	591	pain in neck, pain, headache	15	pain in eyes, headache, mri	query	bubble were full of misleading and unusable information.
15	Computer Science	Health-MW	cancer,tumor	452	pain fatigue weight loss	482	pain radiation vitamin	query	

#	Major	Config	Diseases	QST	Query Search: Entities	BST	BubbleNet: Entities	Choice	Comments
16	software engineering	Health-AC	flu,hepatitis	277	no appetite, gurgling in his abdomen like he was hungry, loss of weight, a lump appeared at the base of his neck and his stomach was swelling (he had peritoneal (stomach) mesothelioma) that causes ascites in the stomach region, allergy treatments	554	blood test, fluid in the belly, jaundice, weight loss, fatigue, lack of appetite, confusion	bubblenet	well structured, complexity management of huge contents,focusability
17	cs	Health-AC	lupus,asthma	198	headache, visual problems, clots, vein thrombosis	296	headache, albuturol, cough, itchy feeling inside ches	query	Not all documents presented in the second method where about asthma
18	Computing Science	Health-AC	anemia,diabetes	258	dehyderation, weekness, angina, fever, stomach pain	43	pain, anxiety, headaches	bubblenet	
19	computer science	Health-AC	asthma,lupus	408	feel short of breath, bronchoconstriction,fatigued	139	plaquenil, rash, fibro	bubblenet	
20	CS	Health-MW	lupus,asthma	133	nausea vdrl plaquenil	178	back pain infection	query	Could not find any answers with bubbles!
21	Electromagnetics	Health-MW	mumps,migraine	2502	fever, pain killer, salivary glands swelling,	356	headaches, hemicrania, pain killer, properanolol	query	The bubbles method was more convenient if the keywords in the bubbles were more relevant.
22	Electrical Engineering	Health-MW	asthma,lupus	683	albuterol, inhaler, theophylline	232	arthritis, blood, pain	bubblenet	
23	CS	Health-MW	mumps,migraine	440	pregnancy, pain killer, paracetamol, fever, swollen glands	679	headache, surgery, buprenorphine, stress, pressure on optic nerve	bubblenet	1) More fun. 2) It helps with useful keywords. 3) I assume otherwise your research would be meaningless :)
24	CS	Health-AC	hepatitis,flu	345	isoniazid, vaccine, liver test, blood	499	fatigue, virus, flu shot, pain, fever	bubblenet	
25	computer science	Health-AC	diabetes,anemia	430	blood sugar-insulin-tiredness	164	allergy-anxiety- body pain	bubblenet	
26	Computing Science	Health-MW	hepatitis,flu	304	liver, cirrhosis, liver transplant, dna test	357	flu shot, throat pain, virus, headache	bubblenet	1) More fun. 2) It helps with useful keywords. 3) I guess otherwise your research would be meaningless :)
27	computing science	Health-AC	asthma,lupus	221	sinusitis , infections , acid reflux , eczema , zithromax	32	pain , fibro , rash	bubblenet	much easier
28	computing science	Health-AC	tumor,cancer	81	grows, carcinoid , adenoma	156	abscess , rash , oral lichen planus , , bump , , reaction	bubblenet	more useful, informatic
29	Electrical Engineering (Communication)	Health-MW	tumor,cancer	594	nephrectomy, manual lymph drainage (mld), radiation therapy	981	esr count, endometriosis, white blood cell count,	query	the search option is not limited to some predefined words and combination of words are not limited to two.

#	Major	Config	Diseases	QST	Query Search: Entities	BST	BubbleNet: Entities	Choice	Comments
30	Business & IT	Health-MW	anemia,diabetes	259	fever iron deficiency fatigue depression weight gain	112	ache numbness body odor	query	The second was slow and difficult to follow
31	Physics	Health-MW	lupus,asthma	305	rash, foot pain, blood in urine	926		query	
32	health	Health-AC	hepatitis,flu	404	pain, fatigue, enzymes, methotrexate	111	sius congestion, pressure, fever	bubblenet	
33	engineering	Health-AC	tumor,cancer	311	remove tumore, physical therapy, lazer beam	192	ionized treatment, chemotherapy,numbness	bubblenet	
34	computer science	Health-AC	lupus,asthma	505	rosaica joint rash toung rheumatoid	155	lung coughing inhaler	bubblenet	
35	Computer Science	Health-MW	asthma,lupus	140	allergy, inflammation, steroids	305	feel dizzy,fatigue,ssri	query	
36	cs	Health-MW	lupus,asthma	237	rash, migraine, nausea, sensitivity to light	81	coughing, wheezing, shortness of breath, chest tightness and pain.	bubblenet	quicker
37	Computer Science	Health-AC	anemia,diabetes	307	hypoglycemia, low blood pressure, iron capsules, iron deficiency, vitamin c	152	upper abdominal pain, headaches, blood sugar	bubblenet	More intuitive interface. Requires quite a bit of refining though.
38	Health	Health-AC	tumor,cancer	202	surgery, chemotherapy, swelling	329	pain, chemo, radiation, surgery	bubblenet	
39	Health	Health-AC	tumor,cancer	202	surgery, chemotherapy, swelling	329	pain, chemo, radiation, surgery	bubblenet	
40	Nursing	Health-MW	asthma,lupus	177	shortness of breath, salbutamol, peak flow.	308	pain, rashes, mental fog.	query	
41	Nursing	Health-MW	flu,hepatitis	136	Sore throat, fever, ache	13		query	Bubbles do not work as instructed on ipad
42	Kinesiology	Health-AC	tumor,cancer	66	cancer, growing, malignant, surgery, death	129	pain, chemo, breast, surgery, test	query	More straight forward. The floating bubbles are a bit difficult to get accustomed to because they move around. Perhaps if they were stable it would be easier.
43	Nursing	Health-MW	flu,hepatitis	572	vomiting, achey, fluids, headaches, rest	38	Error unable to see documents	query	Unable to view second box documents

Table B.1: Results Listing for Task 2

Appendix C

User Comments from the Final Questionnaire

Below are the comments that users made on our system. They pointed out some issues that they faced when dealing with BubbleNet and comparisons to the other interfaces.

Note that the letters A, B and C refer to tag cloud, query-based search and BubbleNet interfaces respectively.

- *I think bubble net along with word cloud provides better insights. Word clouds some how give a more general and big picture of the documents you have they are simpler. On the other hand bubble net provides more thorough information about the document but it is more complicated. But for if you are familiar with them you could get more information and find what you want easier and faster with bubble net. Bubble net has more usages, word Cloud could just help to get the big picture of the whole stack of documents. Search box is the most primary method and takes more time.*

- *First,this page has no question (3) Second, the third interface is quite slow. The search box could be limited to search within a specific domain (optional). E.g, there would be a checkbox that limits the search for the center bubble.*

- *Interface (C) was great but I experienced some slowness and freezing when clicking on bubbles*

- *I think it was not consistence for different experiments. In general C is easier to track and to find info. But sometimes it do not give enough information.*

- *complexity management of huge results, focus-ability, attention economy and HCI-related issues are other important factors in the usefulness, efficiency, pleasure-ability and joyfulness of using Bubbles in information retrieval. Thank You...*

- *B and C could be combined.*

- *You should have a better guide for the bubbles method. It was a little confused how to open a document at first. Also once I figured out how to use the edges it was much better experience, which I think was not explained. I wonder whether the path to a bubble is relevant to what the user sees, but I think it should be. For example if I click on the pain bubble while the flu bubble is expanded it should not be the same pain bubble that I expand from the migraine bubble.*

- *Combining a search with the bubbles could be useful. Search is powerful and the bubble interface can reduce the search space doing a selection of the documents.*

- *I think I am accustomed to interface 2 but prefer the idea of interface 3. I find interface 1 prescriptive and limiting.*

- *I definitely find the search box the most useful, however I can see the appeal of the bubbles interface with the sub-categories of ideas. I think with some work to make it a bit more user-friendly (less movement) it could be a good system.*

Appendix D

Implementation Details

After providing a high-level model of the system, in this chapter we explain the details of our implementation.

D.1 Overview

Most of the parts were implemented in Java, while the user interface is implemented in PHP and Javascript. Documents, entities and relationships are implemented as Java classes until they are stored in the database. Beyond that point, they are treated as database records.

For each component (Document Loader, Entity Extractor, Relationship Extractor and Database Interface) we first designed a Java interface to set a standard way of communication between components. Then, for each method we used, we designed a separate class implementing that interface. This way, the system could be easily adapted for a particular application just by configuring appropriate concrete classes subsuming abstract interfaces.

D.2 Document Loader

The Document Loader interface defines the functionality of Document Loader component, which is accepting new documents and loading them into BubbleNet system. Our dataset consists of files stored in two different formats as we explained in section 6.4.1: `.dcu` and `.txt`. Therefore, we implemented two different Document Loader classes to load documents. Once a document is loaded, a new instance of the Document class is created and the contents of the document is stored in it, as

well as the document address and its date. Also, a unique document ID is generated and assigned to that document. Then the resulting object is passed to the next layer.

D.3 Entity and Relationship Extractors

We implemented two Java interfaces for Entity Extractor and Relationship Extractor components. The former provides the functionality of accepting documents and returning a list of Entity objects. The latter one, similarly, provides interface for accepting documents and entities and returning a list of Relationship objects. Below is the definition of these interfaces:

```
// Entity Extractor Interface
public interface EntityExtractor
{
      public void initialize(); // Called at startup time

      public Entity[] getEntities(Document d)  // Returns
         list of entities extracted from
document d
      public Entity[] getEntities(Document d, int n)  //
         Returns top-n entities extracted
from document d
}

// Relationship Extractor Interface
public interface RelationshipExtractor
{
      public void initialize(); // Called at the startup
         time

      public Relationship[] getRelationships(Document d,
         Entity[] entities); // Extracts
relationships between given entities from document d
}
```

As described in sections 6.1.1 to 6.1.4, we used two different methods for extracting entities and two different methods for extracting relationships. Correspondingly, we have two implementations of the Entity Extractor component, as well as two implementations of the Relationship Extractor.

Alchemy Entity Extractor: AlchemyAPI provides an API for using their services online. Thus, we extracted keyphrases and named entities from all documents in our datasets using this API and stored them so that we can access them offline. At the time of adding documents to BubbleNet, this module loads the extracted keyphrases and named entities, performs the pre-processing stage described below and returns the list of entity objects. The pre-processing steps are:

- Removing unusable entities, such as URLs and symbols
- Trimming up extracted entities (removing extra spaces, etc.)
- Combining keyphrases and named entities
- Removing duplicates

For every extracted entity, an instance of class Entity is generated and appropriate document ID, phrase, score and date is assigned to it. Then an array of entities is returned.

Since the number of entities indicates the number of relationships and therefore, the size of database and processing times, this module can be set in a way that it returns only k top- scored entities.

Co-Occurrence Based Relationship Extractor: This class generates an array of Relationship objects for a given document and its extracted entities. For every pair of entities, the score of the corresponding relationship is calculated using the algorithm explained in section 6.1.2 and an instance of class Relationship with appropriate entities, score, document ID and date is generated.

To limit the number of relationships generated for a given document, this module can be set to pass only relationships with scores higher than a threshold.

MeSH Entity Extractor: This module uses the list of medical headings we extracted from MeSH, as we explained in section 6.1.3, removing all categorization data. We also removed entities shorter than 3 characters, as well as phrases that have a single-letter word at the beginning or the end. This process resulted in a vocabulary with 104,277 entries.

At the time of program startup, this dictionary is loaded into memory. As a document arrives, entries of this vocabulary that appear in documents are extracted

and scored according their term frequencies. Similar to Alchemy Entity Extractor, this module generates an array of Entity objects and can be set to return only k top-scored entities.

WordNet Relationship Extractor: In order to connect to WordNet, we used RiTa.WordNet[1]: a Java library for accessing WordNet API. Similar to Co-occurrence Based Relationship Extractor, this module produces an array of Relationship objects and can be set to return only relationships with scores higher than a threshold.

D.4 Database Layer

Documents, as well as extracted entities and relationships are stored in a database. The Database Layer has responsibility to build and maintain the database and to provide an interface for connecting to the database. As database server, we used MySQL, which is a high-performance server that is easily integrable with both Java environment and web applications.

The design of database schema was a challenging problem, because the database has to be able to answer queries in an appropriate response time, as well as to update and organize newly arrived information with an acceptable speed. BubbleNet can be used for huge text corpora, thus, the database should be designed in a way that it can deal with large amount of data in reasonable time.

To this end, we tried different designs and ended up in a schema that satisfies the conditions mentioned above. We have five tables in the database:

- **Documents:** Table `documents` stores all documents, including their contents, automatically-generated ID, date and address fields and is indexed based on document IDs. When a user wants to retrieve a document, this table is used to retrieve the contents of the desired document or its (physical or online) address.

- **Entities:** Table `E` stores all entities, including their phrase, date and score fields. Each record in this table indicates an entity extracted from documents of a particular date. If an entity appears in multiple documents within a single date, the score in this table is the total score of those entities. Yet, if an entity

[1] http://www.rednoise.org/rita/wordnet

appears in documents of several dates, it also appears multiple times in this table (one entry for each date). Thus, each record corresponds to appearance of an entity in all documents of a particular date.

This table is indexed based on entity phrases, scores and dates and is used to find the most important entities within a particular time interval.

- **Entity-Documents**: Table `ED` stores the ties between entities and documents. Each entry corresponds to appearance of a single entry in a document. Thus, if a document appears in several documents of a single date, there will be one entry for each document in this table. Note that even if an entity appears several times in a single document, it is extracted as a single entity and therefore, it occupies only one row in this table.

  The table `ED` is indexed based on entity phrases, scores and dates and is used to find documents that are most relevant to an entity within a particular time interval.

- **Relationships:** Table `R` stores all relationships, including the phrases of the pair of entities engaged in the relationship, date and score fields. Phrases of the two entities are sorted alphabetically and concatenated using a delimiter character to form a single field in order to make the indexing process easier.

  Similar to table `E`, each entry in this table corresponds to a particular date. Therefore, all relationships extracted from documents of a specific date that engage two particular entities will be summed up and stored in this table as a single row.

  This table is indexed based on phrases, scores and dates and is used to find the strongest relationships between two given entities within a particular time interval.

- **Relationship-Documents:** Table `RD` is used to store the ties between relationships and documents. That is, every single relationship extracted from a document has a row in this table. This way, if two entities appear in several documents in a single date, there is multiple records in this table, one for each document.

This table is indexed based on entity phrases, scores and dates and is used to find documents from which two given entities get the highest relationship scores.

In order to reduce the size of the database, we rounded all dates to the first days of the week. That is, all documents published during a week are labeled with the date of the first day of that week. This reduces the number of possible distinct dates while keeps users able to search within particular time intervals with an acceptable precision.

Figure D.1 shows the five tables described above with a few sample rows.

D.5 User Interface

The user interface for this system consists of two parts: A visualizer module that retrieves information and represents it in a visually pleasant manner, and an intermediate component that connects to the database layer and translates user actions into database queries. Here we describe these two parts:

Intermediate Component: The graphical interface (visualizer) is a client-side application that retrieves data from website and visualizes them. Thus, the intermediate component should be a server-side application that connects to the database and provides an interface for the client-side graphical user interface. We implemented this module in PHP[2], providing the following functionalities:

- Connecting to BubbleNet database
- Accepting requests from the client-side application through GET method[3]
- Interpreting requests and translating them into MySQL valid queries
- Getting query results from MySQL database and extract desired information
- Pack desired information in a format that the client-side application can use
- Send the results back to the client-side application

[2]http://www.php.net/

[3]http://en.wikipedia.org/wiki/GET_(HTTP)#Request_methods

A request from the client-side application contains information such as the function to be called (e.g. get most important entities, get related entities, etc.), dataset name, number of items (e.g. entities) to be returned and start and end dates of the desired interval. The response from the server-side component is the information retrieved from database in an appropriate format.

Graphical Interface: This component is where the entire BubbleNet system meets end users and has the responsibility to provide an interactive graphical environment to the users. We implemented this part in JavaScript[4] which is a safe and fast platform for client-side user interface implementation and is compatible with most of commonly-used web browsers.

Using Ajax[5] technology, the interface first connects to the server-side component and requests for a list of the most important entities and their significant relationships. Results are then unpacked and appropriate bubbles and links are created according to retrieved data. For visualization and graphics tools, we used Paper.js[6], a free graphics library for JavaScript that provides many useful graphical tools for drawing shapes, creating interactive objects and dealing with user's behaviour.

The script simulates physics laws to predict the movements of bubbles, as described in Section 6.3.3. It then captures mouse events to interact with the user:

- When the user points a bubble, that bubble gets a thicker stroke to tell the user that it something will happen if the user hovers or clicks on it.
- When the user hovers on a bubble for a short while, the script expands that bubble as a preview, while keeping the rest of bubbles with less opacity.
- When the user clicks on a bubble, other bubbles are removed, a new request is sent to the server to retrieve related entities to the clicked one, and once the script receives the results, it generates new bubbles and connects them to the clicked one. Physics laws then move this expanded bubble to the center as others tend to stand around it due to the links between them and the central bubble.

[4] http://en.wikipedia.org/wiki/JavaScript
[5] https://en.wikipedia.org/wiki/Ajax_(programming)
[6] http://www.paperjs.org/

- When the user hovers on a link or on an expanded bubble, the script retrieves a list of most relevant documents and shows it to the user. By clicking an item in that list, the document will be shown to the user with the keywords highlighted.

- The user is also able to slide handles of the time interval slide bar to indicate desired time span.

The script also keeps the record of the bubbles a user navigates, so that the user can click on the arrow buttons on the left and right sides to navigate back and forward in the history.

Finally, the user can enter a phrase to fly immediately to its corresponding bubble (if exists). The script accepts the phrase and sends a request to the server to get related information.

Figure D.2 shows a snapshot of the first view of the interface which shows the top-15 important entities of a blog and their significant links. Figure D.3 shows the overview when the time interval has changed. Figure D.4 shows the interface when a bubble is expanded and the list of related documents is retrieved. Figure D.5 shows the preview of related entities to an entity when the user hovers on it.

documents

id	docID	address	content
1	WqeidcioSg	/cshome/smohajer/BubbleNe	Arya Sharma, the scientific director of the federally
2	yAT1TkQo5H	/cshome/smohajer/BubbleNe	Yesterday's post , as expected, provoked several e-mail
3	yu6TvKOaEf	/cshome/smohajer/BubbleNe	that the nonchalantly-delivered recommendation to

E (entities)

id	entity	date	score
10476	54th annual meeting	2008-11-23	0.554203
1892	56-week phase	2009-01-04	1.486225
5216	5htp-nat exts	2009-09-13	1.77755

ED (entities-documents)

id	entity	date	docID	score
63	severe obesity	2009-08-30	yu6TvKOaEf	0.33783
64	competent advise	2009-08-30	yu6TvKOaEf	0.337707
67	impulsivity	2009-04-19	mscrmHRmep	0.903834
68	impulsiveness	2009-04-19	mscrmHRmep	0.90248

R (relationships)

id	entities	date	score
34058	addictive component@glycemic index	2009-01-18	2.9621345888251
34001	addictive component@nicotine	2009-01-18	1.72053333333333
34005	addictive potential@alberta	2009-01-18	0.848528137423857
34019	addictive potential@auckland	2009-01-18	2.38945523498558

RD (relationships-documents)

id	entities	date	docID	score
101	article@treatment	2009-02-15	WqeidcioSg	0.842727272727273
103	article@country	2009-02-15	WqeidcioSg	0.880909090909091
106	caroline richardson@weight loss	2008-08-10	yAT1TkQo5H	1.41624414538555
109	caroline richardson@intervention	2008-08-10	yAT1TkQo5H	2.01295402757744
110	annals of family medicine@caroline	2008-08-10	yAT1TkQo5H	0.799090909090909

Figure D.1: Database tables with sample rows

Figure D.2: Graphical User Interface: An overview of the most important entities and their significant relationships.

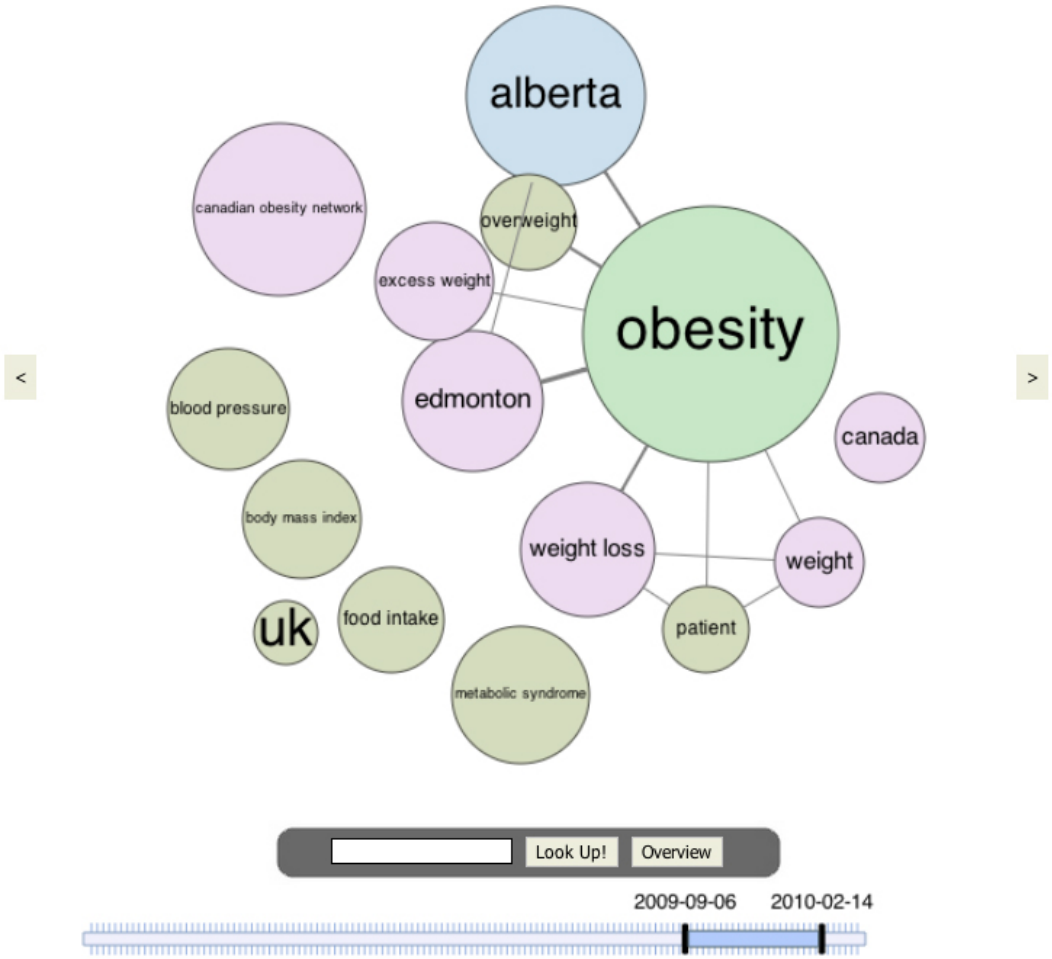

Figure D.3: Graphical User Interface: The overview with a narrower time interval

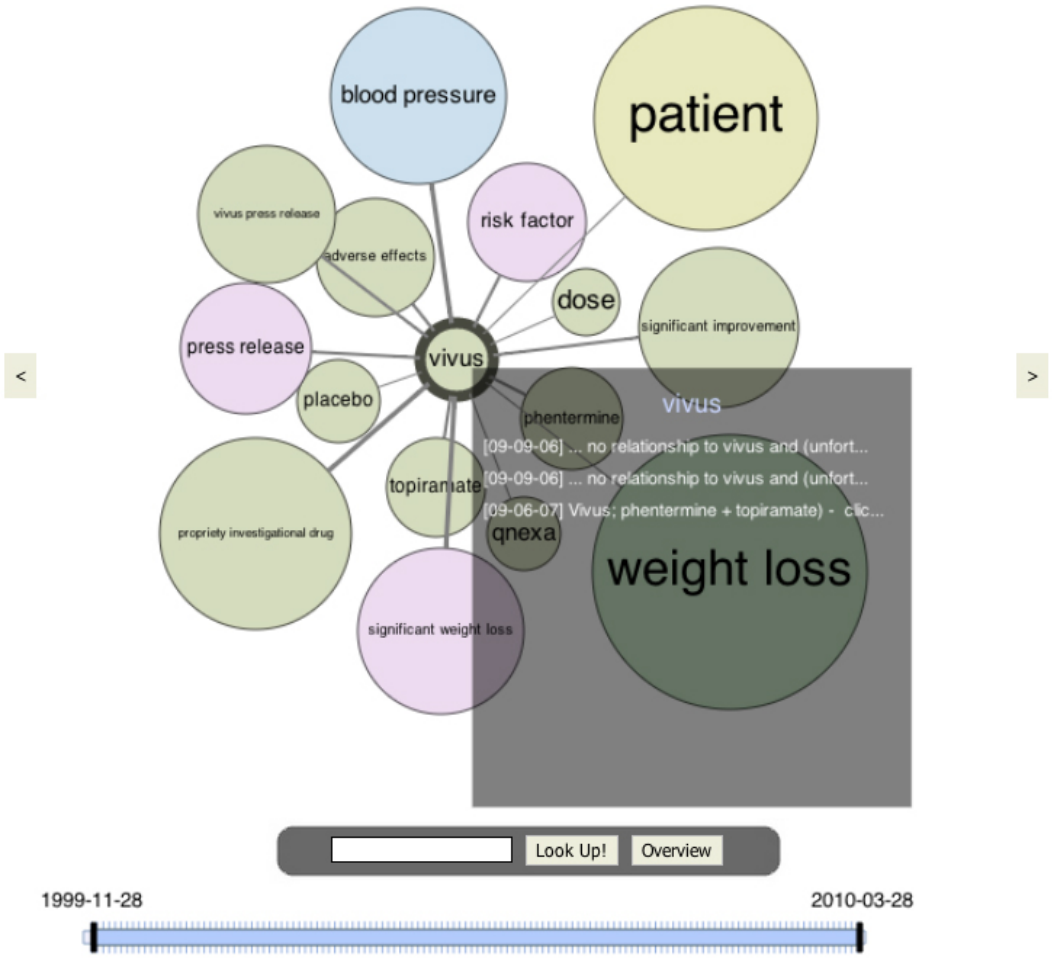

Figure D.4: Graphical User Interface: The bubble ‘vivus’ is expanded and the list of related documents is shown.

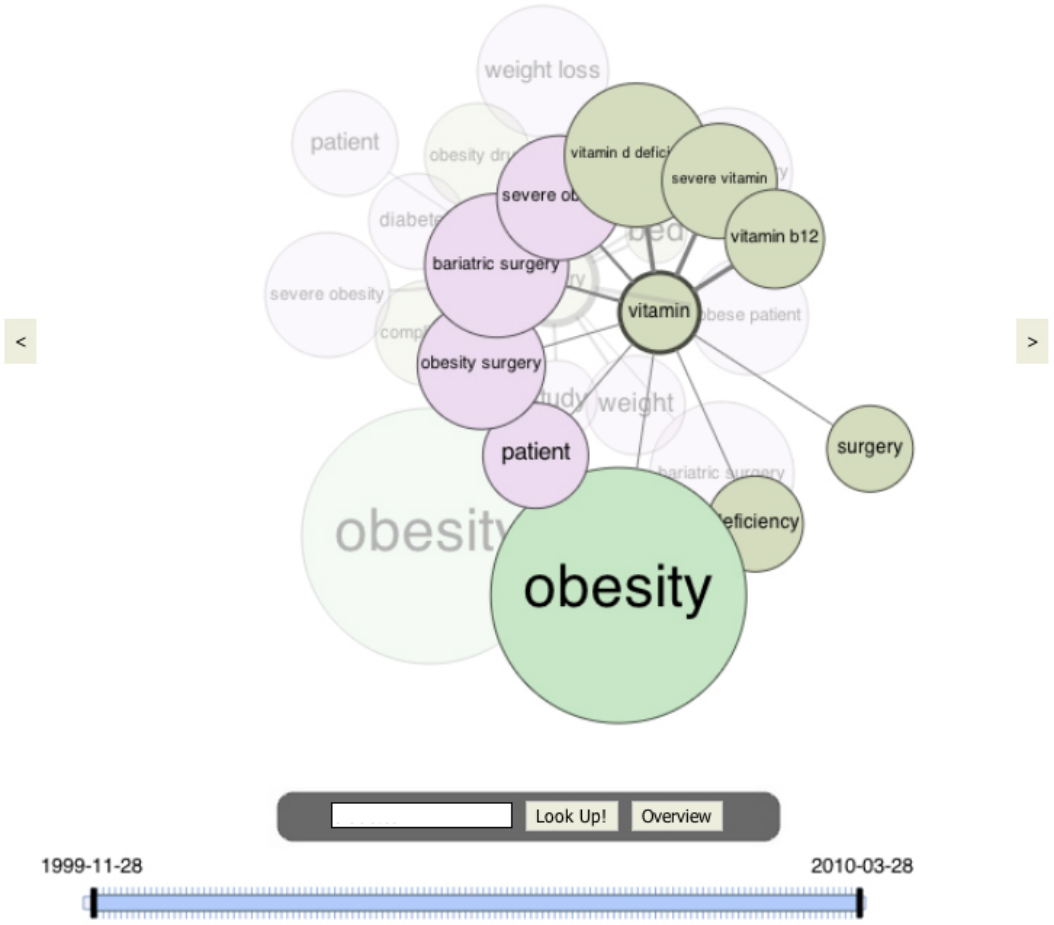

Figure D.5: Graphical User Interface: A preview of related entities to the entity ‘vitamin’