University of Alberta

TEXT DOCUMENT TOPICAL RECURSIVE CLUSTERING AND AUTOMATIC LABELING OF A HIERARCHY OF DOCUMENT CLUSTERS

by

Xiaoxiao Li

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 2012
Edmonton, Alberta

Abstract

The overwhelming amount of textual documents currently available highlights the need for information organization and discovery. Effectively organizing documents into a hierarchy of topics and subtopics makes it easier for users to browse the documents.

This thesis borrows community mining techniques from social network analysis to generate a hierarchy of topically coherent document clusters. It focuses on giving the document clusters descriptive labels. We propose to use different centrality measures in networks of co-occurring terms to label the document clusters. We also incorporate keyphrase extraction and automatic titling in cluster labeling. The results show that the cluster labeling method utilizing KEA to extract keyphrases from the documents generates the best labels overall comparing to other methods and baselines. We also built an interactive browsing web interface for users to examine the taxonomies.

Acknowledgements

I would like to thank my supervisor Dr. Osmar Zaiane for his time, patience, and guidance. My gratitude also goes to the Alberta Ingenuity Centre for Machine Learning (AICML) for supporting this research and Leslie Acker, AICML Administrative Coordinator, for her great help. I would like to express my appreciation to my fellow students and friends who participated in the user survey: Jing Zhang, Kevin Quinn, Ying Xu, Saeed Mohajeri, Haiming Wang, Fan Xie, Weitian Tong, Dapeng Li, Chenlei Zhang, Richard Zhao, Lingjia Deng, Hengshuai Yao. The most special thanks goes to my parents who have loved and supported me for all my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this information-explosion era, the retrieval and representation of the right information is vital for people′s information needs. For textual document collections, the two main types of information needs are: (1) finding a specific piece of information and (2) browsing the topics and structure of a given document collection [21].

A search engine is an effective information retrieval tool for finding a certain piece of information. With most search engines, after submitting a query a user receives a long ranked list of results. Then the user examines the list of results and tries to locate the documents of interest. Because of the large number of retrieved documents, users usually only examine the top results returned. Search engines strive in document retrieval and page ranking to try to display the right pages at the top. They do work well when the query is non-ambiguous and straightforward. However, about 16 percent of user queries are estimated to be Ambiguous Queries, that is to say, they have multiple meanings [65][33]. For example, the query "jaguar" could mean "jaguar the car", "jaguar the animal", "jaguar Mac OS" or "jaguar guitar" etc; the query "AVP" could mean the movie "Alien Vs. Predator", the "Association of Volleyball Professionals ", the company "Avon Products", an "Anti-Virus Program", or the airport code for "Hotell Wilkes-Barre Scranton International Airport". There are even more queries that are Broad Queries that have multiple subtopics [33]. For example, the query "music" covers various subtopics such as "music instruments", "music lesson", "classic music", "jazz music" etc. There are also times when it is hard to describe a query or the user is just too lazy to type more words for the query to be specific enough [20]. In these situations, docu-

ments on all kinds of different aspects of the query, and even irrelevant documents are mixed together and returned to the users. Even an experienced user would waste time and energy in sifting through the long list of results to locate the ones that they really need. They may miss information if they only examine the top documents in the ranked list [13].

The second kind of information need is to browse a document collection without a well-defined goal of searching [20]. A user may just want to gain an overview of a certain document collection and maybe have some discoveries along the way. Just like we want to see the "table of contents" before reading a book, one may want to know the topics and the structure of a document collection. For example, a reader may want to know what topics a blog web site covers to see whether it is of interest; an executive may want to monitor the company emails to have an overview of the subjects discussed, and online forums and conferencing systems would benefit from categories that are generated automatically. For this need, the long list of documents would not be effective either. It would be rare for one to take the time and effort to read all the documents to get an idea of the topics of the documents.

One of the solutions to the above problems is document clustering and labeling. This procedure aims at clustering a document collection into smaller groups where each group is on a different topic. This process could be done recursively until the topics are specific enough. This will generate a hierarchy of document clusters with labels. This representation allows users to effectively zoom in and locate the documents of interest. It facilitates the searching and browsing process [5].

The presentation of a hierarchy of topics and subtopics is superior to ranked lists in many aspects. Other than saving the users′ time and energy, it could help formulate refined queries as users browse the hierarchy of topics, one can easily switch from browsing to searching for some concrete information they saw from the index. It could also give preliminary recommendations of the category of a web page for web sites that are manually archived [66]. Another aspect worth mentioning is that this representation is even more useful on mobile devices than on PCs. Searching and browsing tasks on mobile devices is popular. Mobile devices usually have much smaller screens and keyboards than PCs, users tend to enter very

short queries. Short queries could be more ambiguous and broad than queries with more words. Also, the users are more reluctant to flip the pages in the search results both because of the extra effort and the data usage concerns [10]. A hierarchy of topics and subtopics would vastly help mobile users.

1.1 Thesis Statements

In this thesis we will elaborate on using document clustering and labeling (also known as Automatic Taxonomy Generation ATG) to generate taxonomies that aid the browsing process of a document collection. More precisely we are addressing the following statements:

- TS1: Borrowing community mining techniques from social network analysis to group documents based on term co-occurrences can generate good taxonomies both on the disambiguation of different query senses, and on subtopics of the same topic.
- TS2: Different centrality measures in social network analysis can provide better cluster labeling results than the Degree Centrality measure.
- TS3: Keyphrase extraction and automatic titling can be successfully incorporated in the process of document cluster labeling.

1.2 Thesis Contributions

In this thesis we present a document clustering and labeling method aiming at organizing a collection of documents into a hierarchy of topics and subtopics with descriptive labels. Here we list three main contributions of this thesis.

Our first contribution is extending query sense community discovery algorithm by Chen et al. to taxonomy generation [13]. To generate hierarchical document clusters for a set of documents, we first extract keywords from the documents according to their Document Frequency, and then we generate a keyword graph based on selected keywords and their co-occurrences. Then we use a modularity community mining algorithm to detect different topics on the keyword graph. The

documents are then mapped to each keyword community and document clusters are generated. We do community mining recursively when necessary according to some stopping criteria. This way we generate a taxonomy for a set of documents which would help users in the browsing process. Our method generates keyword communities and document clusters together. This method has an advantage over traditional document clustering methods in that doing community mining on the keywords is less time-consuming. It is also better than word-based ATG methods in that this method can generate more than one cluster labels. Besides, the Fast Modularity method that we use in community mining automatically detects the number of topical keyword communities. We evaluate the clustering performance of our method on real queries on the web and compare our results with a commonly used document clustering algorithm K-Means. We also present a detailed description of the data collection and pre-processing process. The experiments show that our method works well on disambiguating the different senses of a query (Ambiguous Queries), but not as much on the separation of different subtopics belonging to the same topic (Broad Queries).

Another contribution of this thesis is on cluster labeling. Our labeling methods are compared with different baselines according to the ground truth gathered by a user survey we conducted. Based on the keyword communities, we propose to use Betweenness Centrality, and PageRank to extract cluster labels according to their centrality scores. Our experiments show that these two centrality measures both outperform the commonly used Degree Centrality measure in labeling on the top levels. Also, we propose to incorporate keyphrase extraction and automatic titling in labeling the document clusters. Given a document cluster, we extract the important terms from each document, and then select cluster labels based on their importance in each document cluster. We compare our methods with several frequency-based baselines and another advanced baseline named Frequent and Predictive Words method [58]. The results show that the labeling method utilizing KEA for keyphrase extraction from the documents gets the best overall results on all levels in the taxonomy. We also present detailed post-processing methods on the retrieved labels. Additionally, we have found that using external knowledge sources

in labeling is not suitable for this task in that they are time-consuming and that these sources introduce errors.

Moreover, we built an interactive browsing web interface to examine the taxonomies. In the interface, users can click on different queries and see the search results with a presentation of a tree-like taxonomy. Each node in the tree is a document cluster with a label given by our labeling method. Users can browse a document collection with this taxonomy. Given the resources to run the time-consuming off-line parts such as the collection of the documents, this interface can be transformed into a search engine with better search result presentation than just a ranked list.

1.3 Thesis Organization

This thesis is organized into four parts. The first part describes related works in improving the document presentation for browsing and focused on post-retrieval document clustering and labeling. The second part describes our approach in topical clustering and automatic cluster labeling for text documents. The third part details in the experiment setup, experimental results and discussions. Finally the last part gives conclusions about the contributions of this thesis and explores future venues.

The first part has two chapters. Chapter 2 introduces three main methods in improving the ranked list presentation of documents. Chapter 3 is a survey on post-retrieval document clustering and labeling which is also known as Automatic Taxonomy Generation (ATG).

The second part Chapter 4 details our ATG approach. It describes our approach in five phases with the focus on Phase V which is cluster labeling.

The third part is divided into two chapters. Chapter 5 introduces the experiment setup including data collection and pre-processing, evaluation metrics, and parameter settings. Chapter 6 presents the performance of our method both on clustering and on labeling. The interactive browsing interface that we developed in this work is also described in Chapter 6.

Part I

Related Work

Chapter 2

Attempts in improving the ranked list

Some attempts have been made to help users to focus on the partition of documents that they may be interested in. In this section we introduce three major ones: query refinement recommendation, pre-retrieval classification and post-retrieval clustering. We discuss why the first two methods are not suitable for our task.

2.1 Query refinement recommendation

Popular search engines such as Google, Yahoo! and Bing give query refinement recommendations in the form of "Related Searches" besides the search results (See Figures 2.1, 2.2, and 2.3). For example, for the query "jaguar", Google recommends the user to also try "jaguar animal or jaguar xf". This method does show some subtopics of a certain query but it has several shortcomings. First, they all utilize user query logs which may not be available on all document collections. Second, the recommendations do not have a hierarchical structure. Third, they do not make an effort in grouping similar topics. For example, in Figure 2.1, "jaguar xf", "jaguar xk" all belong to "jaguar cars" but they are not grouped together. Lastly, search engines only display the query refinements that have been searched the most frequently. Some query senses that are not popular enough may be left out. For example, in Figures 2.1, 2.2, and 2.3 we can see that none of the search engines suggests "jaguar Mac OS" or "jaguar guitar" because they are less queried comparing to "jaguar animal" and "jaguar car".

Figure 2.1: Query refinement by Google.ca on the query *jaguar*

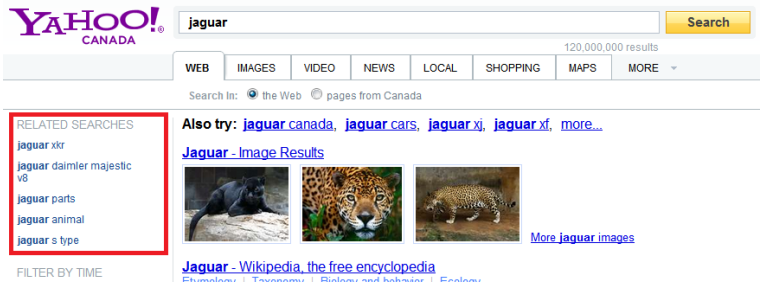

Figure 2.2: Query refinement by Yahoo Canada (ca.yahoo.com) on the query *jaguar*

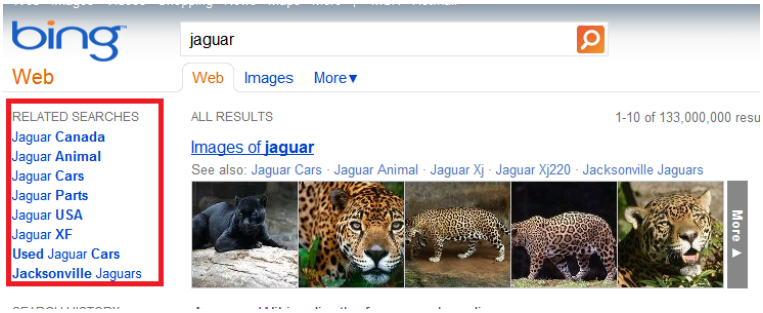

Figure 2.3: Query refinement by bing.com on the query *jaguar*

Another project along this line is the Google Knowledge Graph which was launched on May 16th 2012 and not available in Canada yet at the time of writing. It claims to be able to distinguish the different entities a query refers to. Figure 2.4, which is taken from Google′s blog, shows that when you search for *Taj Mahal*, the Google knowledge graph would not only show you the famous monument in India, but also the Grammy Award-winning musician, and even a casino named *Trump Taj Mahal Casino Resort*. It recommends different objects that respond to the same query. Google uses several knowledge bases such as Freebase, Wikipedia and the CIA World Factbook to get the knowledge graph [64]. It is limited to distinguish named entities and facts, but not queries with different subtopics or senses. Also, it uses the large amount of user query log which may not be available on other document collections to rank the results .

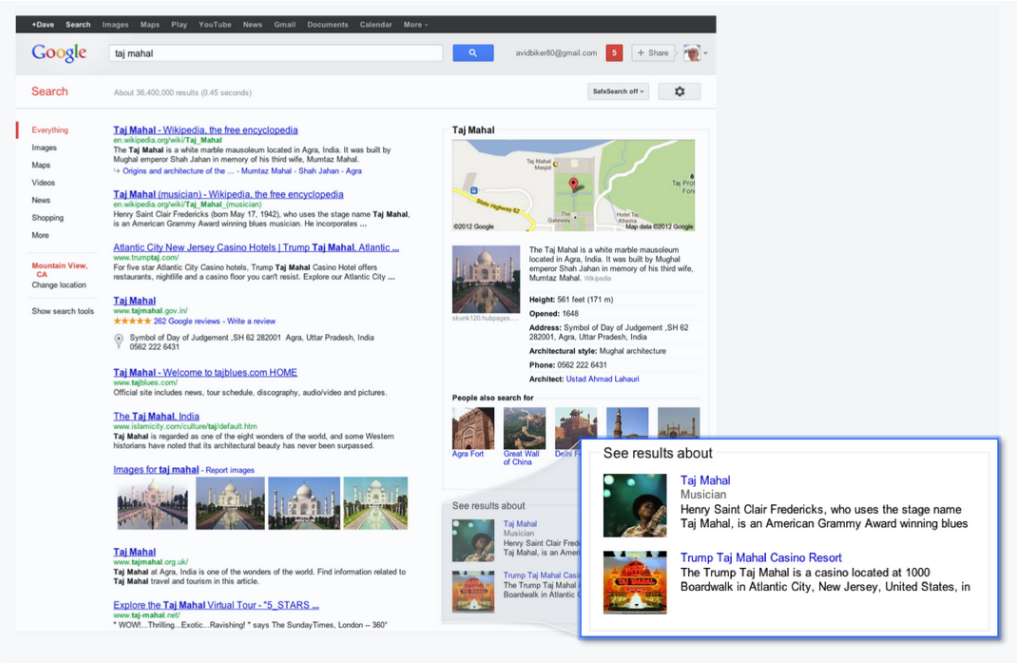

Figure 2.4: Google′s knowledge graph on the query *Taj Mahal*, taken from a Google blog[64]

Duckduckgo.com is another search engine that gives query refinement recommendations. For example, Figure 2.5 shows the query refinement recommendations for the query *jaguar* given by duckduckgo.com. It presents all the different categories *jaguar* could fall into such as *cat, cars, aircraft, companies and enter-*

tainment. A user can click on one and be led to more specific meanings of *jaguar* and even perform a new search. These categories appear to be extracted from the Wikipedia page :*Jaguar (disambiguation)*. This is better than using user query logs in that it covers all the different senses of a query if Wikipedia has them. However, this does not work if Wikipedia does not cover that query. Also, this will not generate different subtopics if a query is a Broad Query.

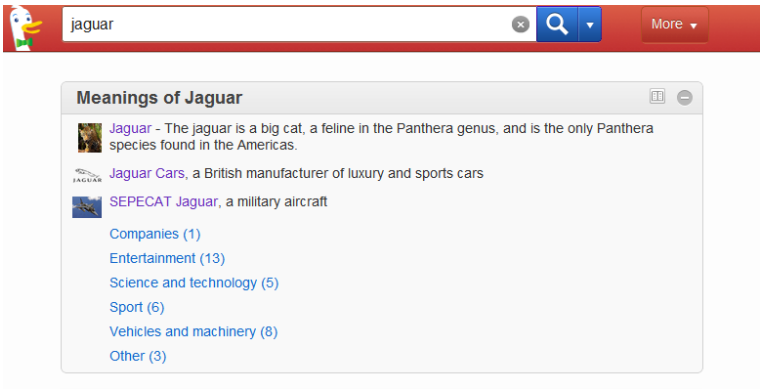

Figure 2.5: Query refinement recommendations for the query *jaguar* by duckduckgo.com

2.2 Pre-retrieval classification

Classification can also be used in bringing documents into order as a pre-clustering method. It classifies each document into one of the pre-defined classes. It could be done before the document collection is formed. Given a document collection, it forms document clusters based on the pre-classified classes. For example, Attardi and Macro, and Chen and Dumais both used the Yahoo! Directory[1] along with other ontologies to classify web pages [12][2]. An ontology is usually made by human experts. For example, Figure 2.6 shows the top levels categories in the Yahoo! directory. The advantage of this method is that the categories they use are well-defined and distinctive. The downside is that it is expensive to create and maintain such an ontology [71]. Also, due to its manual nature, it is not possible for this ontology to cover all the categories or to pick up all the new topics, especially over

[1] http://dir.yahoo.com/

the World Wide Web which is dynamically changing all the time [24]. For example, the category of "Computers and Internet" in Yahoo! Directory does not cover the subcategory "Artificial Intelligence". Another web directory DMOZ[2], which has wider and more up-to-date coverage, only covers less than five percent of the web [40]. There are works that try to enrich the ontologies automatically. For example, the HITS algorithm [37] can fill an ontology with authoritative web documents. However, it cannot generate new topics. The lack of topics can cause documents of the newer topics placed into a wrong category or no category at all. At the same time, documents of mixed topics may fall into the same category [12].

Figure 2.6: Top level categories in Yahoo! Directory

2.3 Post-retrieval document clustering and labeling

Another way to solve this problem is by document clustering and labeling which is the solution this work focuses on. Researchers in the Information Retrieval (IR) community have used document clustering to re-organize and represent documents and have observed superior results than ranked lists [18][20][1][32][71].

Using document clustering and labeling to generate a hierarchy of topics and subtopics is also known as Automatic Taxonomy Generation (ATG) in the IR community [71]. Document clustering methods attempt to group similar documents of

[2]www.dmoz.org

the same topic together. A group of documents generated can be referred as a cluster. The clusters are then labeled with the proper labels that indicate their topics. A user can look at the document clusters and select the topic of interest and be led to relevant documents. It also helps with the navigation process if the user just wants to browse the documents. Comparing to query refinement recommendations and pre-retrieval classification, document clustering can generate the taxonomy fully automatically with no external knowledge.

Some commercial systems that use ATG to represent search results are yippy.com and carrotsearch.com. Vivismo.com was also a famous search engine that provided a taxonomy of search results. It has received various awards, drove interests from Microsoft and Google, and some authors even say that "clustering technology is the PAGERANK of the future" [24]. Now VIVISMO is part of IBM and no longer available to the public. Yippy.com bought VIVISMO′s technology Clusty and used it on their family-friendly search engine. Figure 2.7 is an example of the result representation Yippy.com gives on the query "jaguar". On the left hand side there is a hierarchy of different topics of query. Users can click on the topic that they are interested in and see the documents of that topic on the right hand side. There is very little that we know about the technology other than that they use the snippets of the web pages to do clustering and labeling to generate this taxonomy [24]. A "snippet" is a fragment of a web page with the title and a short paragraph extracted from the page, either from the beginning of the text or a paragraph that has the query. A snippet usually contains a short summary of the web page [50]. CARROTSEARCH[3] is a similar system that claims to give accurate topical document clusters with clear labels without external knowledge bases and generate taxonomies on-the-fly. Figure 2.8 is an example of CARROTSEARCH′s result on the query "jaguar". These systems present about 10 clusters for a query on each level. From Figure 2.7 we can see that there are multiple clusters that actually belong to the same topic. For example, the clusters labeled "International", "Parts", "Sedan, Sales", "Luxury car", and "Sports cars" are all about the query sense "jaguar cars". CARROTSEARCH has the same issue of not combining similar topics. The taxonomies generated by

[3]http://carrotsearch.com/

these systems are not as compact as they should be.

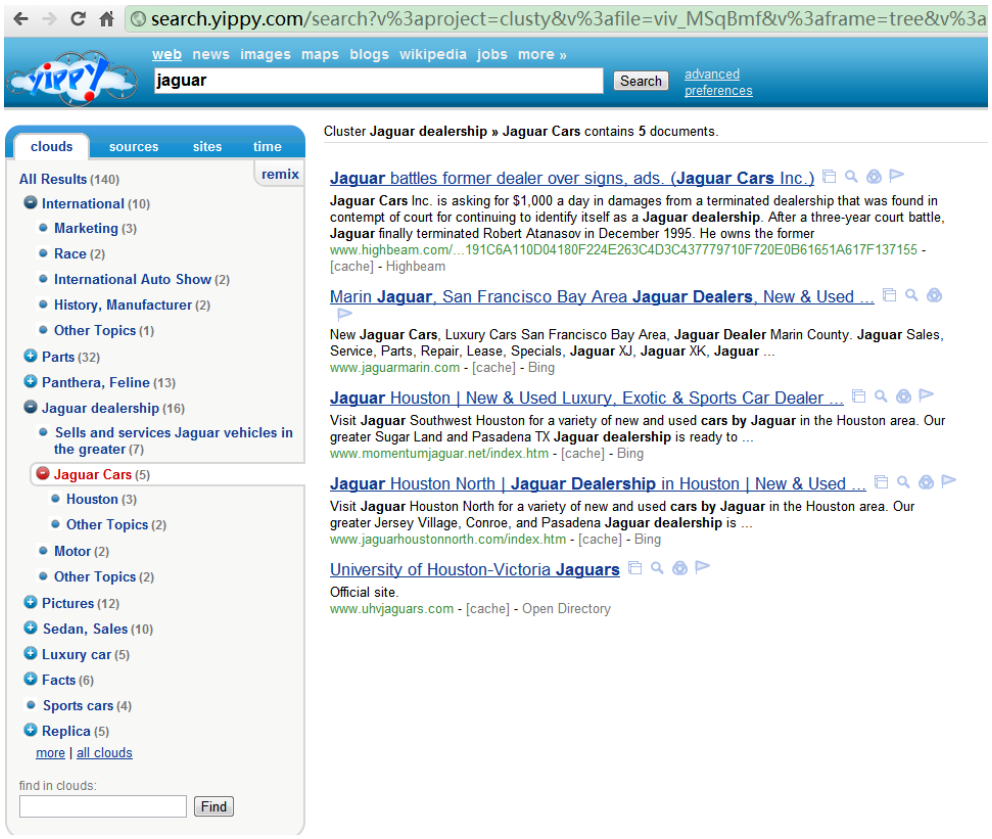

Figure 2.7: Yippy.com′s search result presentation on the query *jaguar*

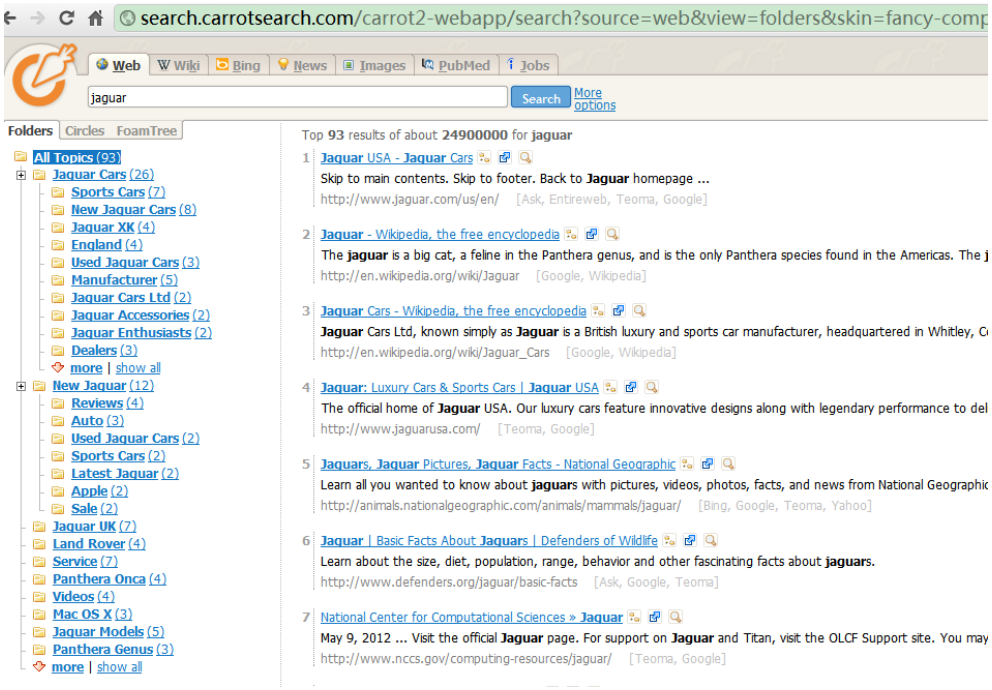

Figure 2.8: CARROTSEARCH's search result presentation on the query *jaguar*

Chapter 3

A survey of Automatic Taxonomy Generation (ATG)

As stated before, using document clustering and labeling to generate a hierarchy of topics and subtopics is also known as Automatic Taxonomy Generation (ATG) [71]. In this section, we first describe the desired properties of a good taxonomy, then we briefly review three major ATG categories including document-based, word-based, and co-clustering methods with some representative examples.

3.1 Desired properties of a good taxonomy

The goal of ATG is to organize a large number of documents into smaller, meaningful hierarchical document clusters, allowing users to effectively browse and navigate the document collection [75]. Before diving into the review of various ATG approaches, let us look at the desired properties of a good taxonomy, both from the clustering perspective, and from the labeling perspective. We develop our method aiming at building taxonomies that satisfy these properties.

3.1.1 Desired properties of Clusters in the taxonomy

Coherency Similar documents of the same topic should be grouped together and not scattered into different clusters. Also, since some documents may cover more than one topics, overlapping should be allowed in the taxonomy [71][77].

Coverage A good taxonomy should cover as many documents in the collection as

possible. Ideally, all the documents in the collection should be assigned to at least one of the clusters in the taxonomy. A document is considered an orphan document if it cannot be assigned to any of the clusters in the taxonomy. The fewer orphan documents, the better the taxonomy is [42].

Compactness A good taxonomy should not be too deep or too wide, in other words, it should be compact. Too many clusters may confuse the users and does not serve the purpose of efficient navigation [42].

Transparency In a good taxonomy a user should be able to tell why a certain document X is placed under a certain cluster Y [21].

3.1.2 Desired properties of Cluster Labels in the taxonomy

Cluster labels summarize the documents in the clusters. They are also very important in the taxonomies. In Ferragina and Gulli′s [24] user survey, 45 users used Vivismo.com (see Section 2.3) for 20 days. 72% considered " the ability to produce on-the-fly clusters in response to a query, with labels extracted from the text" as the most useful feature, 85% reported that meaningful labels "give a good sense of range alternatives". There are several desired properties of cluster labels:

Accuracy Cluster labels are what the users use to infer the topics of the clusters. They should be good indicators of the documents in the cluster, and the users should be able to tell why label X is selected for the documents in cluster Y [77][21].

Comprehensibility Unlike the labels of other kinds of networks such as a cluster of protein or a cluster of famous people, document cluster labels have special requirements. Document cluster labels are not just the most prominent nodes in the network. They should also be grammatically consistent, and meaningful to users [21]. Phrases usually satisfy this property. Even though sentences contain more information, a sentence is unlikely to be general enough to represent the whole cluster. Besides, randomly generalized sentences can be confusing to users.

Conciseness For the ease of browsing, the labels should be as short as possible so that the users can process as little information as possible. At the same time, they should contain enough information for the users to infer the topics of the clusters [77][21].

Small number of labels If the user is presented with a list of labels, the most descriptive ones should be up front. This way the user spends less effort in inferring the topics of the clusters [7].

3.1.3 Desired properties of both the cluster and the labels

There are several other properties that both the clusters and the cluster labels share.

Sibling Distinctiveness Each document cluster in the taxonomy, especially the ones at the top level, should represent a different topic in the document collection. Clusters at the same level of a taxonomy are sibling clusters. Users have a better navigation experience if the sibling clusters and labels can be easily distinguished from one another [42].

General to Specific Ideally the most general topics should be at the top level, as the level goes down the concepts should be more specific. That is to say, a parent cluster should be more general than its children clusters [42]. This applies to both the clusters and the labels. This property is hard to achieve, especially when the cluster sizes are small and there are no apparent general to specific relationships [50].

Speed Since one of the most important applications of ATG is search result clustering, it is crucial that the taxonomies can be generated on-the-fly so that the users would not have to experience obvious delays [77][13].

3.2 ATG approaches based on clustering documents

Traditional clustering methods can be applied here to cluster documents with the Vector Space Model (VSM). Each document is represented as an N-dimensional vector of features. A feature is a word or a phrase called a term, and the value can

be the frequency of the term in that document, or binary indicating if it occurs in the document. Various feature selection methods can be applied. Conventional clustering methods can then be used on these document vectors to group documents into clusters according to certain similarity measures. There are two main ways of clustering in previous research: partitioning and hierarchical clustering. Scatter/Gather, which is the pioneer in using document clustering as a browsing tool, uses the partitioning method [20]. Partitioning methods need to know the number or the size of the clusters in advance, but in real life we hardly have this information. Besides, it is not reasonable to assume that each cluster has roughly the same size. Hierarchical methods have an advantage in that they do not require this information but they are usually time-consuming. The time complexity issue is even more vital with documents than other kinds of objects since they usually have very high dimensions with hundreds and thousands of terms [13]. As for cluster labeling, the set of terms that have high frequency in the clusters can be cluster labels [71]. However, these terms tend to be too general to describe the topics.

Zamir and Etzioni developed the STC (Suffix Tree Clustering) algorithm which is different from traditional document clustering in that it is not based on VSM [77]. Instead of treating a document as a set of words, it treats it as a string. It works on the snippets of documents rather than the full text. It first discovers *base clusters* and builds a "suffix tree", then it merges them into document clusters. Base clusters are formed by grouping documents that share the same phrase (or a "suffix") in their snippets. The phrases in STC are only contiguous terms. Then STC forms a graph with the base clusters as nodes. An edge is formed between two base clusters when the size of common documents exceeds half of the size of documents in either cluster. Finally, it finds connected components in this graph and treats each component as a document cluster. The phrases in each component are treated as cluster labels. A distinguishing feature of STC is that it is linear in the number of documents when constructing the suffix tree. However, the graph construction is exponential in the number of phrases [40]. It means that STC is not feasible to work on full texts since the number of phrases in full texts is much larger than in snippets.

Ferragina and Gulli [24] built a hierarchical clustering engine called SnakeT following the STC algorithm and focused more on cluster labeling [21]. Instead of using just contiguous terms as phrases, they use non-contiguous phrases called *approximate sentences*. STC incorporates two knowledge bases: one is a collection of anchor texts that enriches the snippets, another is an ontology that helps ranking in generating the approximate sentences. The top k approximate sentences that occur in a sufficient number of documents are selected as cluster labels.

Many ATG methods based on clustering documents use snippets for computational reasons but it is obvious that snippets do not maintain all the information. Scaiella et al. pointed out that using snippets have the following shortcomings because they are very short [61]. First, it is hard to get meaningful labels. Second, many classic clustering methods do not work on them. Third, the problems with polysemy and synonymy are even more severe with snippets. Sanchez et al.′s experiments on clustering search results also show that with three different clustering methods, using full text achieve better document clustering results than using snippets [59].

3.3 ATG approaches based on clustering words

Word-based ATG approaches aim at organizing words by theasural relationships [40]. There are many works on automatically generating thesaural relationships between words from a corpus. Some are based on phrasal analysis that investigates the context of a term [31][73]. Other methods use co-occurrence to establish relationships between terms [60][43][39][42]. We briefly describe some methods that are suitable for our task of generating hierarchical clusters.

Sanderson and Croft proposed the subsumption algorithm [60]. Selected terms from a corpus are treated as *concepts*. They build a concept hierarchy by finding pairs of concepts where one subsumes another in a bottom-up fashion. The subsumption relationship is determined by the relative document frequencies of the concepts.

DisCover is proposed by Kummamuru et al. [42]. This monothetic clustering

algorithm is based on the Coverage, Compactness, and the Sibling Distinctiveness properties of good taxonomies discussed in Section 3.1. A document cluster is described by a single feature that all the documents share. The features are then used as cluster labels. DisCover compares itself with two other word-based ATG algorithms: DSP [43] and CAARD [39], and claimed that it has the best hierarchy according to user surveys [71].

Another example is the J-Walker built by Cui and Zaiane [19]. It indexes documents by the nouns (keywords) in their snippets. Given a keyword, documents that contain this keyword are grouped together. It then builds a concept ontology based on WordNet[1]. WordNet is a lexical base that contains "IS A" semantic relationships for nouns. A precedent of a word in the WordNet ontology is its hypernym. In the concept ontology of J-Walker, each leaf node is a keyword with its corresponding document cluster. Each path in this ontology goes from a leaf node, then through all its hypernyms in WordNet, finally up to the root which is the query. The ontology is later trimmed to keep only the necessary concepts.

In general, ATG approaches based on clustering words first generate a concept hierarchy where each concept is a single feature, and then assign documents to the concepts.

3.4 ATG approaches based on co-clustering

There is another ATG approach which is based on co-clustering. Basically, it selects terms from the documents as keywords. Then it clusters the keywords, and at the same time generates document clusters. Similar to using VSM to represent documents as N-dimensional vectors, a keyword can be represented as a M-dimensional vector where the $l-th$ feature is the frequency of the keyword in the $l-th$ document [40]. Co-clustering ATG methods aim at grouping documents with similar keyword distributions together. A co-cluster is a document cluster with its corresponding keyword community [71]. Some examples are FCoDoK that uses weighted cosine distance to rank keywords for labeling[41], FSKWIC does not mention how to label

[1] http://wordnet.princeton.edu/

the clusters[28], and RPSA that uses average TFIDF value over the documents in a cluster to rank the keywords for labeling[49]. The first two give only flat structures, and RPSA generates a hierarchy of clusters.

Apart from these algorithms, Dhillon developed a co-clustering ATG algorithm based on bipartite graph partitioning [22]. It represents a document collection as a bipartite graph where one set contains the documents and the other set contains the keywords. It generates the co-clusters by finding the minimum k-cut vertex partitions. They use a spectral algorithm to solve a real relaxation to this NP-Complete problem. The terms with top internal weights inside each keyword community are cluster labels.

In data mining, co-clustering means to cluster the rows and columns in a matrix simultaneously. However, since the ATG approaches based on bipartite graph partitioning are already categorized as co-clustering, we refer to methods that generate keyword communities and documents clusters together as ATG methods based on co-clustering as well [71]. Recently, Chen et al. proposed a co-clustering method that builds a keyword graph based on the co-occurrences of the keywords [14]. A keyword is a phrase from the documents that they identified with simple heuristics, an edge is formed when the two nodes co-occur in the documents, and the edge weight is the co-occurrence. Then, they use the K Nearest Neighbor (KNN) algorithm on the keyword graph to find keyword communities. It maps documents based on the similarity between a document and a keyword community and thus form document clusters. The keywords with top weights in the keyword communities are selected as labels. They claim that this labeling strategy works well in representing search results. However, they do not have statistical evaluations of the labeling. There are also researches that consult external knowledge base to get keywords. Scaiella et al. use a Wikipedia annotator TAGME to find the Wikipedia page titles associated with each document snippet [61]. In their keyword graph, a node is a Wikipedia page title (*topic*), the edge weights are the topic-to-topic similarities computed based on the Wikipedia linked-structure. Then they bi-section the keyword graph until it creates about 10 clusters or no cluster has size over a threshold. This method only works on snippets because TAGME is time-consuming.

The method of this work falls into this category of co-clustering. We aim to do ATG based on full-text and on-the-fly. By generating a keyword graph and mining communities from the graph we avoided the high-dimensionality in traditional document-based clustering methods and also maintain all the important information from the documents. The community mining on the keywords automatically detects the number of topical coherent keyword communities. Documents are then mapped to the keyword communities and thus document clusters are generated. By choosing labels from a group of keywords we are able to describe a cluster with multiple topics while word-based clustering methods only generate one feature for each cluster.

Part II

Methodology

Chapter 4

Our approach

Our approach that builds the taxonomy of documents for better representation falls into the category of co-clustering as described in Section 3.4. The basic idea is to reformulate the document clustering problem into a query sense community mining problem. *Clustering* and *community mining* both partition a set of objects into several groups. The difference is that clustering is based on the attributes of the objects whereas community mining is based on the relationship between the objects. In our task, rather than clustering the documents, we extract keywords from them to build a graph indicating the sentence co-occurrences of the keywords. We do community mining on this graph to get communities of highly co-occurring keywords. Then, we map the documents back to the keyword communities to form document clusters. This way we keep the important information without being too time-consuming. There are, in fact, parts in our approach that are time-consuming, but these steps can be done off-line along with the crawling and thus not affect the user experience in browsing.

The process of our approach is shown in Figure 4.1. The "crawled documents" are the pre-processed documents and the "ranked list of documents" is a list of documents. The taxonomy can also be easily updated if any changes occur in the collection. Besides search results, our method also applies to any document collection with different topics. We follow the major phases in Chen et al.′s work [13] with some revisions on community mining and refinement, and focusing mainly on the cluster labeling phase which we will discuss with more detail in Section 4.5. All the phrases in our approach work as if they are in a black box. The user sends

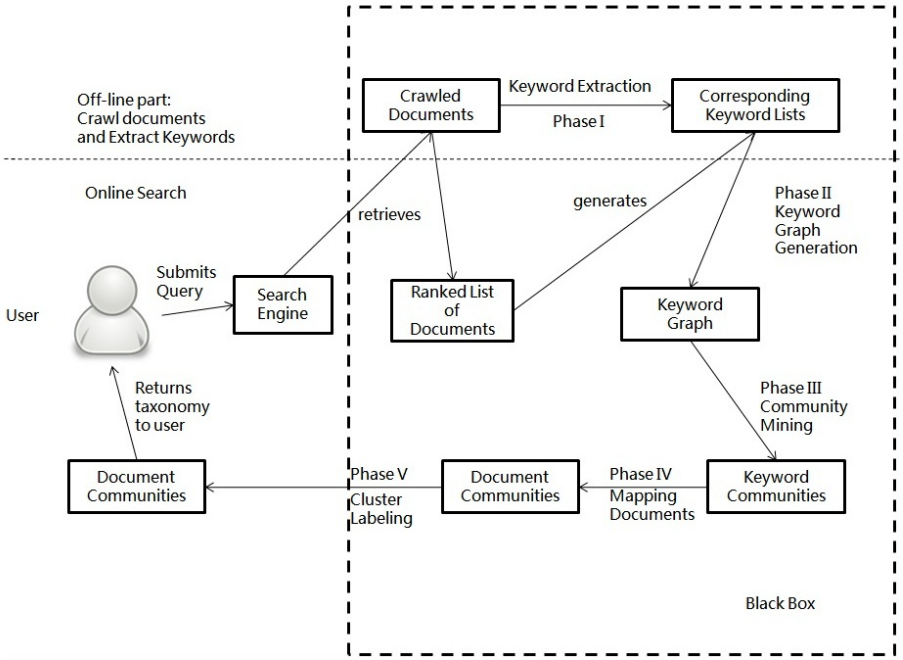

Figure 4.1: General procedure of our approach

a query to a search engine and will see a taxonomy of the query senses along with the documents. The major phases in our approach are:

I Keyword extraction from the crawled documents. Keywords are terms that are representative enough for the document collection. They are not necessarily the most important words from each document.

II Building a keyword graph. In this phase we build a keyword graph with selected keywords extracted in phase I.

III Community mining and refinement. In this phase we find different communities on the keyword graph from phase II. This process can be done hierarchically to generate a taxonomy. Based on the structure of the mined communities, we do community refinement to delete noisy communities and merge communities of the same topic if necessary.

IV Mapping documents to keyword communities and generate document clusters. In this phase we map each document into one or more keyword communities found in phase III.

V Cluster Labeling. Representative labels for each cluster are generated in this phase.

4.1 Phase I: Keyword Extraction

Given a collection of documents, we first extract keywords from each document. Some of the keywords will be selected as nodes in the keyword graph in phase II and act as candidate cluster labels. They should be good indicators of the content of the documents. In this work we choose Noun Phrases as keywords. The reason is that comparing to single words, phrases have more meanings and are more precise. They also satisfy the Comprehensibility property discussed in Section 3.1.2. Comparing to N-grams which are sequences of N words, Noun Phrases which only contain Adjectives and Nouns are less noisy and are more valuable in representing meanings [51]. It has also been observed that Noun Phrases give better results in cluster labeling [50].

Note that this step of keyword extraction is usually time-consuming and should be done off-line (comparing to the other phases which should be done online). They can be extracted when the web pages are crawled, the extracted keywords can be stored and indexed with the crawled web pages. In this work, in particular, we do not have the resource to crawl web pages and index them as a search engine. For practicality, this step is done as follows: given a query q, we send requests to Google and retrieve top k results of this query, we parse the web pages to get the main texts, and then extract keywords from the texts. How we collect the web pages and parse them is discussed with more detail in the data collection and pre-processing section (see Section 5.1).

We extract Noun Phrases from a document through the following steps: 1. Part of Speech (POS) tagging, 2. Lemmatization, 3. Pruning, and 4. Noun Phrase extraction.

4.1.1 Part-Of-Speech (POS) Tagging

First we do Part of Speech tagging to tag each single word from the document with its part-of-speech. We use the Stanford POS Tagger with an English tagging model *left3words-distsim-wsj*[1]. We do this first because we do not want the following steps to affect the results of POS tagging. Note that this step is very time-consuming. For example, in one of our experiments, it took 279 seconds to tag 90 web pages.

4.1.2 Lemmatization

We lemmatize all the words to reduce the inflectional forms. English words usually have more than one form with the same semantic meanings, for example, *car* and *cars*. To reduce the forms to their base forms helps us in building the keyword graph and the community mining process later. Both stemming and lemmatization could achieve this goal. Many researches use stemming because it is easy to do. Stemming methods usually just chop off the end of words according to a set of brutal heuristics. Lemmatization, on the other hand, is more reasonable. It utilizes dictionaries and morphological information, aiming to remove only the inflectional endings rather than chopping a large part off from the words [50]. For example, the word *large* is stemmed to *larg* with the famous Porter Stemmer but it is kept intact with the WordNet Lemmatizer. Therefore, in this work we use the WordNet Lemmatizer provided with the Natual Language Toolkit (NLTK)[2] to get the lemma for each word. A lemma is usually the base form of a word, just like how the word would appear in a dictionary. We also store the original words along with their lemmas. At the post-processing stage discussed in Section 4.5.7, we transform the lemmas back to their most frequent original form so that they make more sense to the users [72].

4.1.3 Pruning

We remove all the stop words from the text. Stop words are words that are very common in every document. For example, words like “you”, “and”, “get” are stop

[1] http://nlp.stanford.edu/software/tagger.shtml
[2] http://nltk.org/

words because they have no discrimination powers of topics. We use a long stop word list with 669 common English words [3]. Besides, in the experiments we have found that terms like "time", "year", and "Monday" etc. appear in many documents no matter what the topic is. The reason is that web documents usually contain the date that they are published. Even though that these words are not in the stop word list, they do not represent the topics and we do not want them to be the labels. Therefore we customize the stop word list by adding 35 more words to it. We also ignore all words that contain non-alphabetic characters [62]. Moreover, to avoid duplications, the first word of a sentence is converted to lower case if it is capitalized.

4.1.4 Noun Phrase Extraction

We extract Noun Phrases based on a lexical heuristic. The rule we use is (Adjective).*(Noun).+. It means that we consider a word or phrase with zero or more Adjectives with one or more Nouns following them as a Noun Phrase [46][47]. We do not use a phrase Chunker because they need to be trained with certain corpus and are not suitable for general documents such as web pages. Also, we set the maximum length of a phrase to be three words. The reason is that the longer the phrase is, the less representative it is. One of our early experiments compares the clustering quality with keywords under the maximum length of 3 and 4, and found that the former gives better results.

In order to save time in building the keyword graph for community mining in the next phase, in this phase we represent each document as a list of pairs of keywords along with how many times these two keywords have co-occurred in a sentence.

4.2 Phase II: Keyword Graph Generation

At this point, each document is represented as a list of pairs of keywords. Each pair of keywords have co-occurred in one sentence. As shown in Figure 4.1, this is the first online step in our approach. Assume a search engine receives a query, it

[3] http://www.ranks.nl/resources/stopwords.html

retrieves a ranked list of k documents along with k keyword pair lists corresponding to these documents. In this phase we use these k keyword lists to generate a keyword graph.

A node in this graph is a keyword we extracted from Phase I. An edge in this graph means that the two nodes have appeared in at least one sentence together. The edge weight is the number of times these two nodes co-occurred in a sentence in all the documents. The key assumption in forming edges using co-occurrences is that words describing the same topic are often used together. Co-occurrences have been shown to carry useful correlation information and be able to identify different topics [11][74][63]. For example, "engine" and "wheels" often occur together to describe the topic of "cars". Different topical groups can be discovered from a proper keyword graph.

4.2.1 Node and edge selection

It is not practical, nor necessary to use all the keywords as nodes in the keyword graph. Words that only appear very few times in the documents are usually not good indicators of a topic and they add noise to the graph. Moreover, too many nodes would generate a large graph that slows down the system. In this work we use Document Frequency (DF) to select keywords as nodes. A term t's DF in a document collection D is the number of documents that has t divided by the total number of documents in D. More formally it is:

$$DF(t, D) = \frac{|d \in D : t \in d|}{|D|} \tag{4.1}$$

where $|D|$ is the total number of documents in the collection and $|d \in D : t \in d|$ is the number of documents that has t in them.

Only keywords with DF higher than a threshold t_{df} will be selected as nodes in the keyword graph. The selection of t_{df} will be discussed in Section 5.3.1.

One would suspect TFIDF (term frequency inverse document frequency) to be a good measure for node selection. TFIDF measures a term t's specificity to a document d in a document collection D. It takes two factors into account, term frequency $TF(t, d)$ and inverse document frequency $IDF(t, D)$. $TF(t, d)$ is the

number of times t appears in d divided by the total number of terms in d. $IDF(t, D)$ is the inverse of $DF(t, D)$, it is custom to take the log of the inverse to avoid overflow. More formally:

$$IDF(t, D) = \frac{1}{DF(t, D)} \; and \; TFIDF(t, d, D) = TF(t, d) * IDF(t, D) \quad (4.2)$$

The reason why we use DF to measure the suitability of choosing a keyword as a node rather than TFIDF is that TFIDF measures the importance of a word in a certain document locally rather than in the whole document collection globally. Chen et al.′s experiments showed that selecting nodes according to DF provides better clustering results than using TFIDF, and using IDF provides the worst results. [13].

Moreover, keywords that are exactly the same as the query *q*, or are contained in *q* will not act as a node. For example, for the query "Michael Jordan", we do not use "Michael", "Jordan", or "Michael Jordan" as nodes in the keyword graph because they obviously cover all the topics that "Michael Jordan" covers. Also, they are linked to words of all the query senses and would add noise to the community mining[13][21].

Then we use the keyword lists from Phase I to connect the selected nodes and add edge weights. Nodes with no edges are removed because they have no connection with other nodes and do not help in distinguishing topics.

4.2.2 Other attempts on node and edge selection

We have described why and how we use a DF cutoff to select nodes and edges for the keyword graph from the full graph. We have suspected various issues caused by the DF cutoff method and have tried other node and edge selection methods. In this section we list these methods and show that none of them is suitable for our task.

By Average Weight and Average Composite Weight

Even after deleting nodes according to their DF, and deleting the corresponding edges, we still have many edges that only have the weight of 1. For example, in the *jaguar* keyword graph, there are 5506 edges in total, 3187 of them have an edge

weight of 1. One would assume that these are not strong connections. Therefore, we also tried using the Average Weight (AW) and the Average Composite Weight (ACW) in Chen et al.′s work to cut off insignificant nodes and edges from the full graph $G(V, E)$ [14]. In this method, edges with weight smaller than a threshold AW, and nodes with weight smaller than a threshold ACW are removed from the keyword graph.

More formally, AW and ACW are defined as follows:

$$AW = \frac{\sum_{r_{ij} \in E(G)} r_{ij}}{n} \tag{4.3}$$

where n is the total number of edges in G, and r_{ij} is the weight of the edge e_{ij}. AW is the average edge weight in the graph G.

For nodes, the weight of a node is called Composite Weight (CW) and it is defined as the sum of the edge weights that it connects with divided by the number of its neighbors. More formally, the CW of a node i is:

$$CW_i = \frac{\sum_{j=1}^{m} r_{ij}}{m} \tag{4.4}$$

where r_{ij} is the edge weight of the edge e_{ij}, and m is the number of nodes that node i connects with.

The ACW is the average node weight, formally it is defined as:

$$ACW = \frac{\sum_{i=1}^{k} CW_i}{k} \tag{4.5}$$

where k is the total number of nodes in the graph G.

After applying this method the graph changed dramatically since many connections have been cut and the graph is not as connected as before. For example, a graph on a query with 301 nodes and 3119 edges is condensed to 102 nodes and 412 edges after this method with an AW of 1.82 and an ACW of 1.58. Community mining on this graph renders way more communities than we want. Besides, the process of calculating AW and ACW is very time consuming.

By qualified neighbors

After the DF cutoff, we have found that a very large number of nodes have been removed. For example, in a collection of 500 documents, there are 12,997 keywords but only 994 made it as nodes in the keyword graph. About 92% keywords have been deleted. We then try to keep the keywords that have DF smaller than t_{df} but with all the neighbors' DF larger than t_{df} to get more candidate labels. However, the experiments show that this changes the keyword graph largely on the number nodes but very slightly on the edges. For example, a graph with 994 nodes and 34985 edges will increase to 1724 nodes and 37255 edges. The reason is that many of these nodes only connect with one or two nodes and more likely to be outliers. We also observed that this badly impacts on the community mining results.

4.3 Phase III: Community Mining

In this step we do community mining on the keyword graph to detect different topical keyword communities. Any community mining algorithm can be applied here, specifically we use the Fast Modularity algorithm which is based on one of the most well-known community mining metrics: Modularity Q [16][53]. After this step we will map documents to the keyword communities to generate document clusters. The mapping of the documents will be discussed in Section 4.4.

4.3.1 Modularity

Modularity Q was first introduced by Newman and Girvan to measure the quality of a network division [54]. It basically compares the division of the graph with a "null model" of the graph that has the same nodes, same node degrees, but randomly generated edges. Modularity considers the differences between the fraction of edges that fall within the detected communities in the real graph's edges, and that in the null model's random edges. More formally it is defined by Equation 4.6 where *m* is the total number of edges, *A* is the adjacency matrix, d_i is the degree of node *i*, and $\delta(i, j)$ is a boolean value indicating if node *i* and node *j* are in the same community.

$$Q - modularity = \frac{1}{2m}\sum[A_{ij} - \frac{d_i d_j}{2m}]\delta(i,j) \quad (4.6)$$

In this work we extend Modularity to weighted graphs. It still uses the same equation with standard Modularity as shown above but here, *m* is the total weight in the network, *A* is the weighted adjacency matrix, d_i is the total weight of all edges node *i* connects with, and $\delta(i,j)$ remains the same.

Typically, a modularity of 0.3 to 0.7 indicates strong community structure [54].

4.3.2 Fast Modularity algorithm

We apply an algorithm named Fast Modularity for community mining in the keyword graph developed by Clauset, Newman and Moore [16]. It greedily optimizes the modularity score in the graph partitioning in an agglomerative manner. It first treats each node as a single community, then it merges a pair of nodes if merging them increases the overall Q score the most. The algorithm ceases when there is no such pair [13].

Modularity-based community mining algorithms are usually slow ($O(n^3)$ where n is the number of nodes). Since community mining is one of our online parts in generating the taxonomy, acceptable running time is crucial. Clauset et al. have worked on this issue and made a Fast-Modularity algorithm[4] that has running time of $O(nlog^2 n)$ [17]. Besides, in our approach we have selected the nodes in the keyword graph carefully as discussed in Section 4.1. The size of the graph usually stays at hundreds of nodes and a few thousands of edges. The running time of this phase in our experiments is just a few seconds on a PC. Given the resources such as a search engine service, this step is fast enough.

One of the reasons why we apply the Fast Modularity algorithm is that it automatically detects the number of communities. This is important in our task because a good taxonomy should be compact, as discussed in Section 3.1.1. In this aspect, our approach has advantage over existing commercial systems such as CarrotSearch and Yippy.com (see Section 2.3), and also some most recent researches of Scaiella

[4]http://www.cs.unm.edu/ aaron/research/fastmodularity.htm

et al. and Chen et al. [14][61]. Their methods partition the document collection to about 10 clusters which is not always the number of real topics.

Modularity based community mining algorithms have their flaws. Fortunato and Barthelemy pointed out that they are not able to detect small communities that are below some threshold [26]. It is also not clear how they detect outlier nodes [23]. For our task, however, these flaws are tolerable. In the keyword graph, if a community contains very few words, it usually means that it is not a strong topic. We can afford to ignore very small keyword communities and outliers.

4.3.3 Recursive community mining

To generate a taxonomy, we apply the Fast Modularity algorithm recursively in a top-down manner until certain conditions are reached. This way fewer errors are introduced to the top levels that the users see first. A node that has no child is referred to as a leaf node in our taxonomy. We assign a community as a leaf when its modularity is smaller than a threshold t_Q. Since modularity measures the strength of the community partitioning, we also use it to measure the need for further splitting [13]. The selection of the threshold t_Q is discussed in Section 5.3.3. Additionally, a community is considered as a leaf node if none of its components′ size is greater than 5 [36]. Although we are doing community mining on the keyword graph, the ultimate goal is to generate a hierarchy of document clusters. After the mapping stage in Phase V, if a community is mapped with less than 5 document, it will be considered as a leaf node as well.

4.3.4 Community Refinements

After community mining, we further refine the community structures following Chen et al. [13].

Noisy communities should be deleted. According to Chen et al., small communities are usually noisy communities. A very small set of keywords is usually not representative enough for a topic. They have also observed that small communities are sometimes formed by words that always co-occur no matter what the topic is [13]. We delete small communities that have fewer than 5% of the total node num-

ber of its parent [13]. We also delete communities that contain zero or only one document after mapping in phase III because they are too small to be meaningful to the users. We delete clusters with fewer than 2 documents rather than a larger number, say 5, documents because we want to minimize the risk of throwing away small communities that are actually about a distinct topic.

Communities of the same topic should be merged. Communities that have a large overlap in the documents that they cover are usually of the same topic [13]. In this step, given a community c, if a document d's overall TFIDF score in c is larger than a threshold t_{merge} (discussed in Section 5.3.2), we say that c covers d. Given two communities, we calculate the document set that they cover, if the overlap in these two document sets is more than half of one of the sets, we merge the two communities.

There are other methods for community refinements. Chen et al. merge communities when the connection between two communities is stronger than the connections within the communities themselves [14]. This method is not reasonable because should these cases happen, the community mining algorithm itself has mistakes in generating such clusters. We have also tried this method in our early experiments and it does not merge communities of the same topic. Another way is to merge communities according to semantic similarity. If two communities have strong semantic similarities between their keywords, they should be merged. However, the calculation of semantic similarity is too time-consuming to do online. Besides, the similarity measures themselves are questionable as discussed in Section 4.5.6.

4.4 Phase IV: Mapping Documents

At this point we have discovered different keyword communities. In this phase we assign the documents to the keyword communities to generate document clusters. In order to do this, we introduce a document′s overall TFIDF score in a community. Given a document d and a keyword community c, d's overall TFIDF score in c is the sum of the TFIDF scores of all the keywords that are both in d and in c. For

all the keyword communities, we assign d to the one that has the highest overall TFIDF score s. Besides, since some documents may contain various topics, it is desired that the system allows one document to be assigned to multiple keyword communities as discussed in Section 3.1.1. Therefore, if d's overall TFIDF score in another community c' is higher than $0.9 * s$, we consider it to be a small difference and assign d to c' as well. In this way each keyword community is associated with a document cluster.

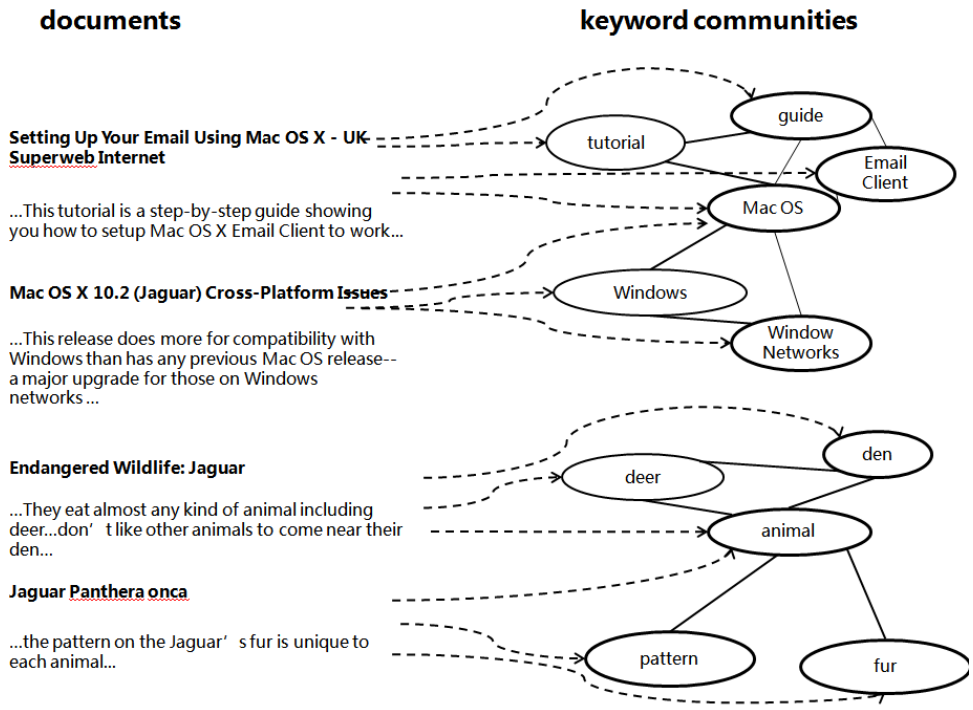

Figure 4.2: Illustration of Phase IV: mapping documents

This procedure is illustrated in Figure 4.2 on some document examples from the query "jaguar". On the left are some documents with bolded titles and a piece of the documents, on the right are some keywords extracted from them in Phase I. A dashed line represents the connection between a document and a keyword, the weight is the TFIDF score of that keyword in that document. The solid lines are the edges in the keyword graph, they represent the co-occurrences of the nodes. One can easily detect two keyword communities on the right, one is about "Mac OS" and one is about "animals". For each document on the left, we calculate its overall

TFIDF score in each of the keyword communities and assign it to the keyword communities accordingly.

There are other ways aside from using overall TFIDF to map documents. One naive way is to assign a document to any keyword community that has at least one keyword in common with it. However, this method renders highly overlapped document clusters. In a data set of 700 documents, about 500 documents have been mapped to more than one community whereas our overall TFIDF method multi-mapped 46 documents. In a general document collection or in web search results, it is unlikely that about 70% documents have multiple topics. Therefore we stick with mapping documents using overall TFIDF.

4.5 Phase V: Cluster Labeling

The final step, which is the focus of this work, is to label the document clusters we get from Phase IV. Each keyword from a community serves as a candidate label. This step aims at finding suitable labels for each community that have the desired properties such as Accuracy, Comprehensibility, and Conciseness as described in Section 3.1.2. For our interactive browsing task, labeling is even more important than clustering since the labels are what the users see first in the taxonomy.

In our work, the labeling of the clusters is independent of the clustering process. In this section we first briefly review the cluster labeling methods in the literature. Some of these methods are used as baseline to compare our labeling methods with. Then we introduce the labeling methods we propose and the baseline methods. We developed various labeling methods based on the keyword communities, the document clusters, and also the connections between a keyword community and its corresponding document cluster.

4.5.1 Related Work in Cluster Labeling

The most common cluster labeling method is to use the most frequent or the most central phrases in a document cluster as labels [50][68][67]. For example, the classic Scatter/Gather algorithm uses terms with the highest weight in the cluster cen-

troid document as cluster labels [20]. In a recent research of Chen et al., the terms with the highest co-occurrence scores are used as labels [14]. It is the same as using the degree centrality in the keyword community which we will use as a baseline. The keywords with top weights in the keyword community are selected as labels. They claim that this labeling strategy is effective in representing the cluster concepts and that it facilitates the visualization of the search results. The performance of these methods is actually not promising since their labels tend to be general and not descriptive of the clusters [67].

Some methods label clusters based on the distribution of the terms in the clusters. For example, Glover et al. propose to use the relative rareness of a word in a cluster to detect labels [30]. They use anchor texts and extended anchor texts to extract candidate labels which is very expensive. Treeratpituk proposed a modification of Glover et al.′s algorithm with a *descriptive score* based on TFIDF [68]. Popescul and Ungar proposed the "Frequent and Predict Words" method that detects terms that are more likely to appear in a cluster than in other clusters as labels [58]. In our experiments, we use this method as one of the baselines.

There are also methods that utilize external sources such as WordNet or Wikipedia to determine the relationship between different terms [15][29]. For example, Tseng et al. try to find the "proper" hypernym of a set of terms to use as cluster labels [69]. In our experiments we have discovered that the hypernyms found by this method tend to be too general to be cluster labels. Chen et al. use a dependency word similarity measure based on WordNet (Lin-similarity) to find the terms with high similarities with other terms in the cluster as labels [13]. Carmel et al. finds the relevant Wikipedia pages of the documents in the document cluster and then use the meta-data such as the title and the category information to do labeling [7]. Scaiella et al. try to find the Wikipedia pages associated with each document and use the page titles as cluster labels ranked by the confidence of the connection of the document and the Wikipedia pages [61]. External sources do enrich the candidate pool of labels. They are not limited by just the terms in the documents. However, due to the manual nature, these sources may not cover all the topics in the documents, especially the web pages. Besides, the sources such as WordNet that are built by

linguistics contain terms that are hard to understand by regular users and the ones that are freely edited by anyone such as Wikipedia contain noises.

4.5.2 Label selection from the keyword community

First, we try to find important terms from the keyword communities as labels. Centrality measures how important a node is in a network. Here we try to use the most central keywords in a keyword community as labels. Among the common measures, degree centrality is used by Chen et al. in their recent work as the sole labeling method [14]. We use it as a baseline.

Degree Centrality

This simple centrality measures the importance of a node by its degree. The degree of a node is the number of edges that tie to it [27]. On weighted networks, the degree of a node is defined as the sum of the weights of all the edges that tie to the node [4]. More formally, the degree centrality of a node n is:

$$C_d(n) = degree(n) \tag{4.7}$$

In our context, degree centrality measures the number of times a keyword co-occurs with other keywords. We rank each keyword in a keyword community by its degree centrality and select the top ones as labels.

Betweenness Centrality

Betweenness centrality reflects a node′s influence on the communications between other nodes in the community. It measures the number of shortest paths between other nodes that goes through a certain node. More formally the betweenness centrality of a node n in a community defined as:

$$C_B(n) = \sum_{i \neq j \neq n \in N} \frac{\sigma_{ij}(n)}{\sigma_{ij}} \tag{4.8}$$

where i, j, n are nodes in a community, σ_{ij} is the number of all the shortest paths from i to j, and $\sigma_{ij}(n)$ is the number of shortest paths between i and j that runs

through n. On weighted networks, the lengths of the shortest paths are usually defined as the sum of the inverse weights on the edges [6]. It takes $\Theta(|V|^3)$ time with a modified Floyd-Warshall algorithm to calculate Betweenness Centrality.

The intuition of using betweenness centrality for labeling is that sometimes terms of the same topic may not directly co-occur in a sentence. They may be connected by other terms. Keywords that play a vital role in connecting other terms may be important. Moreover, betweenness centrality takes the global structure of the network into account whereas degree centrality only measures the local importance of a node [56]. Closeness Centrality is also a popular centrality measure. It is the inverse of a node′s total distance to the other nodes in a network. It measures how close a node is with the other nodes but it does not directly measure how well it connects the other nodes. To keep it simple we only use Betweenness Centrality in labeling.

PageRank on keyword communities

There is another centrality measure, Eigenvector Centrality, that measures the influence of a node in a graph. PageRank proposed by Page and Brin is a variation of it [57][3]. PageRank is a graph based ranking algorithm originally aimed at link analysis. It has been successful in ranking web documents and in social network analysis. Mihalcea and Tarau proposed the TextRank model that applies PageRank on text processing [52].

Formally the TextRank model is as follows. Suppose $G = (V, E)$ is a directed graph with V as the set of nodes, and E as the set of edges. For a node V_i, $In(V_i)$ is the in degree (number of edges pointing to V_i) and $Out(V_i)$ is the out degree (number of edges pointing out from V_i). The TextRank score of V_i is:

$$S(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{4.9}$$

where d is a damping factor. Following conventions, in our experiments we set d to 0.85 as in PageRank [57].

While PageRank is designed for un-weighted graphs, TextRank has a scoring scheme for weighted graphs since there is a strength w of the connection between

two nodes in natural language. More formally the score of a node V_i on a weighed graph is defined as follows:

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j) \tag{4.10}$$

In our keyword graph where an edge indicates the strength of co-occurrences between two nodes, we use the weighted and undirected version of the TextRank model. In the undirected version of the scoring scheme, the in degree is equal to the out degree for each node. We use the TextRank model to rank the keywords in a keyword community and select the top ones as cluster labels.

4.5.3 Label selection from the document cluster

Besides choosing labels based on the keyword community structure, we also extract labels based on the document clusters. A conventional and popular way is to rank the terms by their frequency in the document cluster. In this work we propose to incorporate keyphrase extraction, and automatic titling from documents in cluster labeling. Given a document cluster, the main idea is to first extract important terms from each document, then rank each term by its importance in the document cluster, and finally take the top terms as labels. Also, only terms that are in the corresponding keyword communities are candidate labels. In this section we first describe two frequency-based labeling method using DF and TF. They are later used as baselines in our experiments. We then describe some important term extraction methods that we use for cluster labeling.

Using DF as a labeling method

This is a conventional cluster labeling method based on Document Frequency (DF). Given a document cluster, all the keywords from these documents are ranked by their document frequency in this document cluster. The top ones are selected as labels. We use this as one of the baselines for cluster labeling.

Using TF as a labeling method

Rather than just count the number of documents a term appears in, using Term Frequency (TF) also considers the local importance of a term in each document. This method ranks all the keywords in a document cluster by the sum of their TF value in the documents in that cluster. We use this as yet another baseline.

TFIDF to extract important terms from documents

As described in Section 4.2.1, the TFIDF measure represents the specificity of a term in a document. Here we use it as a way to extract important terms from each document in a document cluster. Up to 30 terms with the highest TFIDF values are treated as important terms for each document in a cluster. They are then ranked by the sum of their TFIDF values in each document in that cluster. The top ones are cluster labels. We use this as a baseline as well.

Keyphrase Extraction from documents: KEA

Keyphrases are short summaries of the content of a document that provide semantic meta data for the documents [72]. For example, "Automatic Taxonomy Generation" and "cluster labeling" can be the keyphrases of this thesis manuscript. Here, we use them as the "important terms" for cluster labeling. We utilize a famous and effective automatic keyphrase extraction algorithm named "KEA". It trains a machine learning model for keyphrase extraction based on several features:

1. TFIDF: As described in Equation 4.2, TFIDF measures a term′s importance to a document in a collection.

2. First occurrence: the number of words before a phrase′s first appearance in the document divided by the total number of words in the document [72].

3. Length: the number of words in a phrase [45].

4. Node degree: the number of semantically related phrases of the phrase in question [45].

The first two features (TFIDF and first occurrence) are from the original KEA algorithm, the latter two features (length and node degree) are additional features from an updated KEA package [72][45].

KEA takes a set of documents along with their keyphrases picked by human as training set to tune the parameters of these features and generates a model for automatic keyphrase extraction. We use a publicly available KEA package[5] to do training and testing for keyphrase extraction. The training takes time but it will not affect the user experience because it only needs to be done once, and it can be done offline. The KEA package provides a training corpus that contains 25 publications. Since we aim at web pages on general, we also add 25 web pages from Wan and Xiao′s corpus for keyphrase extraction in training [70]. Wan and Xiao′s corpus contains news web pages, each with hand-picked keyphrases. We add these documents to the training corpus that comes with the KEA package and trained a keyphrase extraction model with the package′s default settings with the exception of vocabulary. We choose to use free indexing without a controlled vocabulary to select keyphrases from because we aim at general texts, not domain specific documents. Using this model, we extract up to 30 keyphrases from each document in a community. Since KEA does not give weights to the keyphrases in the documents, we rank each term by the number of documents where it is a keyphrase. The top 20 terms are selected as cluster labels.

Automatic titling

A title of a document introduces the topic of the document. Titles inform the users of the semantic contents of the documents [48]. They can serve as "important terms" of each document in our cluster labeling task. We use Lopez et al.′s automatic labeling algorithm that concerns morphosyntactic characteristics gained from statistical studies [48][47]. They claim to be the first scientific study that leads to automatic titling application. They focus on Coverage Rate, which is based on the frequency of title words at different segments in the document. By studying three newspaper corpuses that have titles assigned by authors they have found that the

[5] http://www.nzdl.org/Kea/download.html

coverage rate is highest at the beginning of the documents, and then it decreases, and slightly increases at the end of the documents (see Figure 4.3). They select Noun Phrases (NP) from the first two sentences of each document as candidate titles since they cover 73% of the title′s semantic content. They are ranked by their TFIDF values and the top ones will serve as titles.

In our work, we select all candidate titles according to their method as important terms from each document. In addition, since the focus of our task here is to extract important terms rather than automatic titling, we also include NPs in a document′s title if it already has one. We rank each NP by the number of documents where it serves as a title. The top ones are cluster labels.

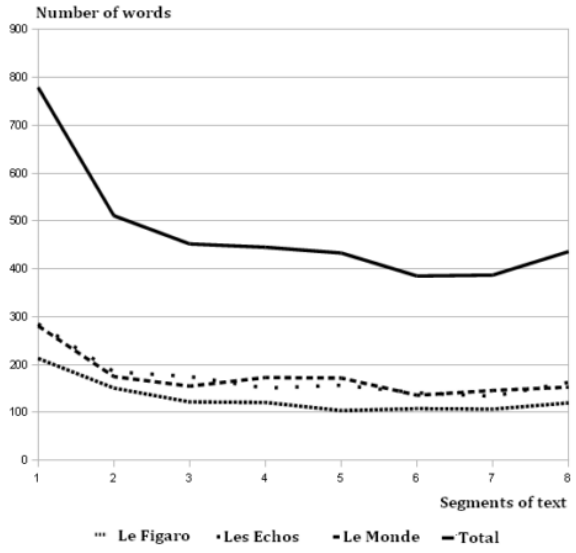

Figure 4.3: Distribution of the number of title words at different segments of documents in each corpus (Le Figaro, Les Echos, Le Monde) and the total of them. Figure taken from Lopez et al. [48]

PageRank on documents

We described how PageRank can be used on texts with the TextRank model to rank nodes in a keyword community in Section 4.5.2. It can also be used on keyphrase extraction from documents [46]. Here we represent each document by a graph where a node is a keyword in the keyword graph; an edge connects two keywords if they co-occur in at least one sentence in this document; and the edge weight

is the co-occurrence of the two nodes in this document. Then we do TextRank on this graph to find keyphrases for a document. Even though PageRank is time consuming, this part can be done off-line after the Noun Phrase Extraction Phase described in Section 4.1. At the labeling stage, given a document cluster, we take all the keyphrases extracted from the documents and rank them by the sum of its PageRank scores in all the documents in that community. The top 20 will be in the label list for that community.

Frequent and Predictive Words method

Here we describe the Frequent and Predictive Words method proposed by Popescul and Ungar [58]. It is used as a baseline to compare our labeling methods with. Different from selecting keyphrases from each document as in our methods, this method takes all the keywords in a document cluster and ranks them by the following frequent and predictive score:

$$Score_{frequent\_and\_predictive}(term) = p(term|cluster) \times \frac{p(term|cluster)}{p(term)} \quad (4.11)$$

In Equation 4.11, the first term $p(term|cluster)$ is the document *frequency* of the term in the document cluster. The second term $\frac{p(term|cluster)}{p(term)}$ is the *predictiveness* which is similar to TFIDF and mutual information because it prefers terms that appear more frequent in a particular cluster than in all the clusters. Terms with high predictiveness values distinguishes a cluster from the other clusters well. The terms with the top frequent and predictive scores are selected as cluster labels. These labels tend to be specific to a cluster and appear enough times in that cluster. This method sifts out general terms that are not descriptive enough and claims to have good results.

4.5.4 Label selection based on the connections between the keyword community and the document cluster

Another way to label a community is to look at the connections between a keyword community and its corresponding document cluster. Recall in Phase IV we

map each document to its most related keyword community/communities according to overall TFIDF score, each keyword in a keyword community ties to different documents by its TFIDF value in that document. For example, the dashed lines in Figure 4.2 indicates the connection between a document and a keyword community. The strength of the connections between a keyword and the documents in its corresponding community can determine the importance of that keyword in representing the community. In this labeling method, we rank each keyword by the sum of its TFIDF score in all the documents in its corresponding document cluster, and then select the top ones as labels. We will refer to this method as the "mapping" labeling method from now on. Comparing to just counting how many documents a keyword appears in, summing TFIDF scores provides a larger difference between different candidate labels (keywords), and is a better approach in labeling.

4.5.5 Combining different labeling results

Preliminary experiments showed that none of the above labeling methods performs perfectly. For example, on some communities method A is better than the others, on other communities method B is the best. Therefore, we try to combine the results of the labeling methods to improve the performance.

One simple combination method is to count "votes". A "vote" of a keyword means that one labeling method has selected that keyword as a label. More formally, the votes of a keyword is

$$votes(keyword) = \sum_{i=1}^{M} \delta(keyword, m_i) \tag{4.12}$$

where M is the total number of labeling methods, m_i is a labeling method, and $\delta(keyword, m_i)$ is a Boolean value indicating if the keyword is a label given by m_i. We then rank the keywords by the number of votes they received. However, this method does not distinguish higher ranked labels and lower ranked labels. Table 4.1 is an example of the top labels of a community about "guitar" given by different labeling methods, for a clearer presentation we only show the top 5 labels. The label "guitar" and the label "bridge" have both been selected 5 times. However,

"guitar" is actually a more suitable label since four labeling methods ranked it as first.

Labeling method	Top 5 labels
Degree Centrality	bridge, neck, series, guitar, pickup
Betweenness Centrality	guitar, part, pickup, bridge, fender
KEA	guitar, fender, pickup, neck, bridge
Automatic titling	guitar, fender, sound, model, bridge
Connection between co-clusters	guitar, bridge, fender, neck, pickup

Table 4.1: Example of different labeling results on the community *guitar*

Therefore, we also take the ranking of a label in a list of labels into consideration. The first label in a label list has a ranking of 1, the second has a ranking of 2, and so on. Since the importance of a label is inversely proportional to its ranking in a label list, we define a new score Combination Score ($Combination_{score}$) for each keyword in a community based on both the votes and the ranking as follows:

$$Combination_{score}(keyword) = \sum_{i=1}^{M} \delta(keyword, m_i) \frac{1}{rank(keyword, m_i)} \quad (4.13)$$

where $rank(keyword, m_i)$ is the ranking of the keyword in labeling method m_i and the other arguments are of the same meaning as in Equation 4.12.

An even more advanced combination method would be to assign different labeling methods different weight. Since different labeling methods have different performance, it is reasonable to give the good labeling methods a larger weight than the others. More formally the combination score with weights would be:

$$Combination_{w\_score}(keyword) = w_i \sum_{i=1}^{M} \delta(keyword, m_i) * \frac{1}{rank(keyword, m_i)} \quad (4.14)$$

where w_i is the weight assigning to labeling method i. The value of w_i need to be determined by a training process that requires pre-defined document clusters and hand-picked labels. We will describe how we pick the labeling methods for combination, and the weights that we assign to them in Section 6.2.

4.5.6 Attempts in utilizing external sources in labeling

Despite the disadvantages of utilizing external sources in cluster labeling discussed in Section 4.5.1, we were still intrigued by this idea because the external sources in fact enriches the candidate labels since sometimes the good labels may not directly occur in the documents, and that the topics are edited by humans. We tried utilizing some sources in labeling but they fail to provide reasonable labels. Moreover, many of these methods are too time-consuming to use on-the-fly, especially the calculation of the semantic similarity, and the anchor tagging of Wikipedia pages.

Semantic Similarity from WordNet

Chen et al. use the terms that have high average WordNet based Lin-similarities with other terms as cluster labels [13]. WordNet is a lexical database built by experts. It contains sets of synonyms called synsets according to word senses. Given two synsets, Lin-similarity calculates the similarity based on the Information Content (IC) of the two input synsets and that of their Least Common Subsumer in the WordNet hierarchy. We have also tried Chen et al.′s method and found that this similarity measure tends to favor general terms rather than the ones that are more suitable labels. For example, given a list of terms, we take their first sense in WordNet as synsets, and calculate the Lin-similarity between each pair of synsets. If a term is a phrase and not in WordNet, we split it up and calculate each word in it separately and use the average as the semantic similarity. The average semantic similarity value for the terms in the above list is listed in Table 4.2 ranked by the values. We also tried other similarity measures such as the path-similarity which calculates the similarity between two senses by the shortest path between them in the WordNet hierarchy. The average path-similarity for some terms about animals is listed in Table 4.3. From these two tables we can see that the more general terms gained larger scores than the topic specific terms which are more suitable labels. Besides, the WordNet similarity is based on different word senses. Here we use the first sense approximation but in real life, there is no way to determine which sense a term is in different contexts without human intervention. Another limitation of using WordNet for semantic similarity is that for terms that are not in the same hi-

erarchy, there is no way to relate them properly. One famous example is the "tennis problem" where the word "player", "racquet", "ball" and "net" are all about the topic tennis but they are not in the same hierarchy in WordNet.

term	average Lin-similarity with the other terms
part	2.57
information	1.76
luxury saloon	1
vehicle	0.75

Table 4.2: Average Lin-similarity for a list of terms about vehicles

term	average path-similarity with the other terms
location	0.86
group	0.78
largest cat	0.75
animal	0.32
specie	0.24

Table 4.3: Average path-similarity for a list terms about animals

Wikipedia Page Titles

Following Scaiella et al., we try to use a Wikipedia annotator TAGME to link the relevant Wikipedia pages with each community and use the most confident ones in this connection as labels [61]. Given a sentence or a set of terms as input, TAGME finds the Wikipedia pages that are associated with the input [25]. Each page is called an anchor, and the confidence of linking this page with the input text is called the *rho score*. Strong connections have a rho score larger than 0.2. It can only handle very small inputs. Since we already have different keyword communities, we use them as inputs to find the Wikipedia pages. For example, given a list of terms about animals, in Table 4.4 we can see the anchors detected by TAGME for them ranked by the rho score of the connections. According to Scaiella et al.′s method of using the rho score to rank the candidate labels, the term "Habitat" would be the top 1 label. However, "Animal" or "cat" are actually more suitable labels

for that community but they received relatively lower rho score. The reason why terms are not descriptive enough have high rho score may be because they are less ambiguous in the Wikipedia senses and TAGME is more confident in linking them with a specific anchor. Besides, TAGME contains errors itself, for example, in Table 4.4, the term "cub" which means a young animal is linked with the country "Cuba".

term	Wikipedia anchor	rho score
habitat	Habitat	0.373
mexico	Mexico	0.3387
leopard	Leopard	0.3329
deer	Deer	0.2453
cat	Cat	0.2226
prey	Predation	0.2077
america	Americas	0.1801
specie	Coin	0.1541
animal	Animal	0.1482
cub	Cuba	0.1009

Table 4.4: TAGME on a list of terms about animals

Wikipedia category information

Sometimes the labels are a set of terms about the same topic and the users need to infer the meanings. For example, the term "ford", "tata", and "land rover" are all car companies but the term "company" is not in the label list. Carmel et al. propose to use Wikipedia category information in cluster labeling since this information is edited by regular users and that Wikipedia has a wider coverage than the hierarchy of WordNet [7]. Following Carmel et al., we want to see if the Wikipedia page categories can help finding the common categories of different terms. The category information is at the bottom of each Wikipedia page. To retrieve this information, we download a full Wikipedia dump[6] since it does not welcome crawlers, then we use an information retrieval tool Lucene[7] to index the page title and category information. It takes about 6 hours on a lab machine with a 7.8G memory and a

[6]http://dumps.wikimedia.org/enwiki/20120403/

[7]http://lucene.apache.org/core/

2.66Ghz*4 CPU. For a given term we search for its corresponding Wikipedia page and take all its categories. For each category we split it and take the Noun Phrases part. In Table 4.5 we present the Wikipedia categories of all the keywords in a community about "jaguar cars" ranked by the number of terms that have them as their categories. We can see that even the top ones only have a score of 3, and not many terms share common categories. Also, the differences between the scores of categories are very small. The top three categories all have the same score but only one is suitable as cluster labels. We can see that the category information is a little noisy may be due to the fact that anyone can add categories to Wikipedia pages. Also, even though Wikipedia is dynamic and covers lots of content, an update of the dump and the category would take so much time and it is not worth it in our cluster labeling task.

Wikipedia category	Number of terms with this category
Motor vehicle manufacturers	3
British Royal Warrant holders	3
Emergency services equipement makers	3
Car manufacturers	3
Tractor manufacturers	2
Dearborn	2
Lawn and garden tractors	2
1903	2
Michigan	2
Motor vehicle battery manufacturers	2
Bus manufacturers	2
Wayne County	2
Electric vehicle manufacturers	2
Truck manufacturers	2
the United States	2
Companies	2
the New York Stock Exchange	2
Ford Motor Comapny	2
Member states	1
Rover	1
Military vehicle manufacturers	2

Table 4.5: Wikipedia page category information on all the keywords in the *jaguar car* community

4.5.7 Post-processing of the labels

The labeling process is not over even though we now have the ranking of each keyword in a community. For each set of labels gained from each labeling method,

we do pos-processing to further refine them. To the best of our knowledge, no other work has a detailed cluster label post-processing discussion. In this section we discuss situations where label refinements are concerned.

Restoring phrases from abbreviations

We want to build a system that assists interactive browsing of a set of documents. If a label is an abbreviation, we should present its original form to the users because otherwise users may be confused. For example, "BED" is selected as a cluster label in a document collection of obesity blog posts. It is not the bed that people sleep in at night; it actually is shortened from "binge eating disorder". We use a simple heuristic to detect the original forms from abbreviations. Before the pruning stage described in Section 4.1.3, we try to find the original form of each word that only contains upper case letters in the document collection. If it proceeds or is followed by a pair of parentheses, then we store the content in the parentheses as a candidate original form for that word and count it as one match between the abbreviation and the candidate original form. To add more confidence, only forms that have been matched more than once are considered. The candidate original form that has the most matches is considered as the original form of that abbreviation. The heuristic may not be able to find all the abbreviations or all the original forms but it does help to a certain extent. At the post-processing stage, we fetch the original form for each abbreviation and present them to the users. For example, the label "BED" would be presented as BED [Binge Eating Disorder].

Dealing with synonymy

Words with the same or similar meanings are synonyms. Sometimes a list of labels for a community contains synonyms. For example, "car" and "vehicle" may all be the label for a community about cars. We do not want to display them both to the users because a synonym does not introduce new information. To discover synonyms, we utilize an external knowledge base WordNet[8] to discover synonyms. WordNet is a lexical database that contains sets of synonyms called

[8]http://wordnet.princeton.edu/

synsets. WordNet synsets are formed by human experts according to a limited, but large vocabulary. We downloaded[9] and stored all the synsets and consult it in this post-processing stage. Even though WordNet does not cover all the words and all the synonyms because of its manual nature, it is good enough for us to use at this post-processing stage for deleting extra synonyms. Given a list of labels, if we spot synonyms, we keep the higher ranked label and take another one off the list. For example, if a community is labeled as "car, mile, vehicle, company, sale, history", we delete "vehicle" from the label list and label that community as "car, mile, company, sale, history".

Note that we do not delete synonyms in phase II when selecting the keywords in the keyword graph. The first reason is that a word can have many senses, it is not reasonable to delete a word when we do not know which sense it refers to in the text. Besides, at phase II we do not know which keyword is more important than another, therefore we would not know which ones to delete even if we spot synonyms.

Lemmas vs original words

Recall in Phase I, we strip each word to its lemma to reduce the inflected forms in the keyword list (see Section 4.1.2). For example, the word "cars" is lemmatized to "car". In the presentation of the labels, however, the lemmas are not as accurate as the original forms of the words. For example, the keyword "Avon products" was lemmatized as "Avon product" while the former is the full name of the company and the latter means products that Avon makes. Therefore we store the original forms of the lemmas and present the most frequent form to the users.

Hypernyms vs Hyponyms

If a term A′s meaning is included in another term B, then A is the hyponym of B and B is the hypernym of A. For example, "cat" is a hyponym of "animal", and "animal" is a hypernym of "cats" WordNet contains an ontology with hypernym hierarchies with hypernym-hyponym relationships from it. If both the hypernym

[9] ftp://ftp.cs.princeton.edu/pub/cs226/wordnet/

and the hyponym are in a label list, one may tempt to delete the hyponym and keep the hypernym but we have observed that this causes more harm than good. For example, a community about cats can be labeled as "cat" or "animal", deleting "cat" simply because it is the hyponym of "animal" does not make a huge difference. However, if we do this on a community about diseases that is labeled as "cancer, diabetes, depression, conditions", the label list would come down to "condition" alone since "condition" is the hypernym of all the other three labels. The hypernym alone is not as representative as the list of labels from which users can infer the topic "disease". In most hypernym-hyponym relationships, the hypernym is more general, sometimes too general to describe the topic of a document cluster.

One might argue that the labels in a list may not be general enough to summarize a document cluster and a hypernym that is general but still specific enough is better. Tseng et al. proposed a hypernym search algorithm based on WordNet to find general categories for a list of labels [69]. In WordNet, all terms are organized in a hierarchy which indicates hypernym-hyponym relationships. The more generic hypernyms are on higher levels. In order to find a hypernym that reveals the general topic of the list of labels but still maintains the specificity of them, they need to choose the common hypernym that is as low-level as possible. For each label in the list, they find all its hypernyms along its path to the root of the hierarchy in WordNet. Then for all the hypernyms of all the labels, they rank them as follows:

$$weight(hypernym) = \frac{f}{m} \times 2 \times (\frac{1}{1 + exp^{-c \times d}} - 0.5) \tag{4.15}$$

where f is the number of labels that is a hyponym of the hypernym in question, and d is the depth of the hypernym in the hierarchy (root has the depth of 0). To normalize the weight to have a value range of 0 to 1, they include nt which is the total number of labels in the list, and c which is set to 0.125.

Their method works well on finding the most suitable common hypernyms for a list of terms. For example, it finds "furniture" as the common hypernym for "chair", "bed", and "desk". However, for cluster labeling on real documents, barely 50% clusters are given reasonable labels in their experiments. After applying their algorithm on one of our data sets we have discovered that the hypernyms found

by this algorithm is still too general to act as cluster labels as shown in Table 4.6. In Table 4.6, each row represents a cluster, the first column is the cluster ID, the second column is a list of labels by one of our cluster labeling algorithms, and the third column is the hypernym found by Tseng et al.′s algorithm.

Cluster ID	Label list (separated by slashes)	Hypernym
C0	kid, food, obesity epidemic, home, adult	person
C1	treatment, diabetes, health professional, excess weight	illness
C2	study, woman, university, BMI (body mass index), issue	person
C3	weight, people, exercise, physical activity, diet	activity
C4	blog, reader, post, article, clinic	artifact
C5	patient, surgery, bariatric surgery, severe obesity, canada	whole

Table 4.6: Example of hypernyms found by Tseng et al.′s algorithm on some clusters in our experiments

We have found that their algorithm highly depends on the input, and thus is not very robust. If one of the input terms is far away from the others in the WordNet concept hierarchy, the algorithm would be forced to find a hypernym that is higher in the hierarchy, which is very general. Also, since it uses WordNet, it is limited by the vocabulary. Tseng et al. have also pointed out that the hypernym structures in WordNet may not reflect the necessary knowledge to analyze the documents [69]. Moreover, a close examination of the label lists in Table 4.6 would reveal that a hypernym may not be a good label. For example, one can infer that cluster $c0$ in Table 4.6 is probably talking about KIDS, the food that they eat, how child obesity is an epidemic, the home environment that they have, and the influences of adults on them. It is hard to include all different aspects of a cluster with just one hypernym.

For the General to Specific property

To fulfill the *General to Specific* property in a taxonomy discussed in Section 3.1.3, we do post-processing according to the relationship between a cluster label l and the labels of its parent and ancestor clusters. We want to keep the labels of lower levels clusters more specific than the higher level ones along their paths. If l falls into any of the following categories, it is deleted from the label list:

1. l is exactly the same or is part of any of its ancestor labels. For example, if the label "weight" is in the label list of a cluster c whose parent is labeled

"weight gain", then "weight" is deleted from the label list of c. The reason is that longer phrases are more specific than shorter ones.

2. l is the combination of its ancestor labels. For example, let's say cluster $c1$ is the child of cluster $root$, and cluster $c1\_1$ is the child of cluster $c1$. Then if "jaguar" is the label of $root$, "car" is the label of $c1$, and "jaguar car" is the label of $c1\_1$, we will delete "jaguar car" from the label list of $c1\_1$.

3. l is the synonym of any of its ancestor labels. For example, if "kid" is the label of a cluster c whose parent cluster has the label "child", then we delete "kid" from the label list of c because it is not introducing new information and it is as general as its ancestor.

4. l is the hypernym of any of its ancestor labels. For example, if "animal" is the label of a cluster c whose parent's parent is labeled "cat", then "animal" is deleted from the label list of c. The reason is that in situations like this instead of going from general to specific it is actually going the other way around which is not desirable.

Longer terms vs parts of them

In the labeling results we have found labels that are a part of other labels. For example, "mac" and "mac OS" are labels for the community about "mac OS". Note that in all steps in our approach, each term is treated as its own. That is to say even though the term "mac" appears in the term "mac OS", we are not counting it twice. At the post-processing stage one may want to favor longer terms because they contain more information. However, we have also observed that a longer term may not be as good a label as a term that is part of it. Besides, they may not even refer to the same thing. For example, "obesity" and "Canadian Obesity Network", "coffee" and "coffee house" are not exactly the same concepts. Therefore we decide not to favor longer terms in post-processing.

Utilizing the author labels

For data sets that already have labels assigned by the authors to the documents, we try to utilize this information as well. Note that this method will not work on all document collections such as search results because they do not have author labels. In the obesity data set from Dr. Sharma$'s$ blog, we have found that 339 out of the total of 490 documents have already been labeled by the author. In Table 4.7, we show the author labels of the document clusters in the obesity data set ranked by the number of documents in the cluster with that label. In Table 4.7, each column starts with the ID of the community and the number of documents in it, then it shows four top ranked author labels with the number of documents in that community that has it as an author label. We can see that the author labels highly overlaps between different communities.

c0 (26 docs)	c1 (123 docs)	c2 (150 docs)	c3 (162 docs)	c4 (21 docs)	c5 (94 docs)
Kids:12	treatment: 23	diet: 20	diet: 24	policy: 3	bariatric surgery: 30
policy: 7	policy: 20	Kids: 16	policy: 23	bariatric surgery: 2	policy: 18
ingestive behavior: 6	prevention: 15	cardiovascular disease: 15	weight management: 20	weight management: 1	treatment: 7
diet: 5	weight management: 9	diagnostics: 15	ingestive behavior: 19	diet: 1	epidemic: 6

Table 4.7: Top author labels in different communities on the obesity data set

One might use the unique author labels of each community as the cluster labels. Table 4.8 shows the unique labels for each community. We can see that many of these labels have only been assigned to one document in a cluster by the author. They are not general enough to describe all the documents in that cluster. Since the author labels do not distinguish one cluster and another, we only use author labels in the post-processing stage. We come up with a heuristic that boosts the author labels. Given a cluster labeling results of our system for a cluster, if there is a match in the label list with the author labels for that cluster, we boost the matches as the top labels. If two or more clusters have the same match, we give the author label to the cluster where it has a higher proportion (number of documents that has it as an author label divided by the total number of documents in the cluster).

c0	c1	c2	c3	c4	c5
junk food: 1	barriers:1	sarcopenia: 1	thrifty gene: 2	NA	Association: 1
	urinary incontience:1	injury: 1	meal replacement: 1		discrimination: 1
	virus: 1	women: 1	joints: 1		

Table 4.8: Author labels unique to each community in the obesity data set

Part III

Experiments and Discussions

Chapter 5

Experiment Setup

5.1 Data collection and pre-processing

To evaluate the document clustering and labeling performance, we need the ground truth of clustered documents and the labels. There is only a handful of publicly available data set judging query sense and subtopic discovery with ground truths, and none of them is suitable for our task to our knowledge [8]. TREC (Text Retrieval Conference) Web track[1] developed a data set of topics and subtopics but it is small and mainly focuses on faceted subtopic types instead of ambiguous subtopic types. For example, for query "what tropical storms have caused property damage and/or loss of life", it provides two subtopics "hurricane" and "typhoon". Another data set is developed at Image CLEF[2] (Cross-language image retrieval evaluations) which is mainly about the diverse geographical distribution of photos on the same topic [8]. Carpineto et al. collected two data sets: AMBIENT[3] and ODP-239[4] in 2008. AMBIENT contains ambiguous Wikipedia entries (Wikipedia pages that has *(disambiguation)* in the title), each with different senses and 100 manually annotated snippets of web pages for each sense. ODP-239 is taken from the Open Directory Project [8]. Many search result clustering researches used these data sets for evaluation [61][9][38]. However, these works are all snippet-based flat clustering methods. As discussed in Section 3.4, our method is capable to work on

[1] http://trec.nist.gov/data/webmain.html
[2] http://www.imageclef.org/
[3] http://credo.fub.it/ambient/
[4] http://credo.fub.it/odp239/

the full content of the document and we do not have to use snippets to reduce the running time. Besides, our method needs to build a keyword network based on co-occurrences and short snippets are unable to provide enough nodes and edges to construct suitable networks for community mining. AMBIENT and ODP-239 only provide a snippet for each webpage and many of the URLs are dead links at the time of writing. Besides, they only have one layer of subtopics for each topic and are only good for evaluating flat clustering. Therefore, we need to collect our own data set for ground truth.

5.1.1 Data Collection

Data collection of the web search results

We created a data set for our task using the Google search engine. For each query, we intentionally searched for some of its senses and gathered the top results returned by google.com for each query sense. For example, for the query *jaguar*, we searched for *jaguar car*, *jaguar animal*, *jaguar mac OS*, and *jaguar guitar* and gathered web pages from each query sense. We merged these pages together as the document collection under the query *jaguar*. We treat the results returned by Google as the ground truth. For example, if a webpage is returned by Google when we searched for *jaguar animal*, then the ground truth of this page is that it is under the topic of *jaguar the animal*. We wrote a crawler that uses a Google API to crawl the URL and the HTML content of the web pages from google.com with a 20 second lapse between HTTP requests to avoid our IP being blocked by Google.

Only web pages that satisfy the following constrains are collected:

1. The page welcomes crawlers. We cannot gather any information on web pages that do not welcome crawlers.

2. The page contains a sufficient amount of text: This work only focuses on the clustering and labeling of textual documents. If a webpage contains too few words, it is more likely that the text in it is not relevant to our query. It could be a short welcome message to visitors; it could be a home page with just banners and pictures, or it could be contact information. Therefore, we

set thresholds to the number of words and sentences a web page contains. We chose 100 as the threshold for the number of words. This is a very low bound since the average number of words per web page is estimated to be 474 [44]. We set 5 as the threshold of the number of sentence. It is also a loose threshold since the average length of English sentences has been estimated as 15-20 [55].

3. Readability works on the page. To get the main text from a webpage we need to do parsing. An effective parsing tool that we use is called Readability. We will elaborate on it in Section 5.1.2. It has some limitations in that it may not work on web pages that do not follow certain HTML standards. For simplicity we only collect pages that Readablity can parse.

For the experiments we collected 5 document sets based on 5 ambiguous queries: "jaguar", "penguin", "AVP", "tiger", and "Michael Jordan". These queries are selected because the concepts that they cover are common enough for users to evaluate. For each query, we selected some of its senses based on the subtopics of the query given by Wikipedia and TREC, and also the "hits" of each query sense from Google. For example, we are aware that the query "Michael Jordan" is the name of a Berkeley professor but we are not sure if we should use the sense "professor" or "researcher". We did a search on Google and found that "michael jordan berkeley professor" returns 431,000 results whereas "michael jordan berkeley researcher" returns 1,730,000 results. Therefore, we search for "researcher" which has more hits and collected documents under that query sense. In order to be able to determine the quality of clusters that are subtopics of the same topic, we also searched subtopics of some of the query senses in Google to collect documents. For example, for query sense "jaguar car" under query "jaguar", we selected two subtopics: "jaguar car history" and "jaguar car dealer". We feed them to Google search engine and merged their results as the documents of the query sense "jaguar car". For each query sense, or subtopic, we collect 30 documents and gather them together as the document collection under a query. A complete list of the queries, their query senses and the subtopics of the query senses (shown in parentheses) along with the

total number of documents for each query is shown in Table 5.1.

Query	Query Senses and subtopics	number of documents
jaguar	animal (animal facts, animal rescue), car (car history, car dealer), Mac OS, guitar	180
penguin	Pittsburgh hockey team, publisher, kids club, algorithm	150
AVP	Volleyball, antivirus software, Avon, movie, airport	150
tiger	Aircraft, Woods, animal, hash	120
michael jordan	basketball player (career, quotes), Berkeley researcher	90

Table 5.1: List of queries, query senses, subtopics of query senses with the number of documents

We are aware that there are risks in collecting documents this way. First, the selected query senses does not cover all the senses of a query and the number of queries we selected is far fewer than the number of existing ambiguous queries on the web. However, they are sufficient to serve the purpose of determining if our method can detect different senses in a document collection. Besides, it is also risky in relying on Google to find ground truth for each document. The reason is that Google is not 100% precise in finding the relevant web pages for a query. However, this method saves lots of human labor in annotating each webpage, and Google is among the best search engines currently. Moreover, one could also argue that because we feed the query senses to Google, they have a higher probability in appearing in the retrieved documents than regular document collections and the document collection is actually biased for labeling. However, rather than being biased it could hurt the labeling method another way around. For example, one of the query senses for "penguin" is "Pittsburgh hockey team", these words may appear more often in the documents collected, but a user may as well want to label the documents as "sports".

Data Collection of Dr. Sharma′s Obesity Notes

One of the motivations of this work is to see if the taxonomy generated by our methods can provide a better navigation tool than Tag Clouds for the readers. Thanks to Dr. Arya Sharma, we are able to access dumps of his blog Dr. Sharma′s Obesity

Notes[5] in XML format. This work experiments on the 2012-05-11 dump. It has 3953 posts. Only blog posts that satisfies the following constrains are collected:

1. The post type is “post” but not “revision”, “draft”, “attachment” or any other types. We only want to deal with the blog posts that are actually articles posted on the blog.

2. The URL of the post does not start with ”http://www.drsharma.ca/?page_id=”. We have found that these pages are no longer available on the web. The users would see “Not Found” if they are directed to these pages. Including them would not help with the browsing process.

3. The same as the search results, each blog post should have a sufficient amount of text. Since we know that all the blog posts are about “obesity” and not random web pages, we lower the threshold of the number of words and sentences to be 15 and 3, respectively.

We get 490 documents that fulfill these requirements and we treat them as a document collection. Another way of looking at this is to treat them as search results under the query “obesity” since the blog is about obesity.

5.1.2 Data pre-processing

After gathering the HTML and XML content of the documents, we need to parse them to get the main, and plain text of the web pages. Many web pages contain noises such as banners and advertisements that are not related to the topic of that page. To identify and purify the main text is especially important in the cluster labeling method based on automatic titling described in Section 4.5.3 since it plays a big part in identifying the first sentences in a document.

Identifying the main text

On XMLs this is an easy task since the main content of a blog post is surrounded by the content markup. On HTML formatted web pages, we use the Readability

[5] http://www.drsharma.ca/

Project to identify the main content of a web page and delete advertisements, banners and other noises. Readability is a project developed by Readability, LLC, It turns a web page into a cleaner view for readers. It is incorporated in Apple′s Safari browser; it also provides add-ons for Chrome, and also apps in iOS, Mac and in Android. For example, Figure 5.1 presents a web page www.decarie.com/en/new/jaguar as seen on a browser. It has banners, buttons and navigational frames at the top. The main text that we want is under the picture of the car. Figure 5.2 presents the same web page after applying Readability. We can see that only the main text has been selected. In our experiments, we use a python port[6] of the Readability project to process the HTML web pages and identify the main texts. At this stage we can also parse out the title of each web page if it has one. Even though titles are summaries of the content of a document, we do not give them extra weight. We treat a title the same as a sentence in a document because not all documents have titles.

We have also tried to write our own heuristics to identify the main text. For example, we delete advertisement components whose id is "ad", we also only take texts wrapped by a certain list of HTML tags such as "td, div, span, p" etc. These heuristics do not work as well as using the Readability project therefore we use the Readability project to identify the main text from web pages.

Purifying the main text

We also purify the main text to plain texts. First, we remove all the HTML markups, images, urls, etc. from the Readability results. We use a module *clean_HTML* by Natural Language Toolkit (NLTK) to do this. Now we have the main text without markups of the web pages. Next, we delete all the empty lines and multiple spaces. We also un-escape the HTMLs (eg. we turn "&amp" in the HTML into "&" in our text).

[6]https://github.com/buriy/python-readability

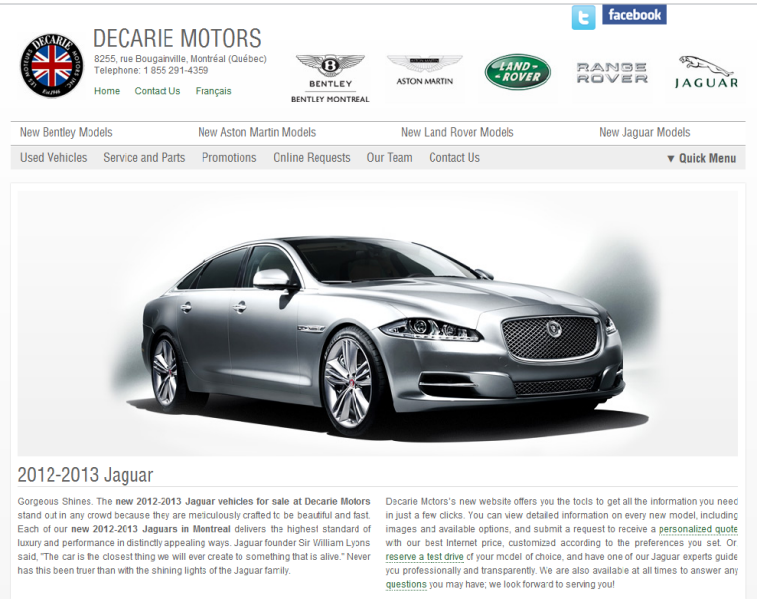

Figure 5.1: The web page www.decarie.com/en/new/jaguar as seen on a browser

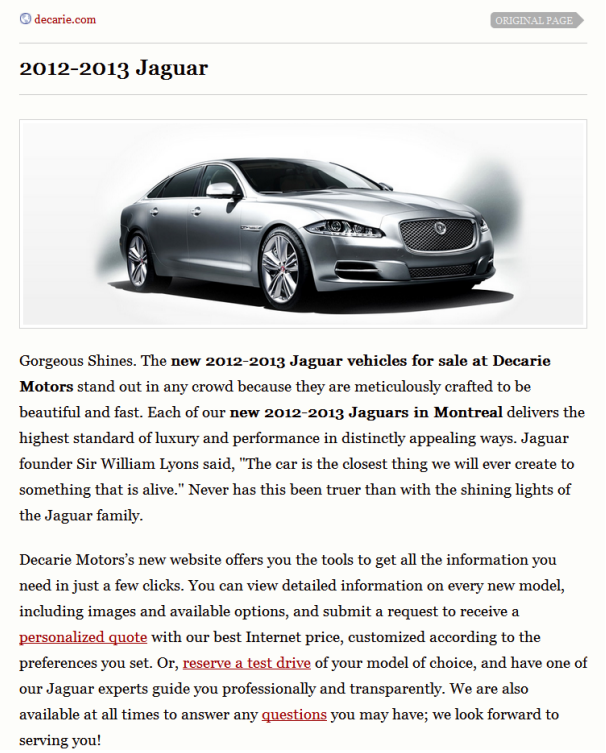

Figure 5.2: The web page www.decarie.com/en/new/jaguar after processed by Readability

5.2 Evaluation Metrics

The evaluation of automatically generated taxonomies is a non-trivial task [42]. Ideally the taxonomies should satisfy every desired property described in Section 3.1. We evaluate the document clustering and cluster labeling results separately.

5.2.1 Evaluation of the document clusters

We measure the document partition quality by comparing our document partitions with the pre-defined ground truth gathered in Section 5.1.1. There are many popular measures for this purpose. Here we adapt the following metrics: Adjusted Rand Index (ARI) and Cluster Contamination (CC).

Adjusted Rand Index (ARI)

ARI measures how close the document clusters generated by our system (P) matches the real clusters in the ground truth (R) [76]. The value ranges from -1 (no agreement between P and R) to 1 (P and R are identical), and 0 means that P is a randomly generated partition. It takes into account the True Positives (a), the False Positives (b), the False Negatives (c), and the True Negatives (d). The values of a, b, c, and d are calculated based on the number of document pairs that are of the same cluster in P and in R. These values are further explained in the confusion matrix in Table 5.2, where 1 means that a document pair is in the same document cluster, and 0 otherwise.

		R	
		1	0
P	1	a (TP) number of document pairs that are in the same cluster in both P and R	b (FP) Number of documents that are in the same cluster in P, but not in R
	0	c (FN) Number of document pairs that are in the same cluster in R, but not in P	d (TN) Number of document pairs that are not in the same cluster in P, or R

Table 5.2: Confusion matrix of the value a, b, c, d given a Real partition (R), and the system′s partition (P)

The ARI value is calculated as follows:

$$ARI(R, P) = \frac{2(a * d - b * c)}{(a + b) * (b + d) + (a + c) * (c + d)} \tag{5.1}$$

It penalizes False Positives and False Negatives. The more similar P and R, the larger the ARI value, and the better the clustering is.

Cluster Contamination measure

The traditional clustering evaluation measures are convenient and reflect the clustering qualities of certain aspects. However, they do not explain why the clustering is not perfect [21]. Besides, they look at the partition of all the clusters as a whole whereas our evaluation is interested in the quality of each individual clusters, especially the ones on the top levels that are presented to the users [21]. None of the traditional measures for clustering fully fulfills our evaluation needs. For example, Purity does not distinguish if a cluster of documents is of the same topic or is a mixture of different topics; F-measure prefers large clusters and penalizes separated clusters that come from a single original topic [21].

Therefore, we also adapt another cluster validation measure specially designed for our task developed by Weiss: Cluster Contamination [21]. In the Cluster Contamination measure, if the documents in a cluster partition all belong to the same topic in the ground truth, then the cluster is considered "pure". The Cluster Contamination for a cluster k is defined as follows:

$$contamination(k) = \frac{a_{10}(k)}{a_{max}(k)} \tag{5.2}$$

where $a_{10}(k)$ is the number of "bad pairs" in k. A pair of documents that are in k together but are not together in any of the clusters in the ground truth (False Positives as described in Section 5.2.1) is called a bad pair. $a_{max}(k)$ is the maximum number of possible bad pairs in k. More formally, let $C = c_1, c_2, ..., c_m$ be the document partition in the ground truth, and $K = k_1, k_2, ..., k_n$ be the document partition of the system. $H = h(c, k)$ is a two-dimensional matrix, where $h(c, k)$ is the number of documents in c that is assigned to k. $a_{10}(k)$, and $a_{max}(k)$ are calculated as follows:

$$a_{10}(k) = \sum_{c} \sum_{c'<c} h(c,k) \times h(c',k) \tag{5.3}$$

$$a_{max}(k) = \sum_{c} \sum_{c'<c} \hat{h}(c,k) \times \hat{h}(c',k) \tag{5.4}$$

where for $i = 0...|C| - 1$:

$$\hat{h}(c_i,k) = \begin{cases} \lfloor \frac{m}{|C|} \rfloor + 1 \;\; if \;\; i < m \; mod \; |C| \\ \lfloor \frac{m}{|C|} \rfloor \;\; otherwise \end{cases}$$

$$m = \sum_{c} h(c,k)$$

Cluster Contamination ranges from 0 to 1. 0 means that a cluster is pure. 1 means that a cluster is "fully contaminated", and the documents in it are an even mix of documents from different clusters in the ground truth. Cluster Contamination measures the purity of topics in a document cluster. A good cluster should have a small Cluster Contamination.

Compactness

As discussed in Section 3.1.1, taxonomies should be as compact as possible. The number of clusters generated should be as close to the ground truth as possible. However, ARI punishes the extra number of clusters but not directly, and Cluster Contamination measure overlooks the compactness of the clustering. For compactness we simply count the number of clusters generated by our methods and compare it with other systems.

5.2.2 Evaluation of cluster labeling

The evaluation of cluster labeling is very subjective. In this section we describe the user survey we designed for determining the cluster label ground truth. We define what a *correct label/match* is, and introduce some of the most popular cluster labeling evaluation metrics in the literature that we use for cluster labeling evaluation, namely match@N, P@N, MRR@N and MTRR@N [42][68]. N is the number of

labels presented to the users. The higher the metric scores and the smaller the N, the better the labels are from the user′s perceptive.

User Survey for cluster label ground truth

To determine the ground truth of document cluster labels we conducted a user survey with results from 11 Computing Science graduate students. We present the document clusters and all the labels selected by all of our labeling methods. The users examined the documents and then selected one or more labels that best describe the document cluster, or enter other labels manually if they think none of the phrases provided are good enough. The reason is that we do not only want to see the statistics of the labels we extracted, but also we evaluate if the labels extracted by our system are good enough, or if others are better than them.

The interface of the user survey is shown in Figure 5.3. The label lists provided are ranked alphabetically to avoid bias. For each level in the taxonomy of a query, we present up to 5 randomly selected clusters for users to examine. The main reason is to reduce the work load of the users. Besides, this cluster selection actually mimics the user behavior since in real life a user may not click on every single one of the clusters. For each label in each cluster, whether it is in the label list or entered by users, we count how many users have selected it as the label for that cluster. The label with the most votes of each cluster is treated as the ground truth.

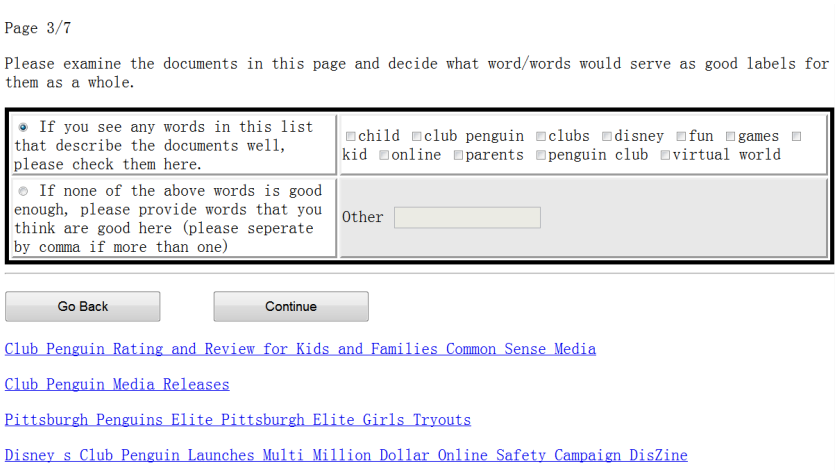

Figure 5.3: Cluster labeling user survey interface

Definition of a correct label

Following Treeratpituk, given a cluster with ground truth label S and its parent label P, we define a system label L as a correct label if: L is identical with "S", "$S\ P$", or "$P\ S$" [68]. This measure is more strict than Treeratpituk because we only measure exact matches.

Match at top N labels (match@N)

Match@N is the number of clusters which has at least one correct label in the top N results in the label list, divided by the number of clusters [68].

Precision at top N labels (P@N)

P@N is the number of correct labels in the top N labels divided by N, and then averaged over all clusters. It measures the percentage of correct labels displayed [42][68].

Mean Reciprocal Rank at top N labels (MRR@N)

Given a list of N labels of a cluster, the Reciprocal Rank (RR) is the inverse of the rank of the first correct label in the list. For example, if the first correct label in the list is the 3rd, then RR is $1/3$. If none of the labels are correct, then RR is 0. MRR@N is the average of RR on all clusters [42][68].

Mean Total Reciprocal Rank at top N labels (MTRR@N)

A cluster may have multiple topics or multiple correct labels. Therefore, instead of just taking the position of the first correct label into account, as in MRR@N, MTRR@N considers all the correct labels in the label list. Given a list of N labels for a cluster, the Total Reciprocal Rank (TRR) is the total of the RRs in the list. For example, if the 2nd and 3rd labels are both correct, then TRR is $1/2 + 1/3 = 5/6$. MTRR@N is the average of TRR on all clusters [68].

5.3 Parameter Settings

5.3.1 Threshold of t_{df}

The threshold t_{df} selects keywords according to their Document Frequency (DF) as nodes in the keyword graph in phase II. A low t_{df} introduces noise into the graph. However, if t_{df} is set too high, some small communities may be neglected. Chen et al. have found that $0.03 \leq t_{df} \leq 0.05$ renders the best clustering performance in terms of ARI [13]. We have tested both ARI and Cluster Contamination in this range on all 5 queries in our data sets. The ARI scores shown in Figure 5.4, and the Cluster Contamination scores shown in Figure 5.5 for each query are the average of all the clusters on the top levels under t_{df} of 0.03, 0.04 and 0.05. We can see that $t_{df} = 0.04$ renders the best results on average, and on most queries. Besides, the threshold t_{df} has an impact on the number of discovered communities. As shown in Table 5.3, $t_{df} = 0.04$ discovers the same number of communities as the ground truth on all queries on the top level, whereas the other thresholds discover the wrong number of clusters on some queries. Therefore, in all of our experiments we set $t_{df} = 0.04$.

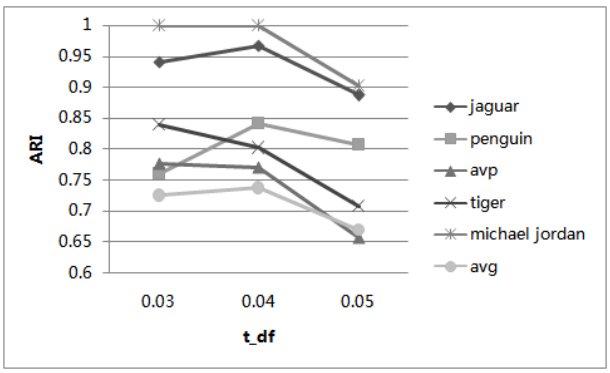

Figure 5.4: The impact of threshhold t_{df} on ARI

5.3.2 Threshhold of t_{merge}

The threshold t_{merge} measures the need to merge communities in the community refinement stage discussed in Section 4.3.4. Chen et al. found that a good range of t_{merge} is between 0.15 and 0.25, and that the clustering is not very sensitive to t_{merge}

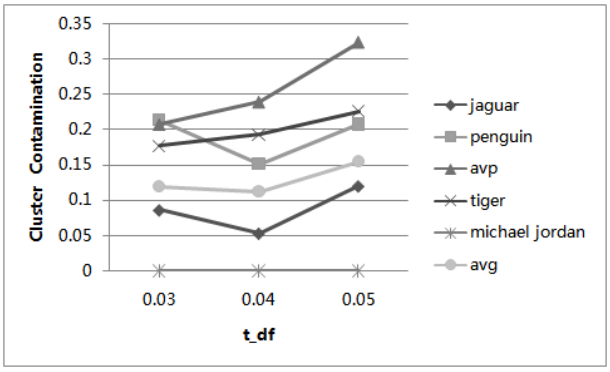

Figure 5.5: The impact of threshhold t_{df} on Cluster Contamination

	t_{df}=0.03	t_{df}=0.04	t_{df}=0.05	Ground truth
Jaguar	4	4	4	4
Penguin	5	4	4	4
Avp	5	5	4	5
Tiger	4	4	5	4
Michael Jordan	2	2	3	2

Table 5.3: The impact of threshold t_{df} on the number of clusters discovered. Each column is the number of clusters discovered under a certain t_{df}, or under the ground truth.

[13]. We have also found that $t_{merge} = 0.2$ renders the right number of clusters on the top levels which are the query senses. We use $t_{merge} = 0.2$ on the top levels in our experiments.

On the lower levels, the clusters are the sub-clusters of the top levels clusters. The documents have lower overall-TFIDFs on these clusters because they contain fewer nodes than their parents. We go beyond Chen et al. and set different t_{merge} values for clusters on the lower levels. t_{merge} divides by 2 as the taxonomy goes down for one level.

5.3.3 Threshold t_Q

According to the Compactness property in Section 3.1.1, a good hierarchy should not be too deep for browsing purposes. Therefore, we need to measure the need for recursive community mining. Chen et al. pointed out that Q modularity scores can help determine if the hierarchy stops at the right time [13]. A larger Q-modularity score means that the communities discovered are well separated; a smaller Q means

that the keyword community is strongly correlated internally. The more query senses, or subtopics there are, the larger the Q score is. If there is only one query sense or subtopic, then Q score is close to 0. In social network analysis, a Q score of 0.3 to 0.7 indicates strong community structure [54]. Chen et al. found that $Q = 0.1$ indicates good separation in query senses for the task of search result clustering. However, our experiments agree with social network analysis in this matter and find that $t_Q = 0.3$ is a good indicator of whether to further split or not in the recursive community mining discussed in Section 4.3.3.

In Table 5.4, We show the Q-score of some of the queries, query senses, subtopics, and sub-sub topics on the queries "jaguar" and "Michael Jordan" which have multiple levels. The Q-score is shown below the description of each community in the table. We can see that the query "Michael Jordan" with 2 query senses has a Q score of 0.4387 (larger than 0.3). The query sense "Michael Jordan Berkeley Researcher" does not have multiple subtopics, and it has a Q score of 0.1301. Query sense "Michael Jordan Basketball Player" has multiple subtopics, with a Q score of 0.3721. "Quotes" is one of the subtopics under the query sense "Michael Jordan basketball player". It has a Q score of 0.5413 and has multiple senses. It splits into to sub-subtopics such as "failures" and "life". They both contain only one sense and both have a Q score that is less than 0.3. We can see similar trends in the query "jaguar" as well. Query "jaguar" has multiple senses and a Q score of 0.3992. Query sense "Jaguar mac OS" does not have subtopics and has a Q score of 0.1864 (less than 0.3). The query sense "animal" (Q score 0.4753) has a very strong community structure with multiple subtopics, and one of its subtopics "jaguar animal rescue center" (Q score 0.250) does not have sub-subtopics and the Q score is less than 0.3. To summarize, we set $t_Q = 0.3$, and only do recursive community mining on a community if its max Q modularity score is larger than t_Q.

query	query sense	subtopic of a query sense	sub-subtopic
Michael Jordan 0.4387	Berkeley Researcher 0.1301	NA	NA
	Basketball Player 0.3721	quotes 0.5413	failures 0.1472
			life 0.052
			...
		...	...
Jaguar 0.3992	Mac OS 0.1864	NA	NA
	animals 0.4753	rescue center 0.250	NA
		...	...
	...	...	...

Table 5.4: Q modularity scores for different queries, query senses, subtopics of query senses, and sub-subtopics

Chapter 6

Experimental Results and Discussion

In this chapter we report the performance of our document clustering and labeling methods. We show the results of the clustering and the cluster labeling separately on the web search results that we collected in Section 5.1.1. The obesity data set it is too domain specific to get ground truth from regular users. We discuss the experiments on the obesity data set at the end of this chapter.

6.1 Document Clustering performance

In this section we show the clustering performance on the metrics listed in Section 5.2.1.

6.1.1 Document Partition Quality

For the document partition quality, we use an effective variation of K-Means[1], which is a common document clustering algorithm, as the baseline to compare our methods with [34][35]. For the parameters in K-Means, we use the keywords extracted by our method as the features, and the TFIDF scores of the keywords as the feature values. We feed the number of communities, found by our method as the parameter k to K-Means. We look at the top level, and the lower levels clusters separately. On the top levels, we are evaluating how well our method discovers different senses of one query, in other words, the disambiguation ability of our method. On lower levels, we are evaluating how well our method discovers differ-

[1] http://www.cs.umd.edu/ mount/Projects/KMeans/

ent aspects (subtopics) of the same topic. While all the levels are important in the browsing process, the top levels carry more responsibility because they are the ones that users see first.

Document Partition Quality on the top levels

The ARI and the Average Cluster Contamination scores over all 5 queries on the top levels of our method and K-Means are shown in Table 6.1 and Table 6.2, respectively. The Average Cluster Contamination score is the mean of the Cluster Contamination score of all the top levels clusters under a query. We can see our method gets higher (and better) ARI score on all five queries than K-Means. On Average Cluster Contamination, our method gets lower scores (less contamination) on all queries except for the query "AVP". After a close examination of each cluster of the top levels under the query "AVP" we found that K-Means renders four pure clusters but the other cluster has a Cluster Contamination score as high as 0.89 (shown in Table 6.3). We also found that K-Means tends to group many documents into one big cluster and leaves the other ones in very small clusters. In the ground truth, there are 30 documents in each of the query sense under the query "AVP". The number of documents in each cluster of our method is "24, 33, 34, 28, and 35" with some multi-mappings. We can see that some documents have been assigned to the wrong cluster, but the number of documents in each cluster is close to that in the ground truth. In the document partition of K-Means, however, the number of documents in each cluster is "1, 3, 18, 30, and 98". This explains why K=Means generates small four pure clusters and another large and highly contaminated one. Our method, on the other hand, does not generate pure clusters, but it does not generate highly polluted ones either, which is more desirable in extracting different query senses for browsing purposes. Overall, our method performs better than K-Means in the partition of document clustering.

Document Partition Quality on the lower levels

On the web search results data set, we collected some subtopics on the lower levels for cettain query senses. For example, the query sense "jaguar animal" is at the top

query	our method	K-Means
jaguar	0.9679	0.5213
penguin	0.8421	0.3187
tiger	0.8024	0.6153
AVP	0.7712	0.3250
michael jordan	1	0.0221

Table 6.1: ARI score of our method and K-Means on the top levels of different queries

query	our method	K-Means
jaguar	0.0533	0.1689
penguin	0.1517	0.352
AVP	0.239	0.1781
tiger	0.1927	0.2611
michael jordan	0	0.4394

Table 6.2: Average Cluster Contamination score of our method and K-Means on the top levels of different queries

cluster ID	our method	K-Means
c0	0.331	0
c1	0.3224	0
c2	0.1957	0.8904
c3	0.0863	0
c4	0.2597	0

Table 6.3: Cluster Contamination of top levels clusters under the query “AVP” given my our method, and by K-Means.

levels for the query "jaguar", it has two subtopics "animal facts" and "animal rescue centre". They are different aspects of the same sense "jaguar animal". The three sets of subtopics are listed in Table 5.1 with parentheses, namely "facts" and "rescue" for the query sense "animal" under the query "jaguar"; "history" and "dealer" for the query sense "car" under the query "jaguar" and "career" and "quotes" for the sense "baseketball player" under the query "Michael Jordan".

The ARI score and the Average Cluster Contamination on these subtopics are shown in Table 6.4 and Table 6.5. From these two tables we can see that on two out of three query senses, our method generates clusters with higher ARI scores than K-Means, and our method generates more contaminated clusters than K-Means on all three query senses. Our method does not outperform K-Means on subtopics on the lower levels. Note that both our method, and K-Means have low ARI scores (less than 0.5), and high Cluster Contaminations (larger than 0.5). This means that neither our method nor K-Means have good performance in cluster partitioning on the lower levels. This indicates that the subtopics which are different aspects of the same topic are hard to separate. Our method uses the correlations between terms to detect different clusters. The results show that this does not work well on the lower levels where there is no clear separation between the vocabularies used by different subtopics.

query sense	subtopics	our method	K-Means
jaguar/animal	facts	0.4277	0.3211
	rescue		
jaguar/car	history	0.1216	-0.0003
	dealer		
Michael Jordan/basketball player	career	0.107	0.328
	quotes		

Table 6.4: ARI score of our method and K-Means on the subtopics on the lower levels

6.1.2 Compactness

For compactness, we compare our results with some commercial systems. On the top levels under $t_{df} = 0.04$, our method generates exactly the same number of

query sense	subtopics	our method	K-Means
jaguar/animal	facts	0.2884	0.2613
	rescue		
jaguar/car	history	0.5122	0.3998
	dealer		
Michael Jordan/basketball player	career	0.6576	0.5501
	quotes		

Table 6.5: Average Cluster Contamination score of our method and K-Means on the subtopics on the lower levels

clusters as the ground truth on all queries (shown in Table 5.3). As discussed in Section 4.3.2, our method automatically detects the number of clusters. This is one of the advantages of our method. Refer back to the clustering results on the query "jaguar" on Yippy.com and CARROTSEARCH shown in Figure 2.7 and Figure 2.8 in Section 2.3. They both display the top 10 clusters on the top level. In Yippy.com, the clusters labeled "International", "Parts", "Sedan, Sales", "Luxury car", and "Sports cars" are all about the query sense "jaguar cars". "Panthera, Feline" and "Facts" are both about the sense "animals". In CARROTSEARCH, the clusters "Jaguar Cars", "New Jaguar", "Jaguar UK", "Land Rover" and "Jaguar Models" are all about the query sense "Jaguar Cars". They fail to combine clusters of the same query sense together, and the clustering is not as compact as it should be. Even though the document collections in these systems are not the same as our data sets, we can still see that our method groups documents of the same topic together and generates a rather compact taxonomy.

6.2 Cluster Labeling performance

In this section we present the labeling performances of different labeling methods on all four metrics discussed in Section 5.2.2: match@N, P@N, MRR@N, and MTRR@N, with N ranging from 1 to 5. From here on, we refer to the "degree", the "betweenness", and the "graphpagerank" methods as the labeling method that selects labels based on degree centrality, betweenness centrality, and PageRank in the keyword community (see Section 4.5.2), respectively. We refer the "DF", "TF", "TFIDF", "KEA", "titling", "docpagerank", and "docfandp" (Frequent and Pre-

dictvie words) method as the labeling methods based on the document cluster described in Section 4.5.3. The "mapping" method is the labeling method based on the connection between the keyword community and the document cluster described in Section 4.5.4. For ease of presentation, we present the labeling quality on the top levels and on the lower levels separately.

6.2.1 Cluster Labeling performance on the top levels

Here we show each labeling method′s performance on the top level. The scores of the four metrics are presented in Figure 6.1, 6.2, 6.3, and 6.4, respectively. The scores are calculated by the average of each metric over all five queries in our data sets.

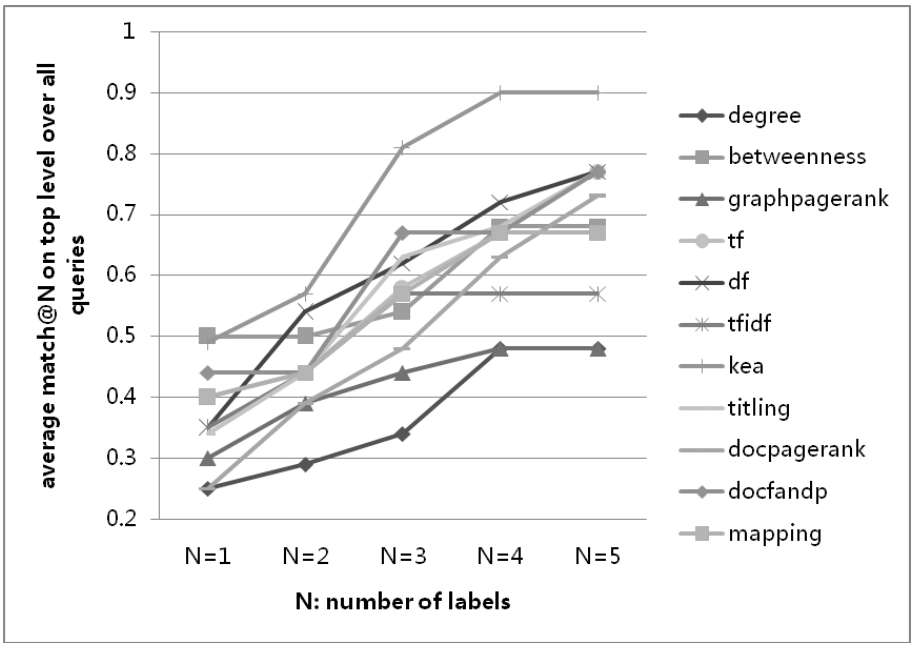

Figure 6.1: Average match@N on the top levels over all queries of different labeling methods

From Figure 6.1, 6.2, 6.3, and 6.4 we can see that the KEA method achieves the highest average score over all queries on all the metrics. The baseline degree method is always the worst. The reason is that the degree central terms may be just common rather than important. They link to many different terms and thus have high degrees, but they are not descriptive enough to be cluster labels. How the other methods perform is hard to see from these figures since the lines are intertwined,

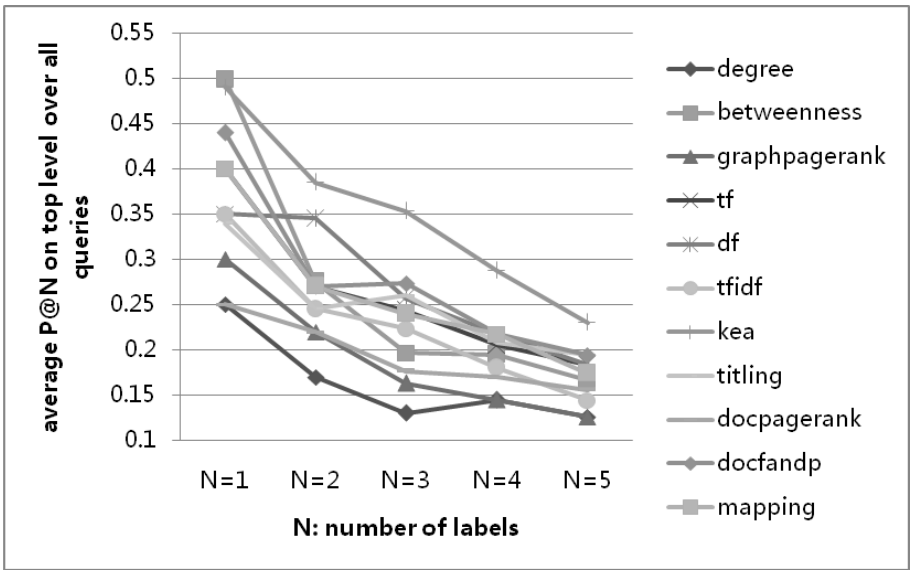

Figure 6.2: Average P@N on the top levels over all queries of different labeling methods

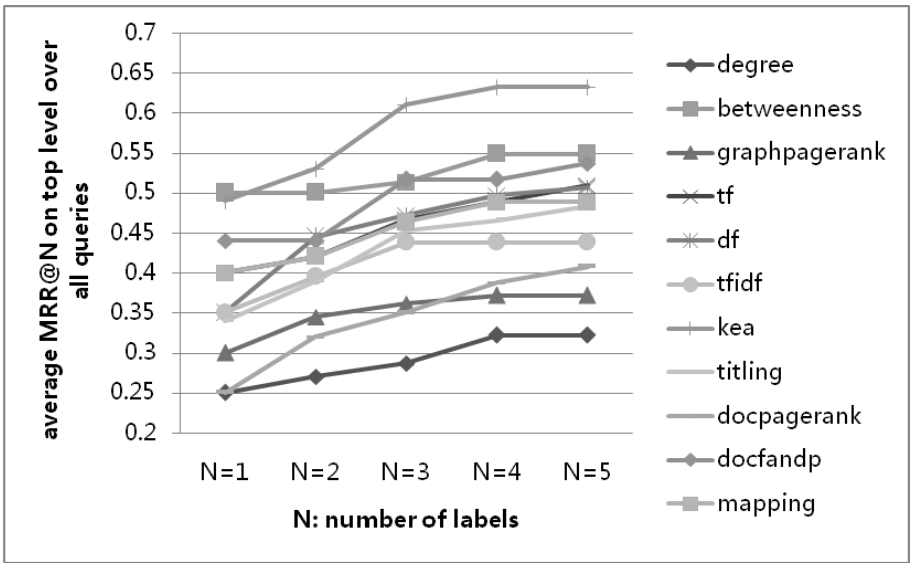

Figure 6.3: Average MRR@N on the top levels over all queries of different labeling methods

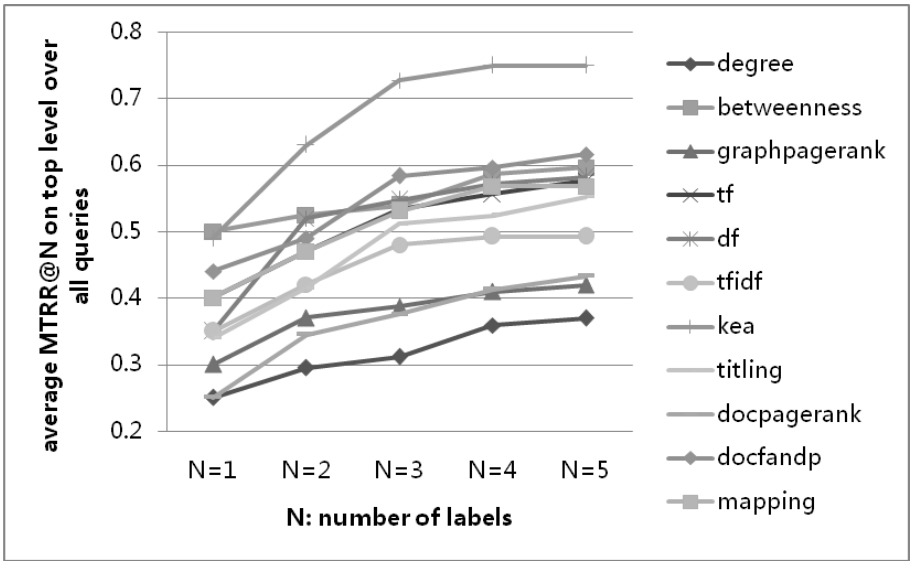

Figure 6.4: Average MTRR@N on the top levels over all queries of different labeling methods

therefore we compare them with the five baselines: degree, TF, DF, TFIDF, and docfandp. On each metric we calculate the number of baselines that a labeling method outperforms. The results are presented in Table 6.6, 6.7, 6.8, and 6.9 on the four metrics, respectively. We can see that on all metrics all of our methods outperform at least one baseline: the degree method. The KEA method outperforms all five baselines on all Ns and all metrics. The betweenness method outperforms all the baselines when N=1 on all metrics. The titling and the mapping method both outperform a decent number of baselines as well.

	betweenness	graphpagerank	KEA	titling	docpagerank	mapping
N=1	5	1	5	1	1	2
N=2	4	1	5	4	1	4
N=3	1	1	5	4	1	2
N=4	4	1	5	4	2	4
N=5	2	1	5	4	2	2

Table 6.6: Number of baselines each labeling method outperforms on Average Match@N on top levels

The good performance on average of the queries does not mean that the KEA method always achieves the highest score on each individual query. For example, in Figure 6.5, on the query “jaguar” when N=5, the mapping method and the be-

	betweenness	graphpagerank	KEA	titling	docpagerank	mapping
N=1	5	1	5	1	1	4
N=2	4	1	5	2	1	4
N=3	1	1	5	4	1	2
N=4	2	1	5	3	1	5
N=5	2	1	5	5	3	3

Table 6.7: Number of baselines each labeling method outperforms on Average P@N on top levels

	betweenness	graphpagerank	KEA	titling	docpagerank	mapping
N=1	5	1	5	1	1	4
N=2	5	1	5	2	1	3
N=3	5	1	5	2	1	3
N=4	5	1	5	2	1	3
N=5	5	1	5	2	1	3

Table 6.8: Number of baselines each labeling method outperforms on Average MRR@N on top levels

	betweenness	graphpagerank	KEA	titling	docpagerank	mapping
N=1	5	1	5	1	1	2
N=2	5	1	5	1	1	3
N=3	3	1	5	2	1	2
N=4	4	1	5	2	1	3
N=5	4	1	5	2	1	2

Table 6.9: Number of baselines each labeling method outperforms on Average MTRR@N on top levels

tweenness method both have the same match@5 and P@5 scores, and even higher MRR@5 and MTRR@5 score than the KEA method. The high average score of the KEA method on all queries may due to the fact that it performs much better on some particular data sets than the other methods. For example, the metric scores of the query "AVP" is shown in Figure 6.6. One can clearly see that the KEA method has the best, and much larger scores on all four metrics. Therefore, the average score of the metrics over all queries is not enough to determine if a labeling method is better than the others.

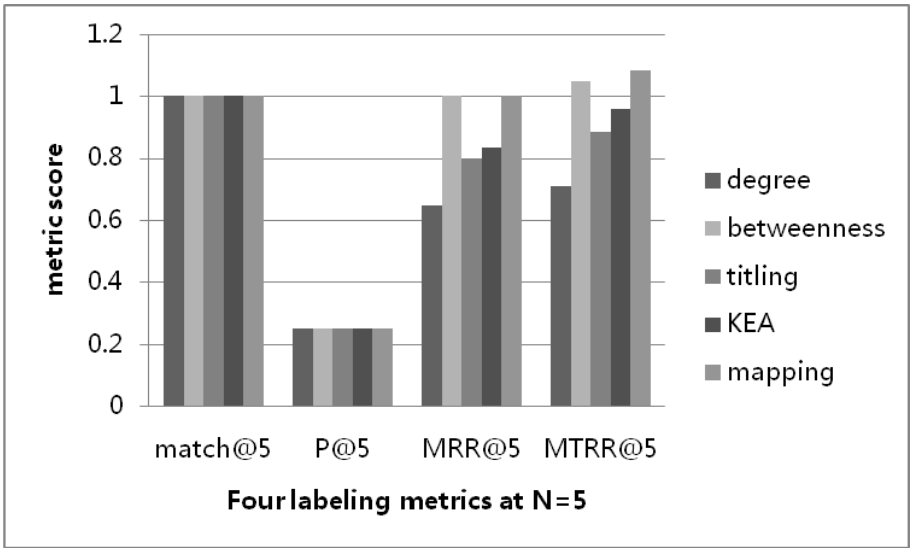

Figure 6.5: Labeling metric scores at N=5 of some labeling methods under the query *jaguar* on the top level

To avoid the unfairness in measuring with averages, we also count the number of queries where a method prevails the other methods on each of the four metrics with N ranging from 1 to 5 with a step of 1 (shown in Table 6.10, 6.11, 6.12, and 6.13). If two labeling methods have the same score on the same query and the same N, they are both counted as the prevailing method of a certain metric. Each time a labeling method gets the highest score is called a win. The largest number of wins under each N is shown in bold in the tables. From these four tables we can see that KEA wins on all the 20 comparisons. It shares the first place 9 times with other methods.

To see if a combination of these methods gets better performance as described in Section 4.5.5, we combine the best winning methods: KEA, betweenness, and

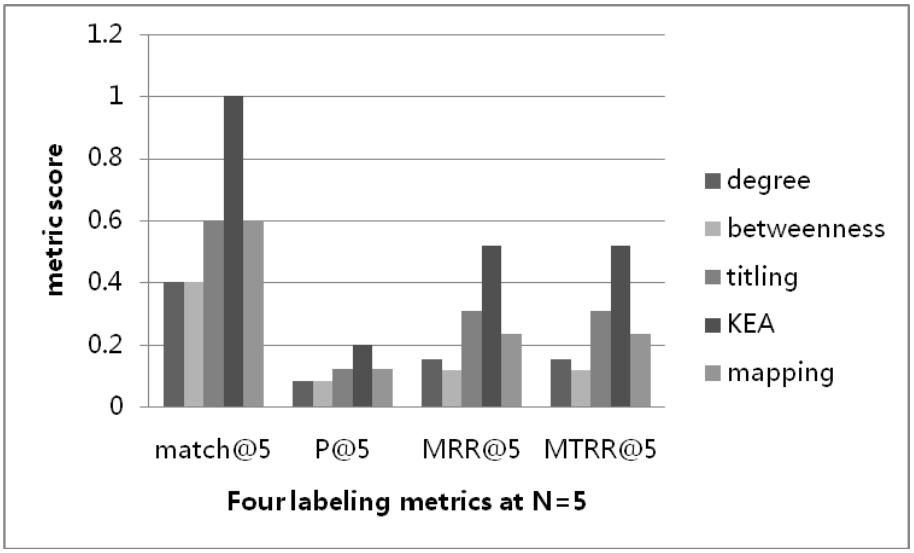

Figure 6.6: Labeling metric scores at N=5 of some labeling methods under the query *AVP* on the top level

the docfandp method. We have found that giving these three labeling methods the same weight does not generate better labels than the KEA method. We have also tried the other weighing schemes such as giving KEA twice the weight of the other methods, but all of them fail to outperform KEA. The combination method has to be done after all the other methods and adds extra time in cluster labeling.

	N=1	N=2	N=3	N=4	N=5
degree	1	1	0	1	1
betweenness	**4**	**3**	1	2	2
graphpagerank	2	1	1	1	1
tf	3	2	1	1	2
df	2	**3**	1	2	2
tfidf	2	2	2	1	1
kea	**4**	**3**	**4**	**4**	**4**
titling	2	2	2	2	3
docpagerank	1	1	1	1	2
docfandp	**4**	2	2	1	2
mapping	3	**3**	2	3	3

Table 6.10: Number of queries where each method is the prevailing method on Match@N on top levels

From the above experiments and evaluations we can see that KEA is always the best among all six our labeling methods and the five baselines in terms of the

	N=1	N=2	N=3	N=4	N=5
degree	1	1	0	1	1
betweenness	**4**	2	0	1	2
graphpagerank	2	1	0	1	1
tf	3	1	1	1	2
df	2	3	1	2	2
tfidf	2	1	1	0	0
kea	**4**	**4**	**5**	**5**	**5**
titling	2	1	1	1	3
docpagerank	1	1	0	0	1
docfandp	**4**	1	2	1	2
mapping	3	3	2	3	4

Table 6.11: Number of queries where each method is the prevailing method on P@N on top levels

	N=1	N=2	N=3	N=4	N=5
degree	1	1	0	0	0
betweenness	**4**	**4**	2	2	2
graphpagerank	2	2	1	1	1
tf	3	3	2	2	2
df	2	2	1	1	1
tfidf	2	2	2	1	1
kea	**4**	**4**	**4**	**4**	**4**
titling	2	2	2	2	2
docpagerank	1	1	0	0	0
docfandp	**4**	3	3	2	2
mapping	3	3	2	2	2

Table 6.12: Number of queries where each method is the prevailing method on MRR@N on top levels

	N=1	N=2	N=3	N=4	N=5
degree	1	1	0	0	0
betweenness	**4**	2	0	0	0
graphpagerank	2	1	0	0	0
tf	3	1	0	0	0
df	2	2	1	1	1
tfidf	2	1	1	0	0
kea	**4**	**4**	**4**	**4**	**4**
titling	2	1	1	1	1
docpagerank	1	1	0	0	0
docfandp	**4**	1	1	0	0
mapping	3	2	1	1	1

Table 6.13: Number of queries where each method is the prevailing method on MTRR@N on top levels

average metric scores, or the number of crossed baselines, or the winning times. In Table 6.14 we show the top 5 labels picked by KEA for each query sense in the five data sets. Beside each label is the number of users that chose it as the cluster label in the user survey. We also show the top selected labels by the users for each cluster in Table 6.14, along with the number of users who have picked it as the cluster label.

6.2.2 Cluster Labeling performance on the lower levels

On the lower levels clusters we use all the subtopics that have labeling ground truth from the user survey. For each query; we average the metric scores over the query senses that have subtopics. For example, for the query *Michael Jordan*, only the query sense *Michael Jordan Basketball Player* has sub-clusters according to our clustering method, we average the evaluation scores of these sub-clusters and then use it as the score for the query *Michael Jordan*.

The four metric scores (match@N, P@N, MRR@N, MTRR@N) of different labeling methods on the lower levels (subtopics of query senses) are displayed in Figure 6.7, 6.8, 6.9 and 6.10. We can see that comparing to the top levels shown in Figure 6.1, 6.2, 6.3, and 6.4, the metric scores on the lower levels are much lower. The KEA method does not have an obvious advantage, and the degree method, which is the worst method on the top levels, performs the best on the MRR@N, and MTRR@N metrics on the lower levels.

Query	Query Sense	Most selected label by the users	Our labels by the KEA method
jaguar	animal	animal-7	animals (7), cat (1), habitats (1), species (1), leopard (0)
	car	jaguar car-7	cars (3), jaguar car (7), dealer (1), history (1), sport car (3)
	Mac OS	mac os-7	OS (5), mac (3), mac OS (7), apples (2), window (0)
	guitar	guitar-9	guitars (9), fenders (2), pickup (0), neck (0), bridges (0)
penguin	Pittsburgh hockey team	pittsburgh penguin-7	pittsburgh (3), pittsburgh penguin (7) , teams (1), hockey (5), league (1)
	publisher	book-6 publisher-6	books (6), publishers (6), penguin book (3), imprints (1), penguin group(2)
	kids club	club penguin-7	clubs (1), club penguin (7), kid (1), games (0), online (0)
	algorithm	google penguin algorithm-6	google (3), algorithms (3), updates (2), google penguin (3), algorithm update (2)
AVP	volleyball	volleyball-8	volleyball (8), beaches (0), tours (1), beach volleyball (6), sport (2)
	antivirus software	antivirus-8	kaspersky (2), viruses (2), software (5), antivirus (8), kaspersky lab (1)
	Avon	avon product-7	avon (5), avon product (7), product (2), market (0), stock (3)
	movie	movie-6	predators (2), movies (6), alien (2), weyland (0), predator movie (3)
	airport	international airport-4	airports (1), scranton (1), international airport (4), avoca (2), hotel (1)
tiger	aircraft	tiger airway-8	aircraft (7), pilot (0), tiger moth (2), tiger airway (8), markets (0)
	Woods	golf-6	woods (1), tiger wood (1), golf (6), opens (1), tournament (1)
	animal	animal-8	animal (8), cubs (0), habitat (0), species (2), cat (0)
	hash	tiger hash algorithm-7	hash (4), tiger hash (6), hash function (6), function (1), file (0)
Michael Jordan	basketball player	basketball-7	basketball (7), player (1), NBA[national basketball association] (5), basketball player (5), games (0)
	Berkeley researcher	machine learning-11	research (3), university (3), machine learning (11), learning (1), berkeley (1)

Table 6.14: Top cluster labels by the users, and by our KEA method for each query sense with the number of users who picked each label

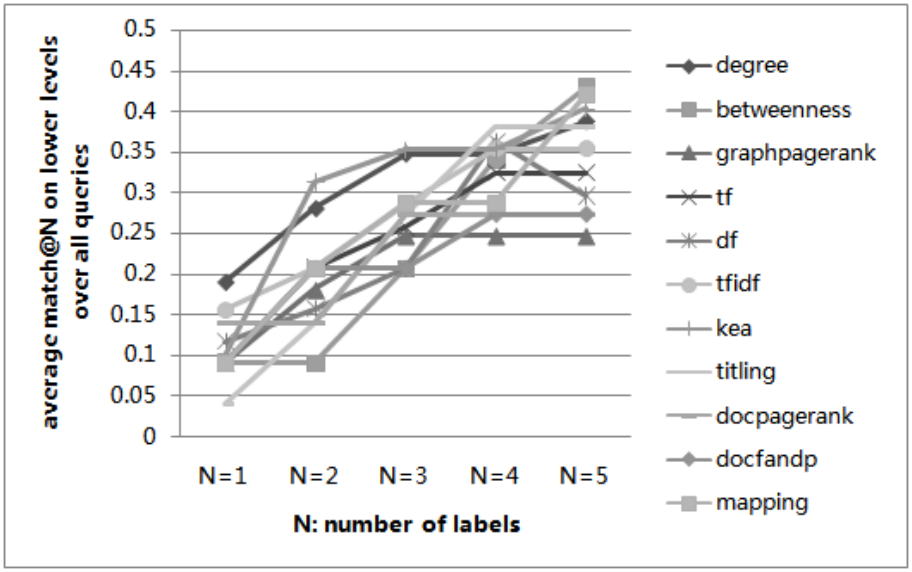

Figure 6.7: Average match@N on lower levels over all queries of different labeling methods

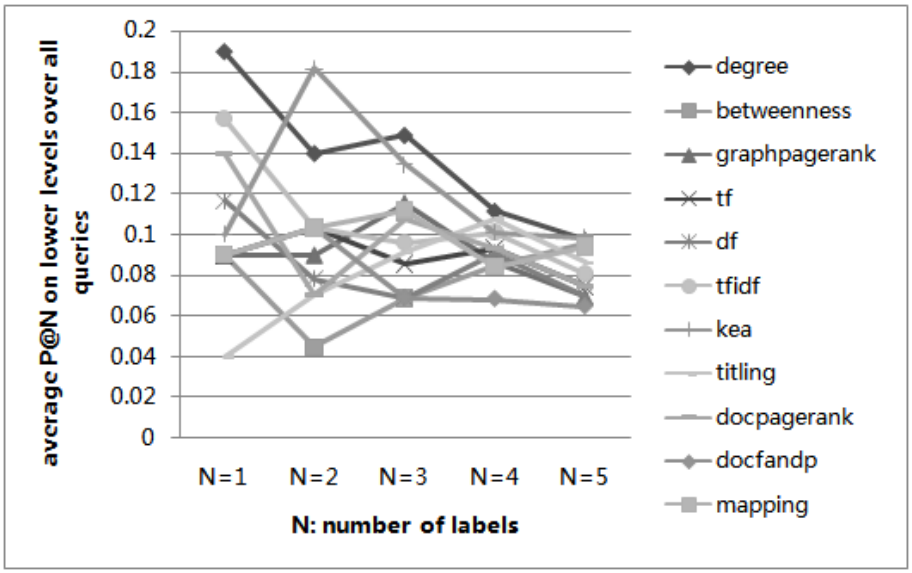

Figure 6.8: Average P@N on lower levels over all queries of different labeling methods

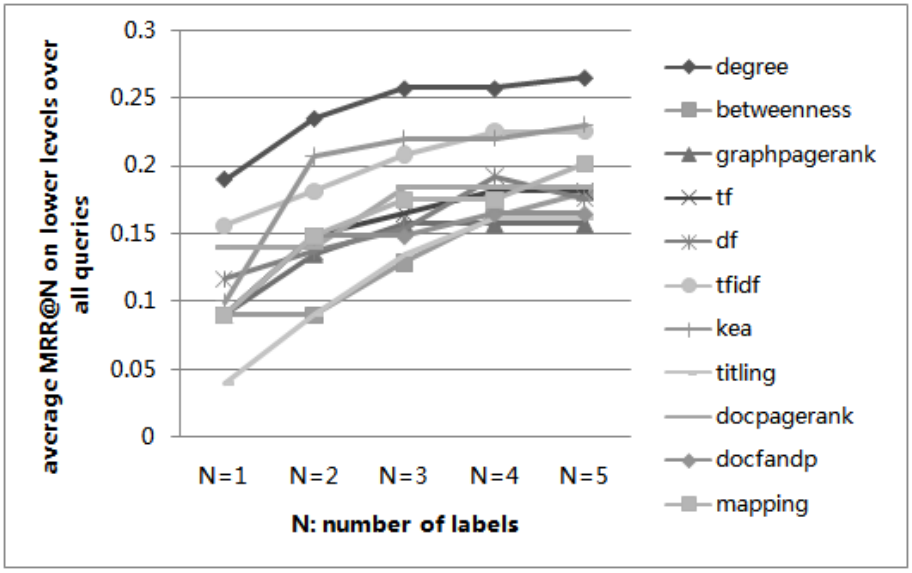

Figure 6.9: Average MRR@N on lower levels over all queries of different labeling methods

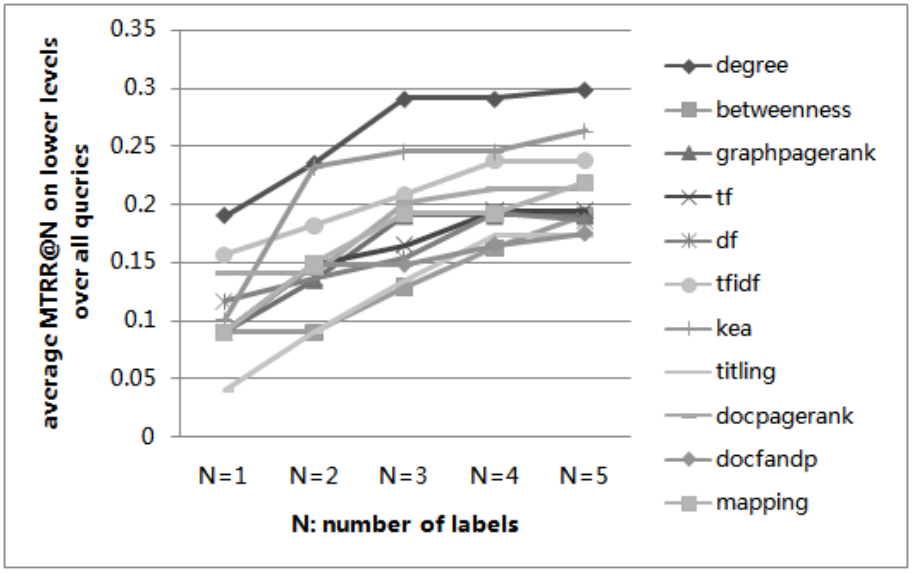

Figure 6.10: Average MTRR@N on lower levels over all queries of different labeling methods

We also count the number of queries where each labeling method is the prevailing method. The results are shown in Table 6.15, 6.16, 6.17 and 6.18. We can see that unlike on the top levels, none of the methods prevails over other methods on all the queries and all the metrics on the lower levels. If we add the wins up we can see that the TFIDF method is the prevailing method 11 times, followed by the mapping method with 10 times; both the degree method and the KEA method win 9 times. The best baseline on the top levels, the docfandp method, only wins once. Overall, the KEA method performs the best on the top levels and acceptable on the lower levels.

	N=1	N=2	N=3	N=4	N=5
degree	**3**	**4**	1	1	**2**
betweenness	2	1	0	1	**2**
graphpagerank	2	3	0	0	0
tf	2	2	1	1	1
df	2	1	0	1	1
tfidf	**3**	2	1	**2**	**2**
kea	2	**4**	**2**	1	**2**
titling	2	2	1	1	1
docpagerank	3	2	1	0	0
docfandp	**2**	2	0	0	0
mapping	2	**2**	**2**	1	**2**

Table 6.15: Number of queries where each method is the prevailing method on match@N on the lower levels

In Table 6.19 we show the top 5 labels picked by the users for some subtopics. Beside each label is the number of users that chose it as the cluster label in the user survey. We can see that the label with the most user agreement is the "jaguar animal rescue center" subtopic with 6 votes for the label " jaguar rescue center". Less than half of the users agree with the other labels. This means that even for human users it is hard to give accurate labels on the lower levels. One reason is that it is hard to differentiate one subtopic from another since the subtopics are in fact about the same topic.

	N=1	N=2	N=3	N=4	N=5
degree	**3**	3	2	2	2
betweenness	2	1	0	1	2
graphpagerank	2	2	1	1	1
tf	2	1	0	1	1
df	2	1	0	0	1
tfidf	**3**	1	1	**2**	1
kea	2	**4**	2	1	2
titling	2	1	1	1	1
docpagerank	**3**	1	1	1	1
docfandp	2	1	0	0	0
mapping	2	1	**3**	**2**	**3**

Table 6.16: Number of queries where each method is the prevailing method on P@N on the lower levels

	N=1	N=2	N=3	N=4	N=5
degree	**3**	3	1	**1**	**2**
betweenness	2	1	0	0	0
graphpagerank	2	2	0	0	0
tf	2	1	0	0	0
df	2	2	1	**1**	0
tfidf	**3**	2	**2**	**1**	0
kea	2	**3**	**2**	**1**	1
titling	2	1	0	0	0
docpagerank	**3**	2	1	**1**	0
docfandp	2	1	0	0	0
mapping	2	1	1	**1**	**2**

Table 6.17: Number of queries where each method is the prevailing method on MRR@N on the lower levels

	N=1	N=2	N=3	N=4	N=5
degree	**3**	3	1	**1**	**2**
betweenness	2	1	0	0	0
graphpagerank	2	2	0	0	0
tf	2	1	0	0	0
df	2	2	1	**1**	0
tfidf	**3**	2	**2**	**1**	0
kea	2	**3**	**2**	**0**	1
titling	2	1	0	0	0
docpagerank	**3**	1	0	**1**	0
docfandp	2	1	0	0	0
mapping	2	1	1	**1**	**2**

Table 6.18: Number of queries where each method is the prevailing method on MTRR@N on the lower levels

subtopic	top labels given by the users
Jaguar/animal/facts	largest cat-2 fact-2 habitat-2 territory-1 hunt-1
Jaguar/animal/rescue	jaguar rescue center-6 rescue center-4 jaguar animal rescue-4 natural habitat-1costa rica-1
Michael Jordan/Basketball Player/Quotes	basketball player-5 michael jordan-2 success-2 basketball-2 quote-1
Michael Jordan/Basketball Player/career	basketball-5 greatest basketball player-4 nba[national basketball association]-4 history-1 bull-1

Table 6.19: Labels given by the users on some subtopics on the lower levels

6.2.3 Running time of different labeling methods

In this section we list the running time of the labeling methods. The parts that can be done at the crawling stage before the collection of the documents are calculated in the off-line time. For example, the keyphrase extraction with KEA, and the automatic titling of each document can be done off-line. The on-line time affects the user experience. A longer on-line time would cause a delay before a user sees a taxonomy after submitting a query. All the running times shown in Table 6.20 are obtained with a 4G memory, and a 1.20GHz PC. We can see that the degree method is the fastest, and the betweenness method is the slowest. The running time of the KEA method, the best labeling method on all levels, is on the faster end. The running time of *docfandp*, the best baseline on the top levels, is the second slowest. Overall, the running time of all the labeling methods, especially the ones under 5 seconds, can be much shorter and acceptable by the users given the necessary resources.

	off-line time (s)	on-line time (s)
degree	0	1.2
betweenness	0	29.34
graphpagerank	0	2.98
tf	0	4.03
df	0	3.39
tfidf	0	2.80
kea	21	2.55
titling	7.88	1.31
docpagerank	8.01	1.73
docfandp	0	13.97
mapping	0	1.49

Table 6.20: Running time of different labeling methods

6.3 Interactive browsing interface

In this section we introduce the interface we developed for examining the quality of our document clustering and labeling methods. It is an interactive browsing web application that runs on browsers built with JavaScript, PHP, and MySql. It shows

the taxonomies of the five query data sets that we collected. It allows the user to follow the taxonomies to zoom in to the documents of interest. A snap shot of the interface is shown in Figure 6.11. The features include:

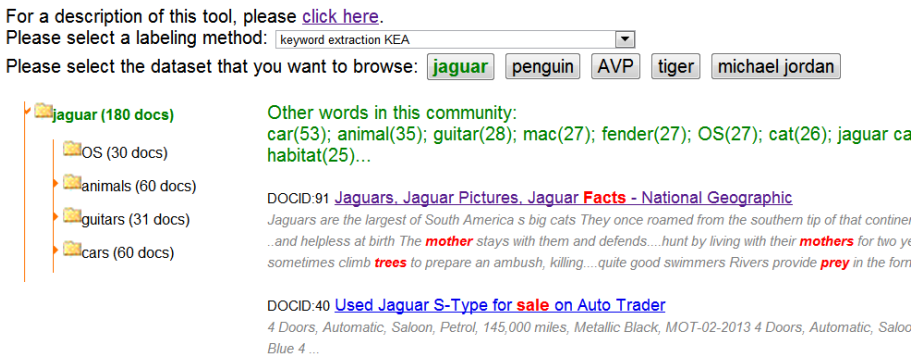

Figure 6.11: Snapshot of the document clustering and labeling interface under the query *jaguar*

Data set and labeling method selection Users can choose which labeling method they want to examine from the top drop down menu. This is mainly for debugging purpose and can be replaced with only showing the results of the best labeling method. Users can also select the query that they want to examine from the buttons at the top. The query that is selected and being displayed is bolded and shown in green. Right now it only displays the five queries that we have collected in the data collection process. However, given the resources to run the crawling and the time-consuming off-line parts, this interface can be transformed into a search engine where a user searches for a query and sees a taxonomy.

Taxonomy display The taxonomy of the query senses of the currently selected query is displayed on the left. Each folder represents a document cluster. Beside each folder icon is the top label in the label list of that cluster. According to the Comprehensibility property of cluster labels discussed in Section 3.1.2 the cluster labels are phrases. We only display one label for each cluster in the taxonomy as the other commercial systems do. However, there are situations in which only one label does not describe all the aspects in that cluster. A user might want to know other labels for a cluster. Therefore, we also display

the top five labels for a cluster if the user wants to see more. For example, if a user puts the mouse over the folder labeled "animals" in the taxonomy of "jaguar", the top five labels would be displayed as a floating box beside it, as shown in Figure 6.12.

Document and keyword display A click on any folder shows the documents and the keywords in that community. They are displayed to the right of the taxonomy. On the top are the top ten keywords in the corresponding keyword community before being post-processed. Each keyword is associated with its score under the labeling method being viewed. The scores give some evidence of why the documents in this cluster are grouped together. Below the keywords are the documents in that folder. We show the title, and the hyperlink of the web page for each document. A user can click on the hyperlink and go to the document as in a search engine. Under each document we show the first twenty words of that document as a snippet to give the user a sense of what the document is about. After the snippet, we highlight up to three keywords in this community that also appear in that document in red and bolded fonts. We also give the contexts of these matched keywords. We display five words on each side of a matched keyword. They give evidence of why a document is in a certain cluster.

Interactive browsing Our interface supports interactive browsing for the users. In the taxonomy, folders with a right arrow on the left have subtopics; folders that are already opened with the subtopics have a "down" arrow on the left; folders with no arrows on the left have no subtopics. If a folder has subtopics, a user can click on the folder and see the subtopics of that cluster. The subtopics of a topic are indented under that topic. For example, in Figure 6.13, if a user clicks on the folder labeled "animals", the two subtopics "cat" and "jaguar rescue cente" are displayed. A user can keep clicking on the folders that have subtopics on the lower levels until she reaches the one of her interest. The folder being viewed is highlighted in green. A click on a folder that is already opened will close it.

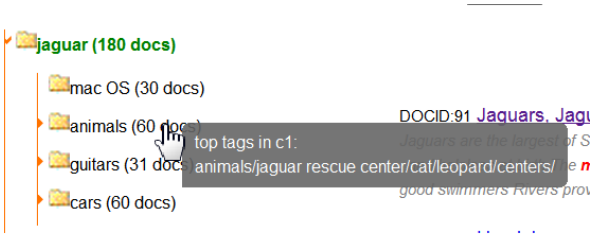

Figure 6.12: Display of the other labels in a cluster by putting the mouse over a folder

Figure 6.13: Snapshot of the document clustering and labeling interface under the query *jaguar* when the folder *animals* is clicked

We display the taxonomy our method generates for each of the queries in Figure 6.14, and 6.15. The labels we display here are by the KEA labeling method. KEA is not always the best on all queries. In Section 6.2.1 we discussed how some other methods are better than KEA on the query jaguar. For example, on the first cluster under the query jaguar, the betweenness method, and the titling method all pick "Mac OS" as the top label whereas KEA picks "OS" while the former is a better label than the latter. However, KEA does have the overall best performance and that is why we display the taxonomies under the KEA labeling method. We can see that the taxonomies we generate are rather compact, and the query senses are well separated and described by meaningful labels on the top levels. On the lower levels, however, our method generates more clusters than we expect, and sometimes the labels are not distinctive enough from the siblings (see the cluster labels under "volleyball" under the query "AVP"). From the evaluation of the experimental results in previous sections, we realize that proper clustering and labeling on the lower levels is the limitation of our work. Our method works well in disambiguating the different senses of the same query, but not so much in separating different subtopics

of the same topic.

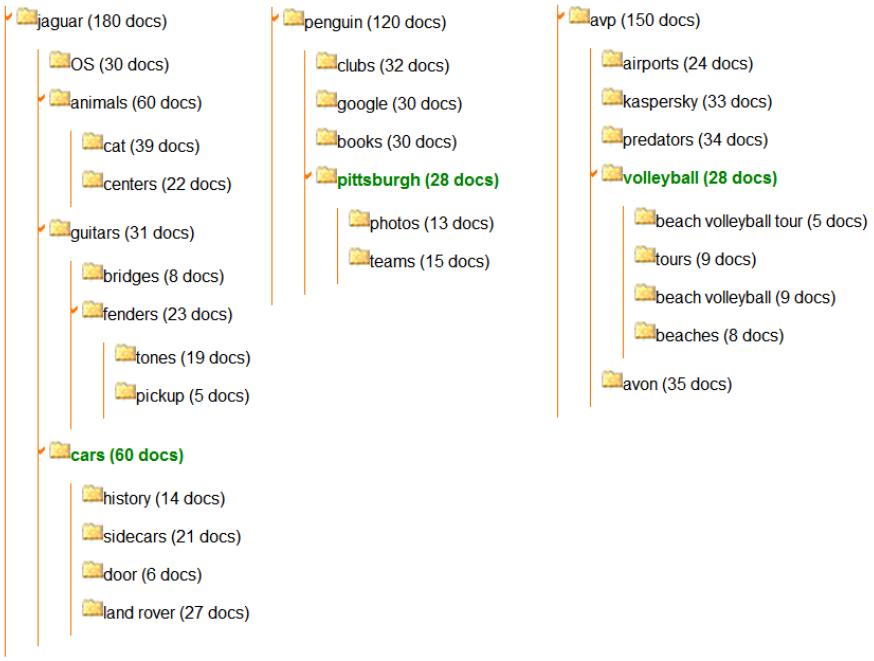

Figure 6.14: Our taxonomy on the query *jaguar*, *penguin*, and *AVP*

6.4 Obesity data results and discussion

Apart from the web search results data sets, we gathered a data set of blog posts on the topic of obesity (see Section 5.1.1). The documents in this obesity data set do not have clustering ground truth, and it is too domain-specific to get labeling ground truth. In this section, we display the taxonomy we generated on this data set, and discuss issues our method has on this data set.

Figure 6.16 is the taxonomy of the obesity data set with, and without the post-processing that utilizes author labels described in Section 4.5.7. For ease of presentation we only show three layers. One thing worth noting is that the value of the threshold t_Q on this data set is different than the 0.3 that we use on the web search results. The reason is that $t_Q = 0.3$ does not generate any clusters on the documents as the max Q modularity score on this data set is less than 0.3. We lowered t_Q accordingly and set it to 0.15. According to the feedback of the author of the blog posts Dr. Sharma, the clustering is not perfect. In his opinion, in the commu-

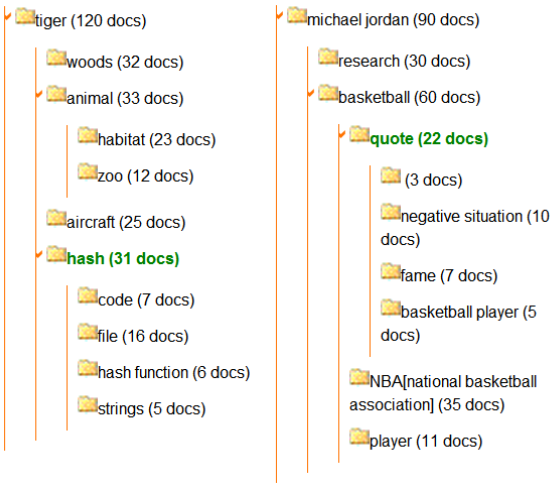

Figure 6.15: Our taxonomy on the query *tiger*, and *Michael Jordan*

nity labeled "kids", some documents are actually about "adults", some are general documents. We have yet to know how bad this imperfection affects the browsing process for the blog′s readers.

Comparing with the web search results, the keyword graph generated on the obesity data set is very dense. Take the "jaguar" search results as an example, some characteristics on this set and the obesity data set are shown in Table 6.21. We can see that in the obesity data, the average degree in the keyword graph is larger which indicates a denser graph, the max Q is smaller which indicates a weak community structure, and the documents that are mapped to multiple communities have a higher proportion which indicates more overlaps between different communities. The reason may be that the topics are very similar and the author is using the same vocabulary over and over again on different topics. For example, even though "childhood obesity" and "disease caused by obesity" can be two different topics but they both involve with terms such as "diet", "exercise", and "disease" etc. It is hard for our algorithm to detect communities in such a dense graph with a weak community structure.

The author labels associated with some documents may be useful but we found that it is hard to use them as ground truth for clustering or labeling. First, not all the documents have author labels. Second, the author labels rarely appear in the documents directly. Lastly, the author labels overlap between documents of differ-

Figure 6.16: Our Taxonomy on the obesity data set without using author labeling in post-processing

	Jaguar	obesity
#edges/#nodes in the keyword graph	10.85	33.25
max Q-modularity	0.493	0.178
#overlapping documents/#documents in the collection	0.02	0.12

Table 6.21: characteristics on the jaguar data set and the obesity data set

ent topics. For example, a document about discrimination policy is labeled with "policy, discrimination, public transport", another document about hospitalization is labeled with "cardiovascular disease" and "policy". There is no obvious reason why these two should be in the same community. We have also tried to group documents just based on their author labels and then generate the taxonomy for each label. However, the results do not separate different topics very well. The taxonomy of all the documents with the label "kids" is shown in Figure 6.17. We do not see a clear separation between the topics of different communities.

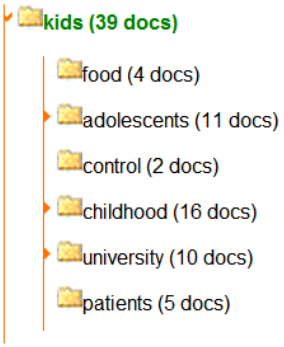

Figure 6.17: Taxonomy of the documents with the author label "kids" in the obesity data set

Part IV

Conclusions

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis we use a co-clustering ATG method based on the sentence co-occurrences of frequent keywords to generate a hierarchy of topics for a document collection. We propose and experiment on different labeling methods based on centrality measures in each keyword community, the important terms in each document cluster, the connection between these two corresponding communities, and the combination of the methods.

In part one of this thesis we introduced various attempts in organizing documents, especially search results, to a better presentation than ranked lists. We showed that post-retrieval clustering and labeling (also known as ATG) is a better method than query refinement recommendation and pre-retrieval classification. Then we presented a survey of ATG, including the desired properties of a good taxonomy, and the three main ATG approaches that are document-based, word-based, and co-clustering based.

In part two we detailed our approach in the clustering and labeling on this task. The first four phases of our approach: keyword extraction, keyword graph generation, community mining and the mapping of the documents generate hierarchical clusters for a collection of documents and address the first thesis statement. The final phase which is cluster labeling introduces different labeling methods that we proposed to use. We address the second thesis statement by using the Betweenness Centrality and PageRank on the keyword communities to extract cluster labels and

compare them to labeling with Degree Centrality. The labeling methods using automatic titling, keyphrase extraction with the KEA method and PageRank proposed in that final phase address the third thesis statement.

Our results showed that on clustering different senses of the same query (top levels in the taxonomy), our method performs well comparing to K-Means on two metrics ARI and Cluster Contamination. However, on separating different subtopics of the same topics which is at lower levels in the taxonomy, our method does not outperform K-Means. In terms of cluster labeling, the KEA labeling method achieves the best overall performances no matter on the top levels or the lower levels.

7.2 Summary of Contributions

This MSc thesis makes the following contributions:

1. A co-clustering based ATG algorithm based on Chen et al.′s work on web search result categorization. It works well in disambiguation different senses, but our evaluations revealed that the ability to separate different subtopics of the same topic well is a limitation of this work.

2. Different labeling algorithms based on the keyword communities, the document clusters, and the connection between these two kinds of communities. Amongst them, the labeling method that extracts keyphrases from the documents with KEA has the best overall performance.

3. A detailed data collection and pre-processing method for collecting web search results, and a user survey to collect cluster labeling ground truth.

4. A detailed post-processing method for the cluster labels.

5. An interactive browsing interface for examining the taxonomies.

7.3 Future Work

To represent a set of documents into a hierarchy is very useful to the users. Our method works well in search result clustering for identifying different query senses,

but we also want good separations between different subtopics of the same topic. There are also other features that can be added to the interactive browsing interface. Many things can be done to extend our work in this thesis:

1. In phase I of our approach, different keyword extraction methods can be used. For example, in clustering based on documents, researchers have found that taking 10-20 terms from each document as features generates good results [11]. This can be utilized to replace or to be combined with the DF cutoff that we use.

2. In phase II of our approach, one can assign different edge weights other than co-occurrences. Even though co-occurrences represent the correlations between different terms to a certain extent, our experiments have shown that keyword graphs with co-occurrences as edge weights do not separate different subtopics of the same topic well may be due to the fact that the subtopics use very similar vocabulary. Other semantic similarity measures can be used as edge weights such as Mutual Information or Pair-wise Mutual Information.

3. When generating document clusters (phase IV), we can use the fact that some documents belong to multiple communities to merge communities. We can also utilize this mapping method to delete some candidate tags. For example, if a term does not have strong connections with the documents in the corresponding document cluster, it should not be used as a label.

4. Cluster labeling in phase V of our work is still an open problem. One possible venue is to come up with an optimized combination method that takes the strength of some good labeling method and generates even better labels.

5. Other external sources can be applied in cluster labeling. For example, one can utilize the HTML structure of the web pages and use the caption of the images and tables to get the topics. We did not explore this in our work because not every document collection is well-structured with HTML. Besides, one can use some ontologies to get the is-a relationship between terms. In a

good taxonomy, the topic of a cluster should be subsumed by its parent cluster. Using an external source may be able to identify such relationships. One could also utilize the author labels in other ways than the ones we have tried.

Regarding the interactive browsing interface, other features can be added to it:

1. Given the necessary resources, the interface can be made into a search engine with better search result representations than ranked lists. A user can submit a query and sees a taxonomy right away.

2. Reflect the dynamic changes in the taxonomies. A document collection could change over time, as well as the search results for a certain query. A user might want to see the trends of the topics over time. A slider can be added to the interface to reflect the change of topics in the taxonomy.

3. An important application of this work is search result clustering. This is even more important on mobile devices. This interface can be built into a mobile application that saves the time and energy in scrolling down the screen to find the relevant results, and saves data usage for the users.

Bibliography

[1] R.B. Allen, P. Obry, and M. Littman. An interface for navigating clustered document sets returned by queries. In *Proceedings of the conference on Organizational computing systems*, pages 166–171. ACM, 1993.

[2] Giuseppe Attardi, Sergio Di Marco, and Davide Salvi. Categorisation by context. *Journal of Universal Computer Science*, 4(9):719–736, sep 1998.

[3] D. Austin. How google finds your needle in the webs haystack. *American Mathematical Society Feature Column*, 2006.

[4] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747, 2004.

[5] R. Berendsen, B. Kovachev, E. Nastou, M. de Rijke, and W. Weerkamp. Result disambiguation in web people search. In *ECIR 2012: 34th European Conference on Information Retrieval*, Barcelona, 2012.

[6] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[7] D. Carmel, H. Roitman, and N. Zwerdling. Enhancing cluster labeling using wikipedia. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146. ACM, 2009.

[8] C. Carpineto, A. Bernardini, M. DAmico, and G. Romano. New research directions in search results clustering. *Proceedings of the First Italian Information Retrieval Workshop*, page 17, 2010.

[9] C. Carpineto and G. Romano. Optimal meta search results clustering. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 170–177. ACM, 2010.

[10] Claudio Carpineto, Stefano Mizzaro, Giovanni Romano, and Matteo Snidero. Mobile information retrieval with search results clustering: Prototypes and evaluations. *J. Am. Soc. Inf. Sci. Technol.*, 60(5):877–895, May 2009.

[11] H.C. Chang and C.C. Hsu. Using topic keyword clusters for automatic document clustering. In *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on*, volume 1, pages 419–424. Ieee, 2005.

[12] Hao Chen and Susan Dumais. Bringing order to the web: automatically categorizing search results. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, pages 145–152, New York, NY, USA, 2000. ACM.

[13] J. Chen, O. Zaiane, and R. Goebel. Web Search result categorization based on query sense communities. *Unpublished manuscript*, 2008.

[14] S.Y. Chen, C.N. Chang, Y.H. Nien, and H.R. Ke. Concept extraction and clustering for search result organization and virtual community construction. *Computer Science and Information Systems*, 9(1):323–355, 2012.

[15] O.S. Chin, N. Kulathuramaiyer, and A.W. Yeo. Automatic discovery of concepts from text. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 1046–1049. IEEE Computer Society, 2006.

[16] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[17] Aaron Clauset. Fast-modularity made really fast. http://cs.unm.edu/~aaron/blog/archives/2007/02/fastmodularity.htm, 2007.

[18] W.B. Croft. *Organizing and searching large files of documents*. PhD thesis, University of Cambridge, 1978.

[19] Hang Cui. Web search result re-organization with ontologies. Master's thesis, University of Alberta, Canada, 2001.

[20] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, pages 318–329, New York, NY, USA, 1992. ACM.

[21] W. Dawid. *Descriptive Clustering as a Method for Exploring Text Collections*. PhD thesis, Poznan University of Technology, Poznań, Poland, 2006.

[22] I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[23] J. Fagnan. community mining: from discovery to evaluation and visualization. Master's thesis, University of Alberta, Canada, 2012.

[24] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. *Software: Practice and Experience*, 38(2):189–225, 2008.

[25] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

[26] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36, 2007.

[27] L.C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[28] H. Frigui and O. Nasraoui. Simultaneous categorization of text documents and identification of cluster-dependent keywords. In *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, volume 2, pages 1108–1113. IEEE, 2002.

[29] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on artificial intelligence*, volume 6, page 12. Morgan Kaufmann Publishers Inc., 2007.

[30] E. Glover, D.M. Pennock, S. Lawrence, and R. Krovetz. Inferring hierarchical descriptions. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 507–514. ACM, 2002.

[31] M.A. Hearst. Automated discovery of wordnet relations. *WordNet: an electronic lexical database*, pages 131–151, 1998.

[32] M.A. Hearst and J.O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84. ACM, 1996.

[33] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. Determining the user intent of web search engine queries. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1149–1150, New York, NY, USA, 2007. ACM.

[34] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

[35] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. A local search approximation algorithm for k-means clustering. In *Computational Geometry: Theory and Applications*, volume 28, pages 89–112, 2004.

[36] Reihaneh Rabbany khorasgani. community mining and its applications in educational environment. Master's thesis, University of Alberta, Canada, 2010.

[37] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[38] H. Kommanti and C. Raghavendra Rao. Association rule centric clustering of web search results. *Multi-disciplinary Trends in Artificial Intelligence*, pages 159–168, 2011.

[39] K. Krishna and R. Krishnapuram. A clustering algorithm for asymmetrically related data with applications to text mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 571–573. ACM, 2001.

[40] R. Krishnapuram and K. Kummamuru. Automatic taxonomy generation: Issues and possibilities. *Fuzzy Sets and SystemsIFSA 2003*, pages 184–184, 2003.

[41] K. Kummamuru, A. Dhawale, and R. Krishnapuram. Fuzzy co-clustering of documents and keywords. In *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, volume 2, pages 772–777. IEEE, 2003.

[42] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th international conference on World Wide Web*, pages 658–665. ACM, 2004.

[43] D. Lawrie, W.B. Croft, and A. Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 349–357. ACM, 2001.

[44] R. Levering and M. Cutler. The portrait of a common html web page. In *Proceedings of the 2006 ACM symposium on Document engineering*, pages 198–204. ACM, 2006.

[45] Digital Libraries and Machine Learning Labs. Kea:description. http://www.nzdl.org/Kea/description.html.

[46] Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 366–376. Association for Computational Linguistics, 2010.

[47] C. Lopez, V. Prince, and M. Roche. Automatic titling of electronic documents with noun phrase extraction. In *Soft Computing and Pattern Recognition (SoCPaR)*, pages 168–171. IEEE, 2010.

[48] Cédric Lopez, Violaine Prince, and Mathieu Roche. Automatic titling of articles using position and statistical information. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 727–732, Hissar, Bulgaria, September 2011. RANLP 2011 Organising Committee.

[49] B. Mandhani, S. Joshi, and K. Kummamuru. A matrix density based algorithm to hierarchically co-cluster documents and words. In *Proceedings of the 12th international conference on World Wide Web*, pages 511–518. ACM, 2003.

[50] C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[51] Q. Mei, X. Shen, and C.X. Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 490–499. ACM, 2007.

[52] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4. Barcelona: ACL, 2004.

[53] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[54] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[55] A. ONeill. *Sentiment mining for natural language documents*. COMP3006 Project Report. Department of Computer Science, Australian National University. Retrieved from http://users. cecs. anu. edu. au/˜ ssanner/Papers/Alex_Report. pdf, 2009.

[56] T. Opsahl, F. Agneessens, and J. Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, 2010.

[57] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank citation ranking: Bringing order to the web.* Stanford InfoLab, 1999.

[58] A. Popescul and L.H. Ungar. Automatic labeling of document clusters. *Unpublished manuscript, available at http://citeseer. nj. nec. com/popescul00automatic. html*, 2000.

[59] Montserrat Mateos Sanchez, Encarnacin Beato Gutirrez, Roberto Berjn Gallinas, Ana Ma, Fermoso Garca, Miguel Angel, Snchez Vidales, and Carlos Garca figuerola Paniagua. Clustering of web documents : Full-text or snippet. pages 488–493. Solutions, 2008.

[60] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213. ACM, 1999.

[61] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita. Topical clustering of search results. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 223–232. ACM, 2012.

[62] O. Shamir, A. Sivan Sabato, and N. Tishby. Learning and generalization with the information bottleneck method. *Inter- national Symposium on AI and Mathematics (ISAIM-2008)*, 2008.

[63] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data mining and knowledge discovery*, 2(1):39–68, 1998.

[64] Amit Singhal. Official google blog: Introducing the knowledge graph: Things, not strings. http://googleblog.blogspot.ca/2012/05/introducing-knowledge-graph-things-not.html, May 2010.

[65] Ruihua Song, Zhenxiao Luo, Ji-Rong Wen, Yong Yu, and Hsiao-Wuen Hon. Identifying ambiguous queries in web search. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1169–1170, New York, NY, USA, 2007. ACM.

[66] Sofia Stamou, Vlassis Krikos, Pavlos Kokosis, and Ros Ntoulas. directory construction using lexical chains. In *In Proceedings of the 10 th NLDB Conference 2005*, pages 138–149, 2005.

[67] H. Toda and R. Kataoka. A clustering method for news articles retrieval system. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 988–989. ACM, 2005.

[68] P. Treeratpituk and J. Callan. Automatically labeling hierarchical clusters. In *Proceedings of the 2006 international conference on Digital government research*, pages 167–176. ACM, 2006.

[69] Y.H. Tseng, C.J. Lin, H.H. Chen, and Y.I. Lin. Toward generic title generation for clustered documents. *Information Retrieval Technology*, pages 145–157, 2006.

[70] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AAAI*, pages 855–860, 2008.

[71] X. Wang and M. Bramer. Exploring web search results clustering. *Research and Development in Intelligent Systems XXIII*, pages 393–397, 2007.

[72] 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*, pages 254–255. ACM, 1999.

[73] W.A. Woods. *Conceptual indexing: A better way to organize knowledge*. Sun Microsystems, Inc., 1997.

[74] Y. Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 1995.

[75] Z. Ying and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the International Conference on Information and knowledge Management, New York*, pages 515–524, 2002.

[76] K.Y. Yip, D.W. Cheung, and M.K. Ng. Harp: A practical projected clustering algorithm. *Knowledge and Data Engineering, IEEE Transactions on*, 16(11):1387–1397, 2004.

[77] Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to Web search results. In *Proceedings of the eighth international conference on World Wide Web*, WWW '99, pages 1361–1374, New York, NY, USA, 1999. Elsevier North-Holland, Inc.