2010

Mobile Search Engine using Clustering and Query Expansion

Huy Nguyen
San Jose State University

Mobile Search Engine using Clustering and Query Expansion

A Writing Project
Presented to
The Faculty of the Department of Computer Science
San Jose State University
In Partial Fulfillment
of the Requirements for the Degree
Master of Science
by
Huy Nguyen
Spring 2010
Advisor: Dr. Robert Chun

ABSTRACT

Internet content is growing exponentially and searching for useful content is a tedious task that we all deal with today. Mobile phones lack of screen space and limited interaction methods makes traditional search engine interface very inefficient. As the use of mobile internet continues to grow there is a need for an effective search tool. I have created a mobile search engine that uses clustering and query expansion to find relevant web pages efficiently. Clustering organizes web pages into groups that reflect different components of a query topic. Users can ignore clusters that they find irrelevant so they are not forced to sift through a long list of off-topic web pages. Query expansion uses query results, dictionaries, and cluster labels to formulate additional terms to manipulate the original query. The new manipulated query gives a more in depth result that eliminates noise. I believe that these two techniques are effective and can be combined to make the ultimate mobile search engine.

Table of Contents

List of Tables

List of Figures

1.0 INTRODUCTION

This section explains the growth of mobile devices and the increased use of mobile internet. It also argues the importance of mobile internet and its potential. The problem addressed by this project is discussed in section 1.3.

1.1 Growth of Mobile Devices

Mobile devices are becoming increasingly popular around the world. The lack of wired line infrastructure in developing countries has resulted in a widespread use of mobile technology. The ease and efficiency of wireless technology has allowed people to take advantage of information and communication technologies via mobile phones without land lines. Because of this there are many development projects currently underway and the growth of mobile technology is endless [1]. Mobile phones provide the opportunity and growth that can transform how people live their daily lives in countries where information and communication technology are limited. Mobile phones are inexpensive and provide vital information and communication access anywhere any time. According to a report from the International Telecommunication Union, the leading UN agency for information and communication technology, “Mobile cellular has been the most rapidly adopted technology in history. Today it is the most popular and widespread personal technology on the planet, with an estimated 4.6 billion subscriptions globally by the end of 2009” [2]. Figure 1 shows the percentage of people who have fixed broadband subscriptions, mobile broadband subscriptions, fixed telephone lines, internet users and mobile telephone subscriptions over a span of a decade [2]. As you can see from the graph, in 2009 67% of the world is estimated to have a mobile telephone subscription.

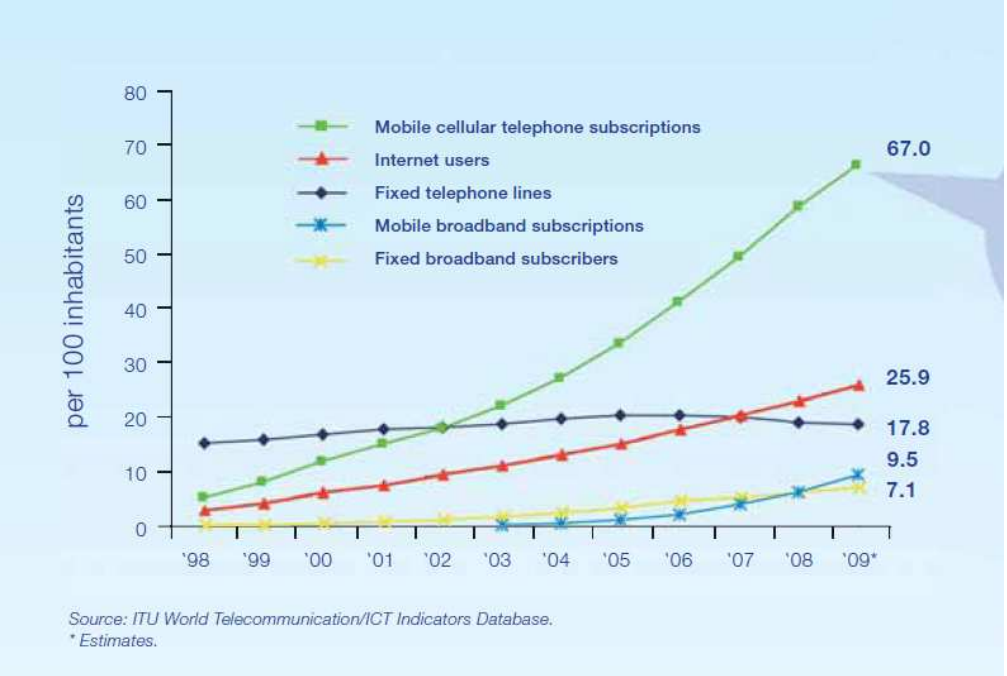


Figure 1. Percentage of different subscriptions over a span of a decade Source: [2]

1.2 Popularity of Mobile Internet

The internet is a powerful tool that is changing the world and is steadily growing. Looking at Figure 1 you can see that by 2009 over a quarter of the world's population will be using the internet. Even though that may seem like a large number, I think it is too small, especially because of what the internet can provide to its community and improve their way of living. The reasons that a majority of people do not have access to the internet is due to the cost of computers and the lack of wired line infrastructure that provides broadband services. Mobile phones are cheaper than computers and can access the internet through a wireless network. Mobile phones have a lower maintenance cost and there is no need to worry about viruses and spyware. Initially mobile phone had limited memory and processing power, and the cost of internet access was expensive with service providers charging by the kilobyte. Today mobile phones do high performance processing and use high speed data communication comparable to

traditional broadband services. Mobile service providers are now providing more attractive billing plans with flat-rate billing that encourages high-volume usage rather than penalize it. This cuts down the cost for users who frequently use their mobile device to access the internet and promotes a new age of anytime-information-access. It is apparent that the increase in mobile internet will change how people look for and interact with information [3].

With great bandwidth and more attractive billing arrangements more and more users are using mobile internet on a daily basis [3]. The graph in Figure 1 shows that in 2009 mobile broadband subscription is over 600 million and has surpassed landline broadband subscription [2]. Even with 600 million mobile broadband subscriptions this is still a relatively small number compared to the 4.6 billion mobile subscribers. In a recent survey done in the United States and Europe it was reported that out of 80% of respondents who have access to mobile internet only 32% make use of it. For the people who do use mobile internet only a third of them report being satisfied with the mobile internet experience [3].

1.3 Problem with Mobile Internet and Informational Retrieval

One of the main problems keeping users from using the mobile internet is the limited screen size and interaction methods on the mobile devices. Currently mobile search interfaces are designed similarly to desktop search engines, but for mobile devices the screen is much smaller and harder to see, with the buttons being closer together, which make them error prone [4]. This in turns restricts the number of results that are displayed on a single screen when searching. Often snippet, text associated with search results, are truncated and sometimes even removed to show more results [5]. Internet content is growing exponentially, and searching for useful content

is already a tedious task on a PC but even more challenging on a mobile phone. Searching for information is like finding a needle in a constantly growing haystack and navigating and looking through pages of results is very tedious, long and a costly experience for the users [4]. According to a study done on desktop searches users have an average query length of only 2.6 terms and scroll through 2.35 pages before they find what they are looking for [6]. For people doing the search on a computer this is an easy task and requires typing 2.6 terms on the keyboard and scrolling down the page and clicking the link to the next page. To do this search on a mobile device with basically the same search interface as the computer requires much more work because of the small screen and limited interaction method. Users are restricted by their text-input and have to press small buttons to enter the 2.6 search terms. Maneuvering from one page of small text to another is time consuming and requires clicking to the bottom of the page and pressing the link to the next page. Typical search engines return a ranked result list to the users and finding relevant information from the long list is difficult especially if they are entering roughly two terms per query [3]. Two term queries are short and have the potential of being vague and ambiguous therefore producing relevant results within the first few results is difficult. Ranked results fail to provide an overview of the different themes of an ambiguous query forcing users to look through pages of results in order to browse the different topics related to their query [3]. Limited interaction methods and the need for efficient vertical scrolling annoys the users and after many pages of results they often abandon the search without finding what they are looking for. A study done on mobile web search indicates that users typically have an average query length of 2.06 terms which is shorter than its desktop search counterpart. It also states that 36% of the time single term queries are used and the vocabulary size of the queries are quite limited when compared to desktop searches. In addition users are less willing to click on pages and less

likely to carry out multiple queries per session. The most shocking thing discovered was that only 8.5% of users who do mobile searching go beyond the first page [3][4]. Users are not finding what they are looking for in a timely fashion because of vague queries and the users lack of patience to go through the long list of results. This is very understandable given the setbacks of the mobile devices with their small screens, limited interaction method and high data transfer cost.

1.4 Current Solution for Mobile Internet and its Deficiencies

To solve this problem mobile operators have pre-classified content for users to browse through. The pre-classified content are listings that mobile operators believe users are interested in and users would browse and click on information they want to see. Some of the pre-classified categories are things like sports, movies, games, mail and all kinds of things mobile operators believe users are interested in. Information in pre-classified documents are structured and organized therefore allowing fast browsing. Users are not forced to look through pages of result, they look through pre-processed results that are filtered and organized to fit on small mobile screens. Some mobile operators even try to personalize listings based on users browsing habits so they can do less clicking to get to what they want [7]. Mobile phones even offer a feature that promotes browsing by including special purpose browsing keys to help users navigate. The current problem with a pre-classified listing is that many times the listing does not have what the users are looking for. There is no way mobile operators can pre-classify all domains and databases. Users are restricted to the pre-classified content the mobile operator provides to them so if it not there then the users have to do a lot of browsing. Even with this problem studies show that browsing is preferred over mobile searching when it comes to getting online content from

their mobile devices [8]. This just goes to show the inefficiency of searching on mobile device and how much work we have to get users from browsing to searching.

Studies show that people who have sophisticated handsets and PDA's are the ones that tend to use the search capabilities. These sophisticated handsets and PDA's have flip-out keyboards and stylus support so they are not restricted by their text-input but are still restricted by the small screen dilemma. People who engaged in a search session tended to have more mobile online activity. The search sessions are also found to be 3 times the duration of browsing sessions and 2.5 times the number of requests and lead to 4 times the number of kilobytes downloaded [8]. They explain this by saying that browsing is done faster because users are usually navigating through pages they had been to in the past so they are more confident when they are browsing. They say searching takes longer because people are usually trying to find something new and that will typically involve more time and effort for the searcher. This may be true but I believe people spend more time searching because it is more interesting and useful than browsing. Searching provides a richer online experience and it gives the users the freedom of finding information that is interesting to them and not restricted by what the mobile operators provide them [8]. Searching is the most logical and intuitive way of finding information but the limited screen sized and limited interaction methods prevent them from using it.

In the beginning stages of the internet browsing was very popular and for directory services like Excite and Yahoo it was the dominant form of information access for desktop computers [8]. As search technology got better more people started searching for information and

today it is the most dominant way of the looking for information on the traditional web. Like traditional web, I believe when search technology is effective and efficient people will transition over from browsing the mobile internet to searching. There is a need for improved search facilities, facilities that need to be carefully adapted to the features of the mobile internet [5]. Only then will we see a significant growth in mobile online activity. One day mobile information access will undoubtedly become as indispensable a method of information access as desktop computers [3]. I believe that mobile devices will become the primary platform for internet access in the future.

2.0 RELATED WORK

This section talks about different solutions for information retrieval for mobile devices. The first solution will be clustering web information to allow fast browsing. I will be describing two clustering algorithms, Lingo and Suffix Tree Clustering (STC). The second solution will be query expansion and relevance feedback to allow precise information retrieval.

2.1 General Clustering

The traditional list based result interface does not scale well to mobile devices search engines because of the amount of data that is out there. The popularity of the internet has caused the amount of information to grow at an enormous rate and because the information is heterogeneous, unstructured and dynamic it is difficult to obtain the needed information accurately and easily. Search results are often huge and many of the results may be irrelevant to the user's goal [9][10]. When search query are too general or vague the results returned can be useless since there is so much information out there. It is difficult for users to find what they are

interested in so they are forced to sift through a long list of off-topic documents [10]. For many years now clustering and semantic grouping has been a proposed solution to this problem. Search engine clustering organizes web page results into groups that reflect different components of the query topic [9]. Results are grouped together based on their shared properties, thus web pages that relate to the same topic are hopefully placed together in the same cluster [11]. Shared properties are derived from frequently occurring words or phrases from different web pages. When content from web pages is not available search engines often use snippets to group the pages. Since clustering is based on words and phrases they are used to label the clusters. The labels help provide the users with an overview of the result set so they don't have to look at every result even if they are the same [12]. Users normally don't browse through pages and pages of results, so by grouping them and summarizing them the users is able to browse thorough different themes and topics related to their query [9]. In traditional search engines the relationship between different web page results are not presented and left to the users [10]. Clustering attempts to utilize the relationship between different web pages automatically to help users quickly drill down into smaller more interesting focused results [12]. Clustering makes short cuts to sub topics from an ambiguous query and web pages can be accessed in logarithmic instead of linear time [4].

When users know what they want, clustering allows them to easily pinpoint what they are looking for by browsing through the list of labels. Clustering is also useful when the users do not know what they are looking for. For example, let us say the user's computer crashes and they get the message "A problem has been detected and Windows has been shut down to prevent damage to your computer". Our user is not sure what this is and because their machine is not working

they decide to look it up on their mobile device. They enter the message as their query and a list of labels is returned to them. They get topics like Blue screen of Death, Fix Windows, Help, Troubleshooting, Problem, Restart your Computer and much more. Figure 2 shows some of the results from searching 'A problem has been detected and windows has been shut down to prevent damage to your computer' using the Lingo clustering algorithm in Carrot2 [13].

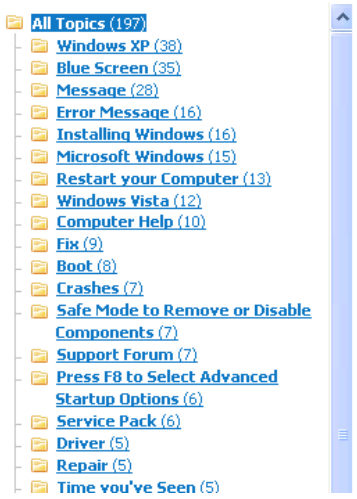


Figure 2. Lingo algorithm example from the Carrot2 website

Blue Screen of Death is one of the biggest clusters returned and if the user decides to explore it the user should determine that the error they got is the blue screen of death. Other labels like "Fix Windows", "Help and Restart Your Computer" should help the user figure out different ways to fix the problem. They also have labels like "Problem and Troubleshooting" which helps the user figure out why the error occurred. The labels were returned from 200 web pages and one can just imagine how much more time consuming it would be if the users had to manually sift through all those results to determine things like what is it, what caused it, and how to fix it. If the 200 pages were returned in a standard ranked list and the web pages that represent "Problem and Troubleshooting" are in the third page of results the user probably would have never figure out why the error occurred. Clustering turned the 200 results into 30 different labels, which are

easy to browse through and find “Problem and Troubleshooting”. The user had no idea what the error message was about when their computer shut down, but now they have the equivalent of looking through 200 results by looking at the 30 different labels. Clustering allowed the user to get a high level view of the whole query topic allowing the users to get a better topic understanding. In addition, clustering lets the users easily filter out labels and topics that are irrelevant to what they are looking for. This allows them to quickly browse through the topic and not spend time scrolling through the results.

Clustering search results on desktop computers is attracting a substantial amount of research and commercial interest but has yet to be deployed to major search engines. Major search engines like Yahoo and Google have shown interest in the technology and have currently adopted some form of implicit clustering to increase the number of different perspective in their first page of results [4]. With all the research and commercial interest in clustering the majority of people still use standard rank list search engines. I believe this is because computers do not have the limited screen size and interaction methods as mobile devices. Desktop computers have a keyboard, mouse and big screen so they can easily maneuver through the list of results. Clustering may be unused on desktop machine but is starting to be utilized for mobile devices [4][12]. With clustering the limited interaction and scrolling problems of the mobile phone becomes a non-factor because users are no longer looking through pages and pages of ambiguous results. They are browsing a few labels that represent the different topics of the query and the users can select an individual label and review the different web pages inside. Users are unwillingly to scroll through 200 results on their mobile phone, but they are more willing to look through the 30 different labels representing the 200 results. By organizing the results into

meaningful groups and categories it is easier for users to make sense of the results and decide what action to pursue [12]. They can decide what groups are relevant and irrelevant. By skipping the irrelevant groups' users do not download the data and this is especially useful for pay as go mobile internet subscribers, because by filtering out the topic they are not paying for results that they do not want to see. Clustering makes the users feel like they are browsing which we know the majority of mobile users are still doing [12]. Browsing is easier for mobile information retrieval because there is less scrolling and typing. Clustering may make mobile retrieval feel like they are browsing, but unlike traditional mobile browsing the users has the option of what categories they want to see. With traditional mobile browsing the mobile operator determines the pre-classified categories the users can see. The users submit a query and the different labels and categories related to the query are returned. This way the users can look up anything they want and are not restricted by what the mobile operator gives them.

2.2 Suffix Tree Clustering (STC)

A popular clustering algorithm is Suffix Tree Clustering (STC). STC is a linear time clustering algorithm that creates groups based on phrases shared between documents [14]. A phrase is an ordered sequence of one or more words, and they are used to summarize the grouped documents. The algorithm assumes that documents with the same topic will share common phrases. STC has many characteristics that make it popular for clustering search results. STC uses phases instead of words to do the clustering. This is a big advantage since phrases make use of proximity information between words [9]. Phrases are useful in constructing precise and accurate labels describing the clusters [15]. Many algorithms that label their clusters using complex and non-intuitive methods make it difficult for users to understand the labels. These

algorithms have a problem creating comprehensible descriptions and returns result that are not feasible and interpretable to humans.

STC is a Description-Aware algorithm meaning that is aware of the description problem and tries to carefully select meaningful phrases that are immediately recognizable by the users [4]. The words that are common in the different documents are kept in the same order when they are chosen as phrases so the users can interpret them as if they were reading it straight out of the original documents. When the phrases are meaningful and precise they are used to describe the output clusters accurately and sufficiently. By looking at a phrase the users should precisely know which documents are present or absent from the cluster. STC also allows overlapping of documents and clusters. Documents often have multiple topics so if we were to confine a document to a single topic we can be losing valuable information about the document and its other topics [15].

2.2.1 The Suffix Tree Clustering Algorithm

STC is a fast, incremental and linear algorithm which is ideal for online clustering [15]. Now that we know a little bit about the advantages of the STC algorithm I will discuss how it works. STC has three main steps. The first step is document cleaning and this step eliminates noise in the documents to prepare it for clustering. The second step, identifying base clusters, searches for all the sets of documents that share a common phrase. The last step is combining base clusters, which merges base clusters based on the percent of the documents they have in common.

2.2.1.1 Step 1 Document “Cleaning”

Before the algorithm can start clustering the different documents into groups it must first clean all the data and remove all unnecessary content from each document. The unnecessary content adds noise and does not help to extract phrases from the documents or cluster them. String of text representing each document is first tokenizes and non-word tokens such as numbers, HTML tags and punctuation are stripped [14]. The non-word token get in the way of clustering and do not say anything important about the documents content so they are removed. Also tokens that are stop words and web-related words that are frequent in webpage’s are filtered out [16]. Stop words are common uninformative english words such as “the”, “it” and “on”. Web-related words are things like “http” and “nbsp”. Than all text is converted to lowercase so they can be easily compared to each other during identifying base clusters step. Next stemming is applied to reduce certain words to their root form, like how “computer”, “computing”, “compute” and “computers” are all transformed to the root word “compute”. A light stemming algorithm which was adapted from the porter stemmer is used to reduce words to their root form. It is considered a light stemming algorithm since only the first component of the porter stemmer algorithm is used to decrease the processing time. The first components of the stemmer algorithm consist of reducing plurals to singulars. Along with reducing processing time the light algorithm allows phrases to be more interpretable by humans [15]. The last step of document, cleaning, is marking sentence boundaries in all of the documents. Sentence boundaries are identified using standard sentences terminators such as period, comma, question mark and the location of surrounding HTML tags like “<p>
<td>” [9]. Sentence boundaries are used to break down documents into phrases. Texts within parenthesis or quotes are considered independent phrases and are placed at the end. Phrases can not cross phrase boundaries for two reasons. The

first reason is because phrase boundaries usually mark a topical shift and interesting phrases usually do not cross these boundaries. The second reason is because it will reduce the cost of building a suffix tree.

2.2.1.2 Step 2 - Identifying Base Clusters

In the second stage base clusters are identified using a generalized suffix tree. For a phrase W_0, W_1, W_2, W_3 … W_i, a suffix is defined as a subsequence W_j, W_{j+1}, W_{j+2}, W_{j+3} … W_i, where $j \leq i$. Given the phrase "cat ate mouse too" the different suffixes are "cat ate mouse too", "ate mouse too", "mouse too" and "too". A generalized suffix tree is a data structure that admits efficient string matching and querying [15]. It does this by creating an inverted index that represents all suffixes from a phrase. The tree structure is used to find string matches from a phrase and can also be used to find suffixes that match the string you are looking for.

Suffix Tree: A suffix tree T for an m-word phrase P is a rooted directed tree with exactly m leaves numbered 1 to m. Each internal node (non leaf node), other than the root, has at least two children and each edge (line between two nodes) is labeled with a nonempty sub-string of words of the phrase P. No two edges coming out of a node can have the same word as the beginning of their edge labels. One of the main features of the suffix tree is that for any leaf l, the sequence of all string edge label from the root to leaf l spells out the suffix of P that starts at position l and spells out to the last word m which is equal to S[l…m] [15]. A terminating character which is not used in the phrase P is added to the last suffix m so that if the last suffix matches a prefix to another suffix the prefix does not become a leaf which contradicts the definition. An example of the suffix tree "I know you know I know" is given in Figure 3 [15]. The example has six leaves

marked as rectangles and numbered from 1 to 6. The internal nodes are circles and the labels are next to the edge. As you can see the terminating character is '$' and it allows leaf 6 to be created. Without the '$' the internal node would become the last suffix and would contradict the definition.

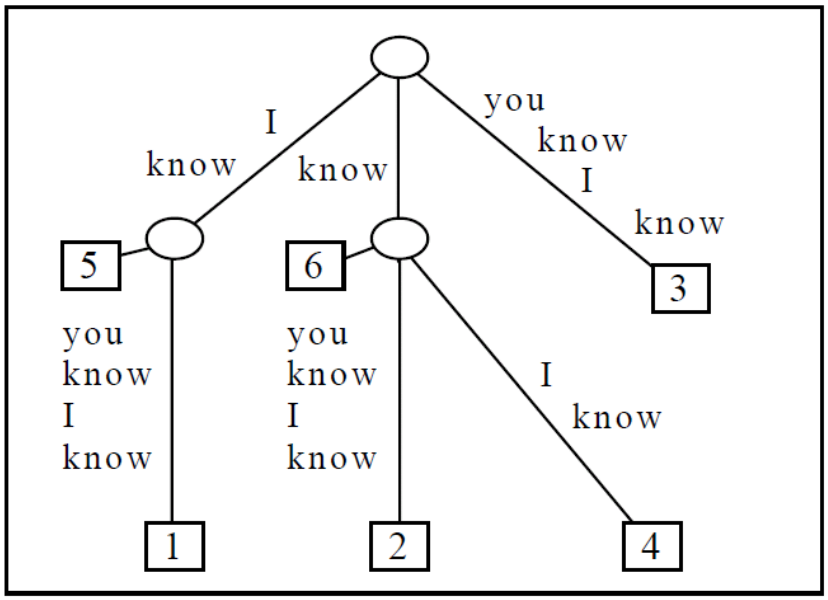


Figure 3. An example of a suffix tree Source: [15]

To build a suffix tree for more than one phrase we need a generalized suffix tree. A generalized suffix tree has every suffix for all phrases we are comparing.

Generalized Suffix Tree: A generalized suffix tree T for phrase P of n phrases P_n, each prase has a length of m_n. The generalized suffix is a rooted directed tree with exactly $\sum m_n$ leaves marked by a two number tuple (d,l) where d ranges from 1 to n representing the different phrases and l ranges from 1 to m_d which represents the different words in each phrase. All internal nodes, other than the root, have at least two children and each edge is labeled with a nonempty sub-string of words in P_n. No two edges coming out of a node can have the same word as the beginning of their edge labels. For any particular leaf (i,j), the sequence of all string edge labels from the root

to leaf (i.j) spells out the suffix of P_i that starts at position j, and spells out to the last word $P_i m_i$ which is equal to Pi[j..mi] [15]. Each phrase P in P_n has a unique terminating character that is appended to the last suffix $P_i m_i$. Figure 4 is a generalized suffix tree for the phrases "cat ate cheese", "mouse at cheese too" and "cat ate mouse too" [15]. The internal nodes are the circles and they are labeled from a through f. Rectangles represent the leaves and the first number in each rectangle represent the phrase in which the suffix originated. The second number represents the position in the phrase in which the suffix starts.

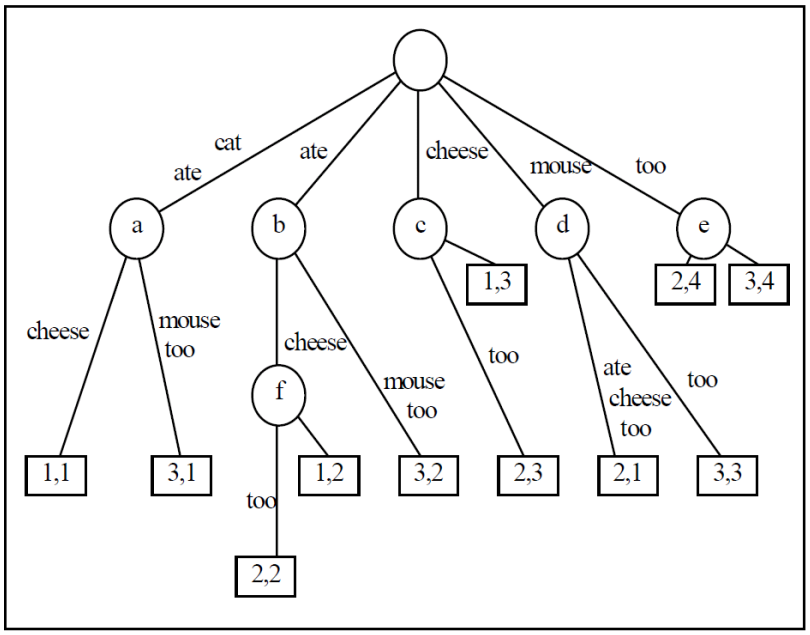


Figure 4. An example of a generalized suffix tree Source: [15]

The straightforward way to build a suffix tree for a phrase S of length m takes $O(m^2)$ time [15]. The algorithm starts with the first suffixes which is the whole phrase S[1… m] and it creates a leaf that attaches to the root using the phrase as the edge label. Than it successively goes to the next suffix S[i …m] which starts with the second word so i=2 is used and it tries to find the longest path from the root whose label matches a prefix of any previous suffixes. If the S[i… m] suffix finds a match with a prefix than an internal node is created to the root with the

prefix being the edge label and the leaves are created depending on what is left after taking out the prefix from the suffixes. If the S[i… m] suffix does not find a match it is attached to the root and a new leaf is created with the edge being the S[i… m] suffix. This process is done until it gets to the last suffix S[m] and by then it should have a complete suffix tree. The algorithm time complexity is $O(m^2)$ since every suffix of the phrase is comparing itself to previous suffix in the tree.

$O(m^2)$ is rather slow, but luckily several linear time algorithms exist for constructing suffix trees. These algorithms also exhibit a time dependency on m the number of words in a phrase. The precise time bound for the algorithms is O(m*min(log|L|,log m)) and |L| is the size of the language we are using. There are three algorithms that meet this time bound and they are the Weiner, McCreight and Ukkonen algorithm. I will not give details about these algorithms, but I will simply mention that the Weiner's algorithm uses larger space requirements than the two and the Ukkonen's algorithm is simpler than McCreight's algorithm. Ukkonen's algorithm has useful online characteristics because it has an incremental approach that can build suffix trees while phrases are brought to memory from disk or network [14][15].

To cluster documents into groups the STC algorithm will use a generalized suffix tree. After the document "cleaning" we will have multiple documents that will have phrases that are separated by phrase boundaries. The STC algorithm will then use one of the above methods to build a generalized suffix tree. The documents and all of it phrases will be in the tree and the internal nodes will represent groups of documents and phrases that is common to it children. Each internal node V will have a label V_p the common phrase. Leaves of the sub-tree of V correspond to phrases that have suffixes that have V_p in it. The leaves of V_p are the group of

documents that will contain the base cluster and the concatenation of all edge label from root to V will be the base cluster label. Leaves are marked with a phrase identifier, and it will contain which document it came from and where the phrase is found in the document [15]. Table 1 gives an example of the different phrase clusters from Figure 3 [15]. The internal nodes that were labeled from a to f represent the phase cluster.

Node	*Phrase*	*Documents*
a	cat ate	1,3
b	ate	1,2,3
c	cheese	1,2
d	mouse	2,3
e	too	2,3
f	ate cheese	1,2

Table 1. The six phase clusters from Figure 4 Source: [15]

Each base cluster or internal node is guaranteed to be a maximized base cluster. A maximized base cluster is a base cluster whose phrase cannot be extended by any word without reducing the group of document. Because all suffix of phrases are compared with every suffixes from other phrases to make base clusters we know we are getting maximized base clusters. After the different documents are in the generalized suffix tree the maximized base clusters are scored based on the number of documents and phrases it has. The score is suppose to represent how useful the maximized base cluster is and will be used to filter out bad base clusters. The equation to calculate the score of the maximal phrase cluster m with phrase m_p is:

$$S(m) = |m| * f(|m_p|) * \sum tfidf(w_i)$$

$|m|$ is the number of documents in phase cluster m, $|m_p|$ is the number of words in m_p and w_i is the words in m_p. The function f penalizes single word phrases and rewards phrases that have longer

phrases. Term frequency inverse document frequency (tfidf) is a technique that is used for assigning weights to individual words. The more frequent the word is in the document then the more important the word is to the base cluster. Tfidf also takes into account that if a term occurs in almost all the document than the term is probably not that important and has a smaller weight. When the tree is traversed and all the scores are calculated the maximal base clusters is filtered based on their score. If the maximal base cluster score exceeds a predefined Minimal Base Cluster Score the cluster would be used and if it does not it would be removed [17].

2.2.1.3 Step 3 - Combining Base Clusters

When we were identifying the base clusters we were finding documents that share the same phrase. However some documents may share more than one phrase since STC allows document overlapping. If this occurs distinct base clusters can overlap and sometimes clusters are identical. To avoid having nearly identical base clusters the third step merges base clusters with high overlap in their documents sets [14][15]. To decide if two base clusters should be merged an equation is used to measure the similarity of the document set overlap. Given two base clusters m_i and m_j, with $|m_i|$ and $|m_j|$ being the number of document in each cluster and $| m_i \cap m_j |$ representing the number of documents common to both base clusters, the equation is:

Merged m_i and m_j if $|m_i \cap m_j|/ |m_j| > \alpha$ and $|m_i \cap m_j|/ |m_i| > \alpha$ else do not merge.

α also known as Merge threshold is a constant between 0 and 1. The values typically used is α=.6. After phrase clusters are merged the union of the documents we used as the cluster and the label would be combined [9]. For labels that are subset of one another the labels would be

merged [16]. Figure 4 shows the base cluster from table 1 merged with the merge threshold being .7 [15].

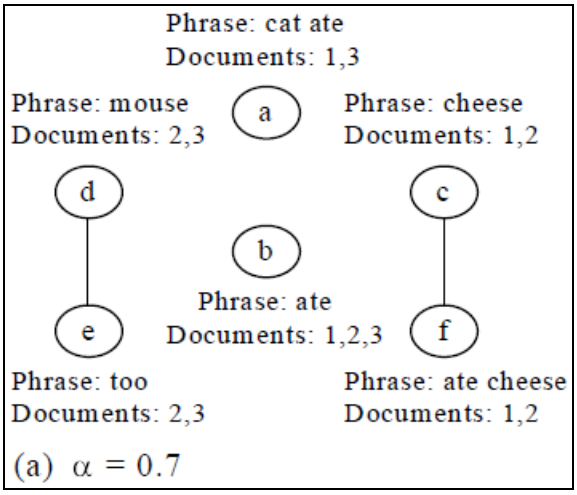


Figure 5. Merged base clusters based on the six base clusters from Table 1 Source: [15]

Base clusters that remerged have a line connecting them. The base cluster are sorted by their scores in descending order and processed one at a time. Each base cluster is compared to the rest of the base clusters and when there is a merge a link will connect the two base clusters. When two base clusters are merged the score of the two base clusters are added together. After the merge process is done then all base clusters are sorted by their scores and the base cluster become the different cluster of the documents.

2.2.2 STC Evaluation

Oren Zamir and Oren Etzioni evaluated the STC algorithm and compared it against other cluster algorithms [14]. To do this they had 10 queries and they clustered 200 web pages using the different algorithms. To calculate the precision of each clustering algorithm they manually went to each cluster and assigned a relevance judgment for every webpage in that cluster. They

went through every web page and used its content to determine if the page was relevant to the cluster. Figure 6 shows the comparison between the various clustering algorithm and their average precision [14].

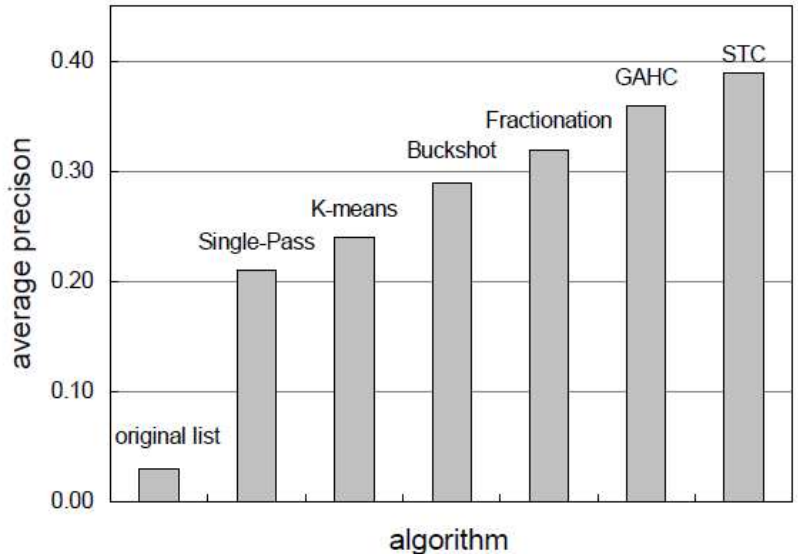


Figure 6: The average precision for the different clustering algorithms Source: [14]

The figure shows that STC algorithm scored the highest in the experiment. They believe the algorithm had good results, because STC uses phrases to identify its clusters and that it naturally allows overlapping [14].

Oren Zamir and Oren Etzioni also showed how clustering snippets affect the performance of different algorithms. They did the same test as before but instead of using webpage content they use the webpage snippet to do the clustering. Figure 7 shows the average precision for the clustering algorithms using snippets compared to the average precision on the original webpage collections [14].

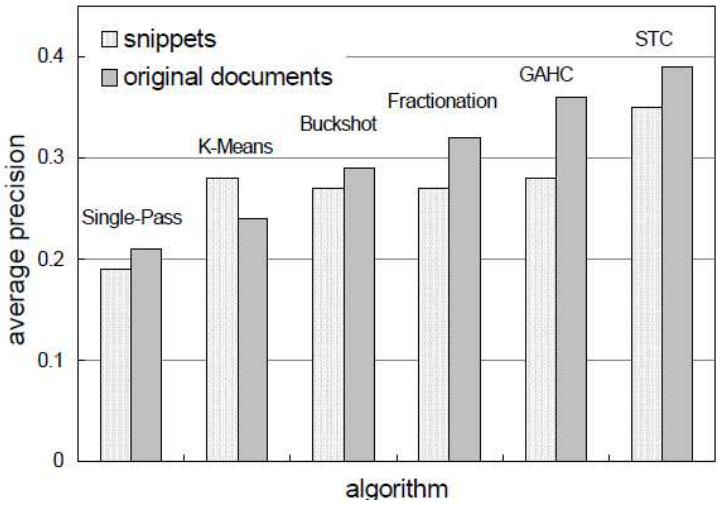


Figure 7. Average precision for clustering algorithms with and without snippets Source: [14]

The average precision for clustering snippet data has a slight decrease compared to using the webpage content. Snippets are search engines attempt of extracting meaningful phrases from the original web pages. The fact that snippets only use meaningful phrases eliminates some of the noise present in web pages that might cause misclassification and this is why they believe using snippets barely decrease the average precision.[14]. Clustering snippets uses less memory and time than clustering web pages. It was also notice that when using web pages to do clustering there were more clusters and each cluster had more web pages. This is due to the fact that web pages have more content than snippets so more matches are found [16]. Clustering snippets returned by search engines is a reasonable and speedy alternative to using web page content [14].

A great advantage of the STC algorithm is that it uses phrases to discover and describe its resulting groups. This advantage makes STC a great algorithm but it also causes problems. The order of words in phrases is important in STC and this is a problem for documents that relate to a phrase but have the words in a different order. Another problem with using phrases to do

clustering is that if a document does not have any phrases but is still is relevant to a cluster it will not be included. STC clusters document based on phrases and if the document doesn't have the phrases it will not be included in the cluster. Overlapping is another advantage of STC but it can cause problems when clusters become too big and users are forced to sift through the results. Since documents can be in more than one cluster the different cluster can become large and have too many documents in them.

A big disadvantage of the STC algorithm is that it uses a suffix tree. A suffix tree is an efficient way for identifying common phrases between documents, but it has a large memory requirement and poor locality characteristics. Suffix tree also tends to remove high quality phrases leaving less informative ones. The different thresholds and parameters of a suffix tree play a significant role in the cluster formation and tuning them is extremely difficult [17].

2.3 Lingo

Lingo is another clustering algorithm that uses the description comes first motto [14]. Although the algorithm does take into account the quality of its document allocation the majority of its computation is finding meaningful phrases to label its clusters. Lingo is a description-centric algorithm, and unlike STC it dedicates the first few steps to finding good labels and descriptions. The creators of Lingo believed that if a cluster cannot be described than it has no value to the users and must be dropped. Careful selection of label candidates is important and the topics must be unique and cover the different topics in the search results. If labels are unambiguous we have the same problem as rank result list and we must start over and try again. The aim of clustering is to cut down the number of result users need to sift through. Clustering allows us to step back and look at the bigger picture. We do not care about the individual

document but rather about the group of documents and the underlying semantic label description that represents them. To the users the labels must be understandable so that when we generate labels we will find meaningful, concise and accurate ones that give an overview of the topic.

The majorities of clustering algorithm are Data-Centric and focus on cluster content discovery and then try to find a fitting label based on the content of the groupings [10]. Very often similarity among documents do not correspond to human readable labels therefore the labels that represent these clusters are useless and the whole cluster becomes meaningless to the users. Figure 8 shows two search engines that use the Data-Centric Scatter/Gather algorithm. The left one comes from Lin and Pantel and the right one comes from Hearst and Pedersen [4].

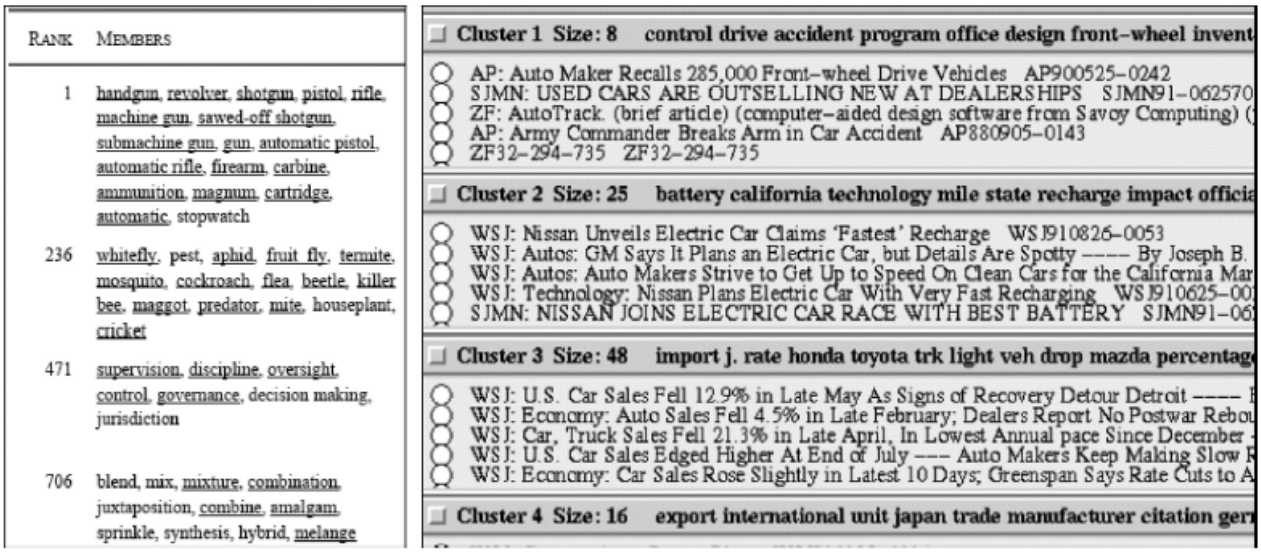


Figure 8. Clustered web results using Data-Centric Scatter/Gather algorithm Source: [4]

As you can see the description label for each cluster are insufficient and it becomes impossible to comprehend the meaning of the web pages in each cluster. These algorithms first clustered the web pages and than found frequent key words to describe their clusters. Because these key words do not take in account word proximity and phrases they are hard to understand. They might know

how the certain web pages are similar, but they do not know how to describe the actual relationships [18].

To avoid the labeling problem Lingo reverses the Data-Centric process and finds good cluster labels first and then assigns documents to each label [18]. They do this by combining several existing methods to put extra emphasis on meaningful cluster labels. They extract frequent phrases from the input documents, hoping that they are the most informative human readable description for the cluster labels [10]. Only after Lingo finds human-perceivable cluster labels will they start to find similarity between documents and labels for their clusters. The quality of the clustering is a result of ensuring both description and content of the resulting groups are understandable to humans [18]. We already establish that the Lingo description label process takes this into consideration, but we have not explored how Lingo groups its content to make it understandable by humans. Lingo might use frequent phrases to label the different cluster, but they do not use these phrases to find the different cluster groups. To find different cluster groups Lingo uses abstract concepts discovered using singular value decomposition (SVD) [10]. SVD is used to get abstract concept out of input document to create different cluster groups rather than finding frequent occurring words to make clusters. In most documents it is very common for people to use synonyms and pronouns to avoid word repetitions. This makes it difficult for clustering algorithm like STC, which generate label and clusters solely on frequent occurring terms. When using synonyms and pronouns to replace certain words, this usually leads to many phrases that are similar but differ by on or two terms [10]. Lingo overcomes this problem by using SVD to identify abstract concepts. These abstract concepts are phrases and words that do not relate to each other and represents the different topics obtained from the

different snippets. These abstract concepts may have phrases and words that clearly represent the topic but are usually hard for humans to interpret as labels. That is why the phrases and words from the different abstract concepts are not used as labels. Instead Lingo compares the abstract concepts to the frequently occurring phrases so they can use them as labels.

2.3.1 The Lingo Algorithm

Now that we know a little about the Lingo algorithm and some of its advantages lets look at how the algorithm works. Like most clustering algorithm the first step is preprocessing and then frequent phrase extraction, cluster label induction and last cluster content allocation.

2.3.1.1 Preprocessing

When Lingo is used for web result clustering we cluster the data based on web page snippets instead of the actual webpage. The main goal of preprocessing is to remove all characters and terms that can affect the quality of group labels. Lingo is pretty efficient when dealing with noisy data but because snippets are so small without sufficient preprocessing SVD would return abstract concepts of meaningless frequent terms [19]. Four steps are performed to prune snippets so that it can produce meaningful terms: text filtering, document's language identification, stemming and stop word marking.

Text Filtering:

Text filtering is the removal of all terms that are useless and bring noise to the description of the cluster. HTML tags like <body>, &amp and <b> are removed from the snippet because they have no value to the cluster labels and are only there for presentation purposes. Also non-

lettered characters such as #, $ and % are removed but white spaces and sentence markers (e.g '.' '?' '!') are kept to identify sentence boundaries. Note that stop words are not removed because they will be used for language identification [19].

Language Identification:

In order to do stemming and stop word marking for the input snippet Lingo must determine the language. Different languages have different stemming algorithm and stop words so they must first determine what language the snippet is before they can proceed. Lingo automatically determines the language for each one of its snippets so clustering can be done on multiple results from different languages. This is very convenient for users because they don't have to select what language they are using, and they can view results from different languages. Lingo uses the 'small word technique' for its language identification algorithm. The way this algorithm works is it gets the stop list for all the languages it supports and it goes through the document looking for words that are in the different stop lists. The stop word with the highest number of occurrence is selected and the language that the stop word came from will be identified [19].

Stemming:

After determining what language is used for each snippet an appropriate stemmer algorithm will then be used for that language to remove suffixes and prefixes. Doing this guarantees that different forms of terms are treated as a single term and this should decrease label ambiguity. Because bare stems are difficult for humans to understand the original term is stored and presented to the users [19].

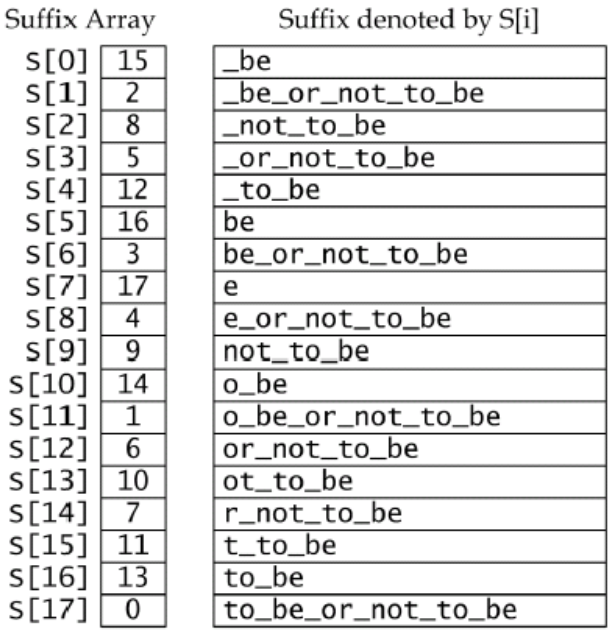
Stop Words Marking:

Language recognition allows Lingo to determine what stop list to use to mark the stop words in the input snippet. Even though stop words have no descriptive value they can help users understand and disambiguate long phrases. For example compare “Chamber Commerce” to “Chamber of Commerce”, the first phrase may not make sense to the users if it was used as a label. For this purpose Lingo only marks the stop words and does not completely remove them. By marking the stop words Lingo can prevent stop words from being used as cluster labels and ignore them when they start or end a phrase when clustering. If stop words are not marked they could have meaningless cluster labels such as “the”, “it” and “on” [19].

2.3.1.2 Frequent Phrase Extraction

The next step in the Lingo algorithm is frequent phrase extraction. The process is done so that they can discover phrases and single terms that can potentially explain the verbal meaning behind the SVD derived abstract concepts [20]. In order to pick phrases and single terms that Lingo will use as its cluster labels they use a suffix array.

A suffix array is an alphabetically ordered array of all suffixes of a string. Each suffix of a string is represented by a single integer that is equal to the position of where the suffix starts. Because the array only stores the position of where the suffix starts in the string, it is not necessary to store the text of the particular suffix. To get the actual suffix all you need to do is get the substring of the original string, starting at the integer value in the suffix array to the end of the string. Figure 9 has an example of a suffix array for the string “To_be_or_not_to_be” [19].



Suffix Array		Suffix denoted by S[i]
S[0]	15	_be
S[1]	2	_be_or_not_to_be
S[2]	8	_not_to_be
S[3]	5	_or_not_to_be
S[4]	12	_to_be
S[5]	16	be
S[6]	3	be_or_not_to_be
S[7]	17	e
S[8]	4	e_or_not_to_be
S[9]	9	not_to_be
S[10]	14	o_be
S[11]	1	o_be_or_not_to_be
S[12]	6	or_not_to_be
S[13]	10	ot_to_be
S[14]	7	r_not_to_be
S[15]	11	t_to_be
S[16]	13	to_be
S[17]	0	to_be_or_not_to_be

Figure 9. Example of a suffix array for the string “to_be_or_not_to_be” Source: [19]

The figure shows the suffix for each element in the suffix array, for clarity the suffix is shown next to each array element, but we must remember that only the suffix start position is stored. We can also see that the array is sorted by the suffixes. Because the text being sorted are all suffixes of the same string there are algorithms that exploit this and can sort suffix arrays in O(NlogN) time complexity. Suffix arrays can be used to search for substrings in a string allowing a wider choice of text searching operations. Searching things like “Is X a substring of T?” can be answered in O(P+lognN) time, where P is the length of the substring X and N is the length of T [19]. Suffix array have longer construction time than the alternative suffix tree but have less space overhead. Space overhead used by suffix trees varies from 120% to 240% the size of the original text. Space overhead for suffix arrays is only 40% the size of the original text [19].

Suffix arrays are used to facilitate the process of frequent phase extraction. To be considered as a candidate to become a cluster label a phrase or single term must:

1. Occur in the snippet at least a specific amount of time depending on the term frequency threshold. It is a widely accepted assumption that frequently occurring terms in snippets have strong descriptive power. Also, by omitting infrequent words they are doing less computation and saving time [19].
2. Not cross sentence boundaries. Phrases usually do not extend beyond one sentence so sentence boundaries where marked in the preprocessing phase to prevent this.
3. Not begin or end with a stop word. Phrases that start with or end with stop words are not readable cluster labels and therefore omitted. Stop words that appear in the middle of the phrase are important to cluster labels and should be kept [19].
4. Be a complete phrase. A complete phrase is a phrase that cannot be extended by adding proceeding (left complete) or trailing words (right complete) without making the phrase different form the rest. Complete phrases are better description for cluster than partial phrases [19]. For example, President George W. Bush is a better than President George.

To find phrases that match these rules a suffix array is loaded up with the input snippet that are preprocessed. The phrase discovery algorithm works in two steps. In the first step left and right complete phrases are discovered by linearly scanning the suffix array in search of frequent prefixes, because the suffix arrays are sorted prefixes are grouped together and easy to find. Each time a prefix, is identified information about its position and frequency is stored. To get the left and right complete phrases they scan through the suffix array twice. In the second step the left and right complete phrases are combined into a set of complete phrases [19]. After they get the set of complete phrase and terms they only keep the ones that exceed the term frequency threshold. Suffix arrays are used because they are convenient in implementation and very

efficient [19]. Lingo can identify all frequent phrases in O(N) time where N is the length of all input snippets [20].

2.3.1.3 Cluster Label Induction

After obtaining the frequent phrases and single terms that exceed the frequency threshold they can be used for cluster label induction. There are three steps to the cluster label induction and they are: term document matrix building, abstract concept discovery, phrase matching and label pruning.

Term document matrix building:

The term document matrix A, also known as the t x d matrix is built based on the collection of input snippets after the preprocessing phase. In the t x d matrix, t stands for the unique terms and d stands for the number of documents, or in our case snippets. In the matrix the column vectors represent each snippet and the row vectors denote the terms used to represent the documents. Lingo uses single terms to construct the A matrix instead of phrases because it feels that it is more natural to use sets of single words to represent abstract concepts. Additionally single terms allow finer granularity of description. The element a_{ij} is a numerical representation of the relationship between term i and document j. There are many ways of calculating a_{ij}, and the one Lingo uses is the tfidf scheme that we saw in the STC algorithm. Again tfidf is a technique that is used for assigning weights to individual words. In Lingo words that occur in the snippet titles are scaled by a constant factor s=2.5. In figure 10 we have an example of a t x d matrix with d=7 snippets, t=5 terms and p=2 phrases that was obtained by using the frequent phrase extraction with the suffix arrays. The matrix is weighted using the tfidf

weight scheme and is normalized [19].

The t=5 **terms**:
T1: Information
T2: Singular
T3: Value
T4: Computations
T5: Retrieval

The p=2 **phrases**:
P1: Singular Value
P2: Information Retrieval

The d=7 **documents**:
D1: Large Scale Singular Value Computations
D2: Software Library for the Sparse Singular Value Decomposition
D3: Introduction to Modern Information Retrieval
D4: Using Linear Algebra for Intelligent Information Retrieval
D5: Matrix Computations
D6: Singular Value Analysis of Cryptograms
D7: Automatic Information Organization

The 5x7 **term-document** matrix after column length normalization (tf-idf weighting):

$$A = \begin{pmatrix} 0.00 & 0.00 & 0.56 & 0.56 & 0.00 & 0.00 & 1.00 \\ 0.49 & 0.71 & 0.00 & 0.00 & 0.00 & 0.71 & 0.00 \\ 0.49 & 0.71 & 0.00 & 0.00 & 0.00 & 0.71 & 0.00 \\ 0.72 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.83 & 0.83 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

Figure 10. Example of a t x d matrix Source: [19]

The term document matrix can also be used for information retrieval. To do this we must take a users query and represent it by a vector in the column space of the term document matrix. So the query becomes a pseudo document that is built from the query terms. To do the query matching we must compute the distance of all documents to the query vector. The most common measure calculates a cosine between two vectors using vector dot product formula. Figure 11 shows information retrieval for the query ‘Singular Value’. As you can see document 1, 2 and 6 returns as matching search result. The other documents do not match and are ignored [19].

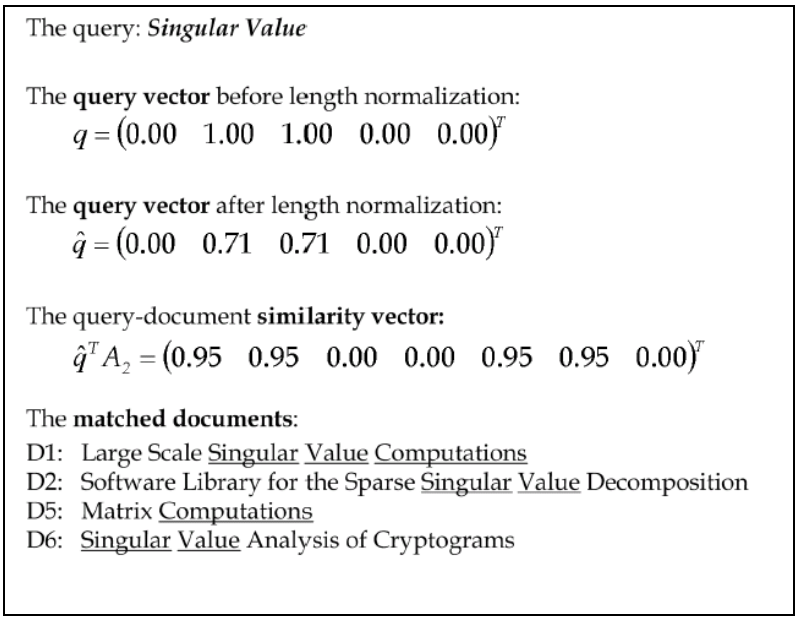


Figure 11. Information retrieval for the query 'Singular Value' Source: [19]

Abstract concept discovery

To get the abstract concepts from the matrix A they use singular value decomposition (SVD). SVD is a technique of extraction which attempts to reduce the rank of a term frequency matrix to get rid of noisy and synonymous words and exploit the underlying latent structure of concepts in the snippets. SVD is an algebraic method of matrix decomposition that is used for finding the orthogonal basis of the original term document matrix. The basis consist of the orthogonal vectors that hypothetically represent the different topics presented in the original term document matrix. To do this SVD breaks the t x d matrix A into three matrices U, $\sum$ and V such that A= U $\sum$ V^T. U is a t x t orthogonal matrix whose column vectors are called the left singular vectors of A. V is a d x d orthogonal matrix whose columns vectors are called the right singular vectors of A. Finally $\sum$ is a t x d diagonal matrix having singular values of A ordered decreasing along its diagonal [19]. Figure 12 shows an example of the three decomposition matrixes U, $\sum$ and V obtain from the term document matrix from Figure 10 [19].

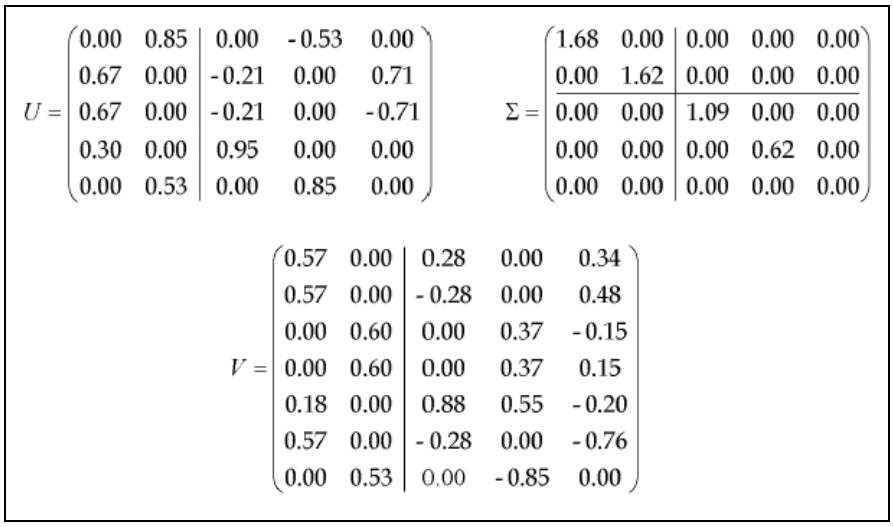


Figure 12. Three decomposition matrixes U, ∑ and V from Figure 10 Source: [19]

The rank rA of matrix A is the number of non-zero singular values along its diagonal. The first rA columns of the U form an orthogonal basis for the column space A and represent the different abstract concepts from the original term document matrix. We should note that only the first k vectors of matrix U should be used to determine the different abstract concepts. In Lingo the Candidate Label Threshold q, the rank rA of matrix A and ∑'s singular values are put into a formula to determine the value of k. The bigger the value of q the more abstract concepts are derived from U. Figure 13 shows the U and ∑ matrices obtained from SVD decomposition of the term document matrix A from Figure 10.

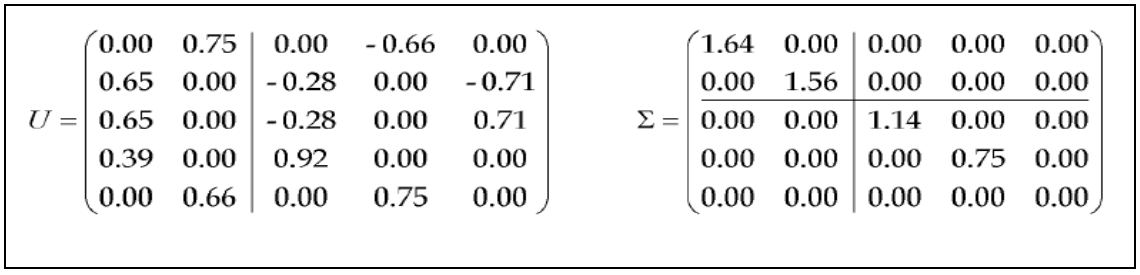


Figure 13. U and ∑ matrices obtained from SVD decomposition of Figure 10 Source: [19]

Assuming we use q=.9 the value of k =2 which means we have two abstract concepts that we will take into consideration. Column vectors of U are pair wise orthogonal which results in rich

diversity among the different abstract concept found because they cannot share the same terms. Basis vector also are expressed in the A matrix term space so the different rows are terms that represents the different abstract concepts. With k=2 the two abstract concept obtained by the matrix U are ‘Singular Value Computations’ and ‘Informational Retrieval’. As you can see ‘Singular Value Computations’ is a good abstract concept that represents a subset of the different snippets, but it is not a good label. It not a good label because the subset of snippets that the abstract concept represent does not have the word ‘Computations’ in it. ‘Computations’ is a term that SVD decided was important but not common among all the subset of snippets. Also the abstract concept are a collection of terms that are similar to the documents but the terms are not in any particular order and can not be used as labels. To find an appropriate label to this abstract concept Lingo must do phrase matching [19].

Phrase matching:

Frequent phrases and single words obtained from the frequent phrase extraction can be used to build a P matrix. The P matrix size is $t \times (p+t)$ where t is the number of frequent terms and p is the number of frequent phrases. Matrix P is built by treating the phrases and single terms as pseudo-documents and using the tfidf weighting scheme with length normalizing. Figure 14 show the P matrix for the two phrases “Singular Value” and “Information Retrieval” along with all the single frequent terms obtained from the frequent phrase extraction [19].

$$P = \begin{pmatrix} 0.00 & 0.56 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.71 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.71 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.83 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix}$$

Figure 14. The P matrix for the phrases and terms Source: [19]

They consider single term at this stage because they believe that frequent single terms describe abstract concepts better than phrases. Since both abstract concepts U and frequent phrases P are expressed in the same matrix A term space they determine how close a phrase or single term is to an abstract concept by computing the cosine vector dot product. The cosine dot product will return a distance measure that will compare and calculate which phrases or single term will be used as a verbal representation of the different abstract concepts. Having the P matrix and the ith column vector of the U matrix, a vector m_i of cosines of the angles between the ith abstract concept vector and the phrase vectors can be calculated as: $m_i=U_i^TP$. The column phrase that corresponds to the maximum component of the m_i vector will become the human readable description for the ith abstract concept. The value of the cosine will become the score of the cluster label candidate. Figure 15 present the M matrix for the label candidates e1 “singular value” and e2 “information retrieval” along with their scores. Because we choose the maximum component only one phrase or single term is used to describe an abstract concept [19].

$$M = U_k^T P = \begin{pmatrix} \mathbf{0.92} & 0.00 & 0.00 & 0.65 & 0.65 & 0.39 & 0.00 \\ 0.00 & \mathbf{0.97} & 0.75 & 0.00 & 0.00 & 0.00 & 0.66 \end{pmatrix}$$

The e=2 **candidate cluster labels**:
E1: Singular Value (score: 0.92)
E2: Information Retrieval (score: 0.97)

Figure 15. The M matrix for the label candidates Source: [19]

Candidate label pruning

Lingo does not want labels that are similar so they prune overlapping label descriptions. To do this they take V to be a vector of cluster label candidates and their score. Lingo then creates another term document matrix Z where the cluster labels candidates serves as documents. From there Lingo does column length normalization and calculates Z^TZ which returns a matrix of similarities between cluster labels. For each row they pick columns that exceed the label similarity threshold and keep only the cluster label candidate with the max score [19]. Our label description candidates ‘Singular Value’ and ‘Informational Retrieval’ are clearly different from each other. Because of this we will skip this phase in our example and move to the cluster content discovery phase.

2.3.1.4 Cluster Content Allocation

After Lingo gets the different cluster labels that represent the different abstract concepts it needs to assign the different snippets to the labels. To do this they use matrix Q which has the cluster labels presented as column vectors. They then take $C = Q^TA$ where A is the original term document matrix for the snippets. So the result of all the element c_{ij} of the C matrix indicates the similarities between the jth documents to the ith cluster label. A snippet is only added to the cluster label if it exceeds the snippet assignment threshold. Snippets not assigned to cluster labels will be defaulted to a cluster labeled ‘Other topics’. Figure 16 has matrix C and the different clusters of snippets and their labels. As you can see ‘Matrix Computation’ does not belong to a cluster label so it is default to ‘Other topics’ [19].

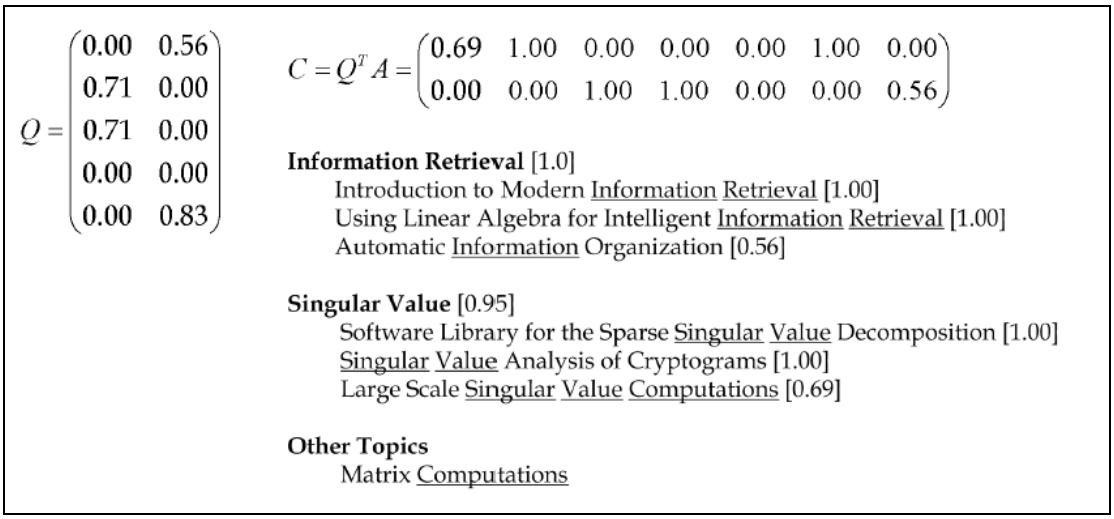


Figure 16. Matrix C shows the different clusters and their labels Source: [19]

Now we need to scores the different clusters and sort the results. The formulas to score a cluster is Cluster Score = Label Score * Number of Snippets. In Figure 13 the score of each cluster is right next to the cluster label and in our example they are normalized so that the highest score is 1.

2.3.2 Lingo problems

Lingo is a great description-centric algorithm but there are disadvantages. Lingo uses SVD decomposition to find unique abstract concepts and it uses suffix array to find frequent occurring phrases to describe them. Suffix arrays and SVD decomposition are computationally demanding and the whole process is time consuming. Another problem with Lingo is it has many thresholds and variables that greatly impact the labels they choose and the content that are assigned to them [19]. These variables may change depending on the users doing the searching and the type of searching they are doing.

2.4 Query Expansion

Typical queries usually return a few relevant web pages but occasionally miss many important ones that are potentially important to the users. This is due to the fact that there are a lot of relevant web pages out there and many of these pages are indexed by terms that do not match users' queries. For typical web searches the average query size is 2.6 terms [6]. This is generally too vague and can return pages that are not relevant to the users. Terms that may relate to relevant web pages may not be indexed by them and therefore are excluded from the results. The goal of query expansion is to improve retrieval effectiveness, it does this by finding terms that the relevant web pages are indexed by and adding them to the original query [22][23]. By adding additional search terms we get a more specific query that is less ambiguous and will return web pages that reflects the underlying information needed. In order to get relevant web pages the query and the web pages must have term overlap. Query expansion increases the chances of term overlap and tries to prevent term mismatches.

The goal of query expansion is to have a high precision and recall. Figure 17 shows how to calculate precision and recall for a given query.

$$\text{Precision} = \frac{(\text{number of relevant web pages retrieved from query})}{(\text{total number of web pages retrieved from query})}$$

$$\text{Recall} = \frac{(\text{number of relevant web pages retrieved from query})}{(\text{total number of existing relevant web pages})}$$

Figure 17. How to calculate precision and recall

Users want high precision because they do not want to sift through irrelevant web pages trying to find what they want. They want high recall because they want to be able to find relevant web pages without having to resubmit new queries. Adding terms that are relevant to web pages that users are trying to find will increase recall since they are retrieving web pages that would have

been missed. The terms will also increase precision since they are getting more relevant web pages.

Web pages are heterogeneous, unstructured and dynamic therefore finding relevant terms is difficult. In natural language many terms and phrase have synonyms therefore search engines will retrieve web pages regardless of the possible multiple meanings. Another reason users get irrelevant web pages is because many time users do not know the domain of their search query [24]. Because they do not understand the domain of the query topic they do not know the proper terms to use to find relevant web pages. Query expansion generates suggestions to the users for deepening the search context for a specific context [25]. Now that we know the importance of query expansion I will how explain how WordNet, cluster labels and relevant feedback are used to find terms for expanding queries.

2.4.1 WordNet

WordNet is an online lexical database that is manually-built to provide word relationships between different sets of synonyms. WordNet is similar to a standard dictionary in that it contains definitions of words and their relationships, but it differs since it organizes its terms conceptually and not alphabetically. WordNet has synonym sets that represent lexicalized concepts. For instance the term “software” is in the synonym set {software, software system, software package, package}. These synonym set are organized in a hierarchical tree structure that show semantic relationship between the different synonym sets. The semantic relationship between two synonym sets is a parent child relationship with the child being “a kind of” the parent. So given the “software” synonym set we have the children synonym set {program,

freeware, shareware, upgrade, groupware}. So the children synonym set is “a kind of” the parent synonym set. For example “freeware” is a kind of “software”. Also the software synonym set can have a parent set and for our case it the {code} synonym set. Figure 18 shows the parent child relationship for the “software” synonym set [26].

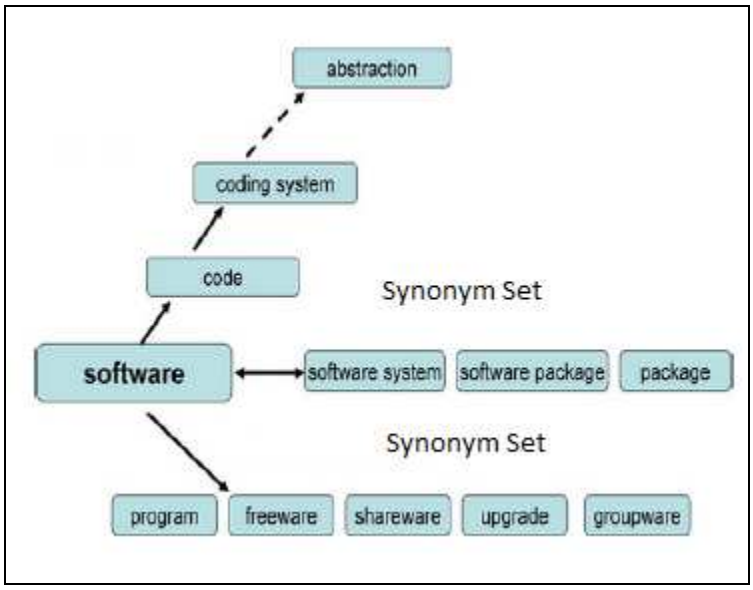


Figure 18. A parent child relations ship for the “software” synonym set Source: [26]

The purpose of WordNet is to provide a combination of dictionary and thesaurus that is intuitively usable for automatic text analysis. WordNet shows synonyms and relationships between terms that can be use for many applications including query expansion [26]. The various semantic relationships between terms in this hierarchy can be used as terms for query expansion. When users type in a query the different terms that are synonyms or are terms that are a part of the parent child relationship will be used to expand their query. By choosing the specific terms from WordNet the users are getting precise results that are hopefully what they had in mind. WordNet has proven that it can be used in query expansion, but it strongly relies on the characteristics of queries [26].

The terms generated from WordNet are not current and are collection independent.

WordNet is manually-built and outdated since the internet is changing every day. Search engines need to be up to date with current events and new products on the market because users are not going to wait. WordNet terms are collection independent because they are not based on web pages. WordNet is manually-built by humans just like the dictionary, and like the dictionary the relationship between the terms are static, structured and depend on human understanding and interpretation. As mentioned before the internet is heterogeneous, unstructured and dynamic so adding terms from WordNet can be counterproductive. Things that are logical and make sense to humans may be different on the internet, therefore terms that are added to the users query can bring noise and the search may return large amount of irrelevant results that will decrease precision [23] [26].

2.4.2 Cluster Labels

Query expansion terms can also be generated based on dynamic clustering of query results [25]. The different clustering labels can be used to identify the context of an ambiguous query [25]. By identifying the right context we disambiguate the query and get a more specific result. Clustering organizes the different topics returned by the query, and it allows the users to have an overview of the query. They can see the different topics and choose the cluster label that they see fit for query expansion. The cluster labels allow users to see different topics they may have missed if they were using typical search engines. To get the different cluster labels for query expansion the users needs to first enter the initial query. After the initial query results are returned the web pages are clustered and labeled. The different cluster labels are used to expand the original query.

Unlike WordNet cluster labels are collection dependent because the labels are generated by the title and snippets of the different web pages. Cluster labels are terms and phrases that co-occur with the initial query. As data on the internet changes so does the clustering labels. Current events and new products that are on the internet will be picked up by clustering algorithms and can be used for query expansion. It is well known that for specific topics terms and phrases that describe these topics tend to co-occur more frequently than it would for different topics. Because of this clustering algorithm can exploit this known fact to find different clustering labels to use for query expansion. There are many times users have been unsure about a query and decide to make the search anyways. As you browse through the results you start to see different terms and phrases and you realize these are the terms I should have used initially. The terms may have just passed your mind, or maybe they are new and you never thought or imagine that they can be helpful. So after you find the terms you like you copy and paste them to modify your original query. When you are using cluster labels for query expansion you are doing the same thing as when you where copying and pasting. Instead of browsing through the result of the initial query you are browsing through the different clusters that hopefully pick up the terms that you wanted to see. By just looking at the cluster labels you are saving time because the clusters group the semantically related pages so you can quickly get an overview of the specific result set and you can filter out the result that do not want to see [9]. By easily selecting the cluster label you want to use for query expansion you are saving time since you eliminate copying and pasting.

The problem with using clustering labels for query expansion terms is that sometimes the different labels are not meaningful. Sometimes different terms and phrases that co-occur with the initial query is pure coincidence and the cluster labels have no semantic meaning. The cluster

labels are generated purely from the data out there and sometimes information on the internet does not make sense [22]. Internet content are made up of web pages that are put out there by anyone and they can talk about anything they want. People occasionally put out spam that are indexed by a lot of terms in their title and snippets so that people would click on their pages and see their advertisements. These web pages usually are irrelevant to what the users is trying to find and bring noise to the query expansion process.

2.4.3 Relevance Feedback

Relevance feedback is the process of taking an initial query and taking the web pages returned from it and using the information about whether or not they are relevant to perform a new refined query. Terms and phrases that come from relevant web pages that are chosen are used as terms for query expansion, and terms from irrelevant web pages are used to filter the results. The idea behind relevant feedback is that the new reformulated query is expected to retrieve more web pages identical to the one that was marked as relevant and to remove ones that are identical to the irrelevant ones. I will mention two ways to get relevance feedback: implicit feedback and explicit feedback.

Implicit feedback is where the system attempts to estimate what the users may be interested in by looking at what web pages they clicked on after a search query to determine what pages are relevant and irrelevant. Implicit feedback assumes that if users click on a web page they are marking that page relevant and if they do not click on a web page they are marking it irrelevant. Some implicit feedback search engines go beyond just clicking the page to see if a page is relevant. They measure how long users stay on a given web page, check if users print

from the web page, and check to see if the page was bookmarked [27]. Because implicit feedback is done by the system automatically, many times different users' queries are saved to increase efficiency and to potentially be use to employ the "wisdom of crowds" to increase the chances of picking relevant pages to generate query expansion terms [28][5]. Implicit feedback is done to save time and create less work for the users when they are using relevance feedback to expand their queries. The terms are often automatically added to a users query or used to modify a user's query behind the scenes. In real time setting users usually do not want additional tasks so search engines do the query expansion without asking the users [27]. The problem with doing query expansion behind the scenes is sometimes query expansion can create noise, and I will explain noise later in this section.

Explicit feedback is based on users' indication of which web site of a search result are relevant and irrelevant. Users explicitly mark selected web pages manually as either "relevant" or "irrelevant" and terms and phrases that are extracted from these two types of web pages will be use to either to expand a query or to filter a query to eliminate noise. Terms and phrases extracted from irrelevant web pages are used to filter out the new query and they do this using the exclude words from search feature. This feature is a way to filter out terms that are returned from search engines. By putting '-' before terms you are telling the search engine to not return web pages that have that term in them. Terms and phrases extracted from relevant web pages are used to expand the new query. For explicit feedback users usually have the choice of using the different terms to expand and filter their results. Because users have to select relevant and irrelevant web pages along with choosing what terms to use this can be time consuming and may discourages the users [27].

2.4.4 Query Expansion Example

Query expansion is a technique that has been proven to be helpful and is implemented by many search engines [22][26]. A good example is the Yahoo! Web search engine in 2010. The expanded query suggestion appears just below the search bar [15]. Figure 19 shows a list of query expansion terms for the initial query “nfl”.

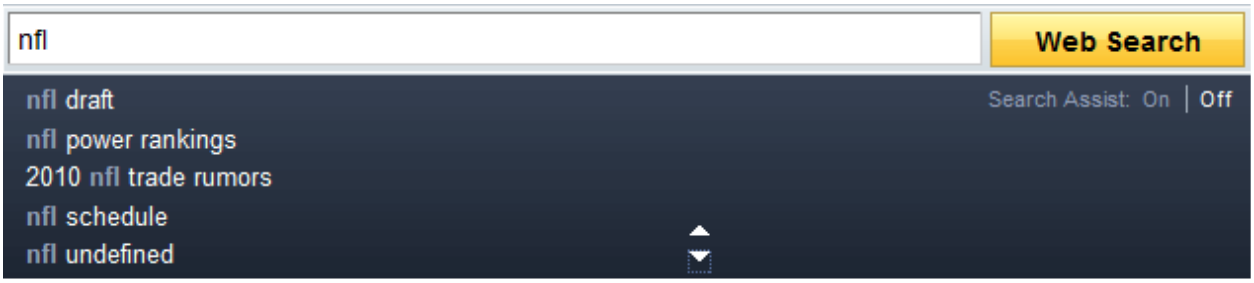


Figure 19. Yahoo! Web search engine in 2010 that uses query expansion

One of the main problems with this interface is that users can only select one query expansion term to refine their search. A lot of times when users are doing query expansion they want to select multiple terms and phrases that they can use to refine their search and this interface does not provide that. If users want to select multiple phrases and terms they need to make multiple searches and do copying and pasting.

2.4.5 Query Expansion Problems

In the Yahoo! Web search engine the query expansion terms are suggestions that they can use or ignore. Many search engines use query expansion automatically, and the users are never aware that query expansion terms are being use to refine their query [22][23]. These search engine use query expansion automatically to reduce overhead time that is wasted when users are trying to decide which query expansion terms to use. Search engines that use query expansion automatically are very confident in the terms they choose and usually have complicated

algorithms in order to choose their terms wisely. Automatically query expansion tends to improve the average overall retrieval performance by improving certain queries and making it worse on others. Quite often automatic query expansion ends up hurting the overall retrieval performance because of semantic noise. Semantic noise that is added to results leads to query drift and low precision. To make sure the benefits outweigh the cost, search engines that use automatic query expansion do a lot of testing to make sure query drift is small enough so that query expansion can still be helpful. A common test done to evaluate automatic query expansion is to look at web pages that are already classified and run searches twice. One test will be done with query expansion and the other test will be done without. Because the web pages are already classified search engines can easily calculate precision and recall to compare the two searches [24].

When query expansion is done manually especially in explicit relevance feedback it can be very time consuming. There is a lot of overhead when users have to scan through pages and decide which pages are relevant and irrelevant, they also need to select terms they want to use to expand their query. Users can select multiple terms by looking through the suggestion and select the ones that they like and dislike. This is done at runt-time and this iterative process requires clicking on web pages and terms evaluating their relevance. This overhead and extra time discourages users from query expansion and keeps them from using it. Manual query expansion needs to be done efficiently and intuitively in order for it to be utilized.

3.0 MOBILE SEARCH ENGINE

So far I talked about the popularity of the mobile phone and the increasing usage of

mobile internet and its importance. I mentioned the problem with mobile internet and how the small screen size and limited interaction methods prevent people from using search function on mobile devices. Then I spoke about clustering and query expansion and how it helps the users quickly find data from the increasing heterogeneous, unstructured and dynamic internet. My research aims to developing an innovative solution that improves the users experience for searching on the mobile internet. I will combine both clustering and query expansion to help solve the problem of mobile phones small screen size and limited interaction methods.

3.1 Tools

To do the clustering I use an open source project called Carrot2 which was founded by Dawid Weiss and Stanislaw Osinski. Carrot2 is a program that can be used to cluster web results and it can be found at http://project.carrot2.org/ [13]. The open source project can use either STC or Lingo as its clustering algorithm. To do the searching and query expansion I used three different major search engines Google, Yahoo! and Microsoft Live. They each have api's that are easily used to query their databases. The good thing about using major search engine api' is that as their information retrieval process is enhanced my mobile search engine will get those changes. The mobile search engine that I created is not actually implemented on a mobile phone. I use Java Swing to mimic a mobile phone screen size and input capabilities. The screen size and scrolling is similar to an iPhone. Users can scroll through the results by selecting any point on the result pane and moving it up and down.

3.2 Product

My mobile search engine clusters results, and thus allows users to either browse the

content or refine their search by using query expansion and relevant feedback on the cluster labels. The mobile search engine also allows the users to do a typical search and get a ranked result list back. To start this process the users first enters a search query. The picture on the left in Figure 20 shows an example of entering the query “sharks” in the text area box. If the users decides to do a ranked result lists search than they can press the “Search” button. The center picture in Figure 20 shows the results for “sharks” after pressing the “Search” button. The different web pages returned will be represented by a title of the web page (in bold), the snippet describing the web page (below the title) and the URL (embedded in the title as a link).

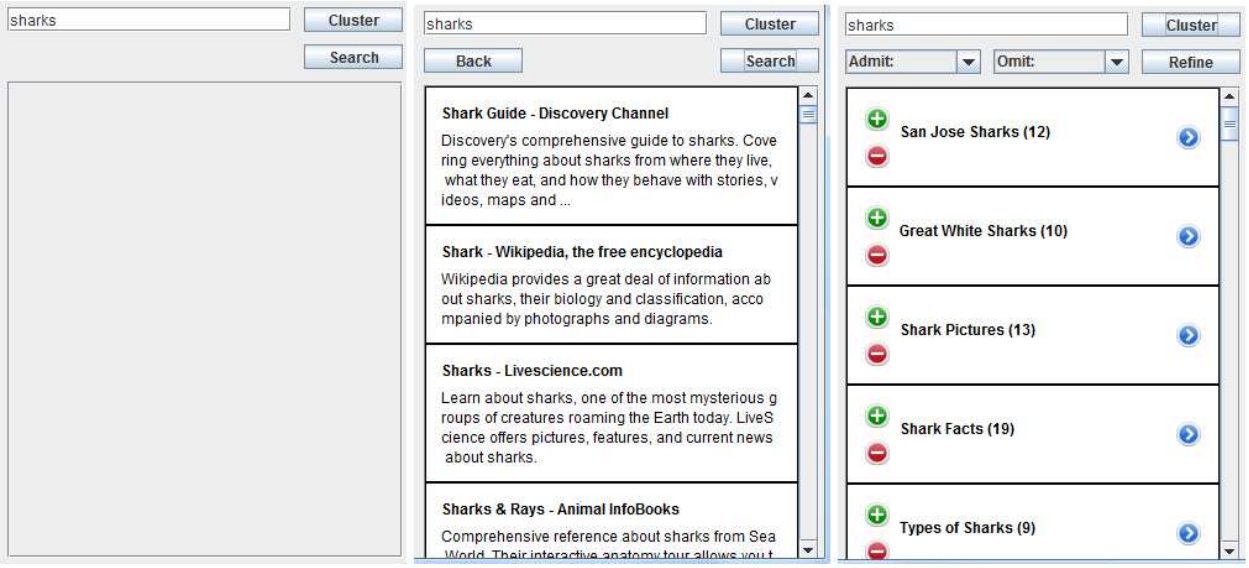


Figure 20. The initial screen, the screen after “Search” and the screen after “Cluster”

The first result from this example is titled “Shark Guide – Discover Channel” and users can get to this page by clicking on it. The snippet describing the web page is right below it. There were 199 web pages returned by this search. Many times users know the subject and the different terms to use to find what their looking for. Because of this we give the users the option of using a typical search engine. Using this search functionality allows the users to get the result faster and there is no time spent clustering results and looking through the different labels. The users can

search terms that will return a list of web pages sorted by rank that search engines assign to them. If the users decide they want to see the different cluster labels from their query they should press the “Cluster” button. The right picture in Figure 20 is the result of clustering the term “sharks”. When “Cluster” is chosen the different cluster labels are displayed (in bold) with the number of web pages in parenthesis next to them. The cluster labels give an overview of topics covered in the search results and help the users identify specific groups of web pages they are looking for. For instance “San Jose Sharks” will have web pages that are related to the San Jose Sharks hockey team in California. Only 50 cluster labels are returned which is a lot less than the 200 result that the regular search engine returned. Scrolling the 200 results with the small mobile screen is time consuming when compared to scrolling the 50 cluster labels. Clustering makes the users feel like they’re browsing, which we already know is the preferred way of information access for mobile phones.

After the users press the “Cluster” button they can view the web pages in the cluster by selecting the blue arrow next to the cluster label. The left screen in Figure 21 shows the cluster label “Great White Sharks” and the blue arrow right next to it. The right screen in Figure 21 is the different web pages that are in the “Great White Sharks” cluster label after pressing the blue arrow. The layout of the list of web pages in the cluster is the same layout that is used for the list of result returned by pressing the “Search” button.

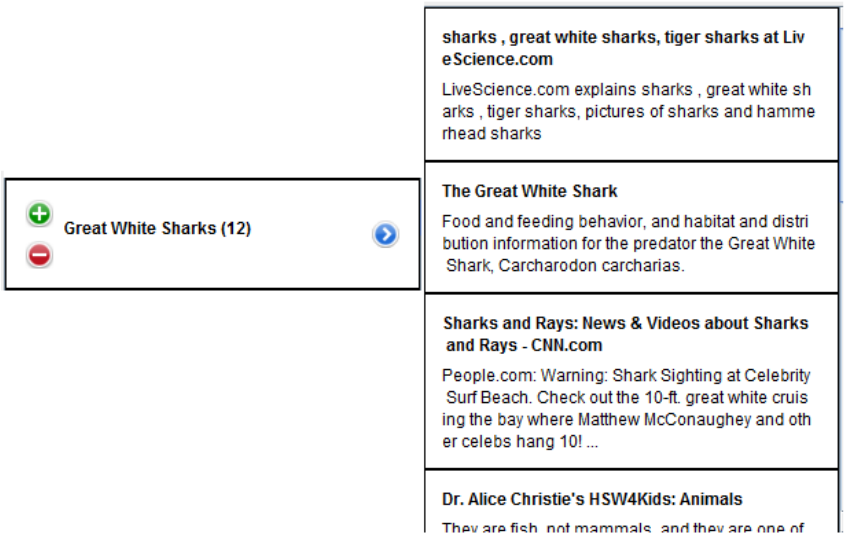
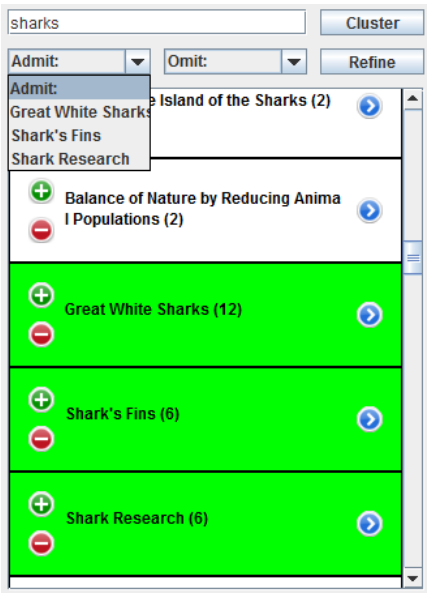


Figure 21. A cluster label and the result of looking into a cluster label

If users do not found a cluster label that has exactly what they are looking for they can use query expansion. To do this the users need to look through the different cluster labels and find cluster labels that they feel are relevant and irrelevant. When users find a cluster label that is relevant they press the green plus sign that is before the cluster label. After the users presses the green plus sign the whole box that surrounds the cluster label will turn green and the cluster label will be added to the drop down box that says “Admit”. The users can view the different cluster labels that they mark relevant by clicking on the drop down box. The drop down box will drop down and all the cluster labels will be listed. If users decides that they no longer want the cluster label they can press the red minus sign to take out the cluster label and the box surrounding the label will turn white. Figure 22 shows the different cluster labels that are marked relevant for the query “sharks”. Notice the “Admit” drop down list with all of the relevant cluster labels that are selected.



Firgure 22. The relevant cluster labels selected

For cluster labels that users find irrelevant they can press the red minus sign that is before the cluster label. After users presses the red minus sign the whole box that surrounds the cluster label will turn red and the cluster label will be added to the drop down box that says “Omit”. If users decides that they no longer want the cluster label to be marked irrelevant they can press the green plus sign to take out the cluster label and the box surrounding the label will turn white. Figure 23 demonstrates the different cluster labels that are marked irrelevant for the query “sharks”. Notice the “Omit” drop down list with all of the irrelevant cluster labels.

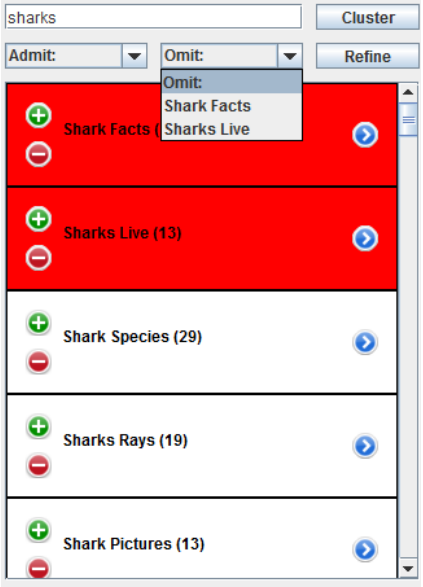


Figure 23. The irrelevant cluster labels selected

One of the advantages of having buttons to decide if cluster labels are relevant or irrelevant is that it saves users time since they do not need to type these phrases with their mobile phones. Skilled mobile phone users can type up to 21 words per minute, while PC users can do at least 60 words a minute [28]. Relying on users to type in words would probably prevent them from web surfing via mobile phone.

After users has selected the relevant and irrelevant cluster labels that they want to use for query expansion they can press the “Refine” button. The refine button will take the relevant and irrelevant cluster labels and use the Boolean advance search option found on most search engines to do query expansion. The Boolean advance option allows users to get a more specific result by taking phrases and putting “AND” between them so they can get results that have all the phrases in them. They can filter and make sure none of the results have a given phrase by putting “-” before each phrase. To use the Boolean advance option on a search engine I need to prep my relevant and irrelevant cluster labels. The first thing is to check if the string in the text area box is

contained in any of the strings in the relevant cluster labels. If it is not than I add it to the list of relevant cluster labels. The reason this check is necessary is because if the original query or what I have in that text area box is part of the list of relevant cluster labels than I do not need to use it for query expansion. The next step is to go through all the relevant cluster labels and surround each label with quotes and concatenate them with the string “AND” between them. Then I go through all the irrelevant cluster labels and surround them with quotes and concatenate “-” before each cluster label. After I finish I get a new string that has all the relevant and irrelevant cluster labels and I query it against a standard search engine. The web page results that I get back should be specific to the different phrases I picked from the relevant cluster labels and should filter out web pages that relate to the irrelevant cluster labels. The unique thing about my query expansion is that users decide what cluster labels are relevant and irrelevant and they are allowed to choose more than one. Query expansion search engines like Yahoo! only allow users to select one query expansion phrase, and I do not believe that is sufficient.

3.3 Clustering vs. Query Expansion

Now that we know a little bit about my mobile search engine and how it works, let us look at how it can be used. Clustering and query expansion complement each other in my mobile search engine. The two methods can benefit from one another and help the users find web pages efficiently and intuitively. I have already spoken about how clustering is ideal for mobile searching because the way it groups results and allows users to easily find relevant web pages by browsing through the different cluster labels instead of looking through pages of results. Clustering has had little commercial success for desktop search engines, but because of the unique limitation of mobile internet there has been a great deal of research on clustering on

mobile devices [4][12]. In the next three sections I will speak about three scenarios where query expansion improves clustering.

Clustering is a great tool for organizing search results on mobile devices but occasionally the list of cluster labels are vague and not meaningful, and thus the users do not find exactly what they're looking for in one particular cluster. They might see multiple cluster labels that they are interested in, but this becomes a problem for clustering search engine because they only allow looking at one cluster at a time. To view the different web pages in multiple clusters would require going in and out of each one of the clusters looking at their associated web pages individually. Because clustering is supposed to separate the different topics from the results, the different web pages in each individual cluster groups may not be relevant to users. The users want to find web pages relating to the combined cluster labels, but because the two clusters are different from each other the combined result from them might not make sense. For example if a user wants to see what is going in football they would cluster the query "NFL". Then as a result they see two clusters that they are really interested in: "Denver Broncos" and "Draft Picks". Figure 24 shows the different cluster groups that are returned after clustering the search term "NFL". As you can see the two clusters that the user likes are highlighted green.

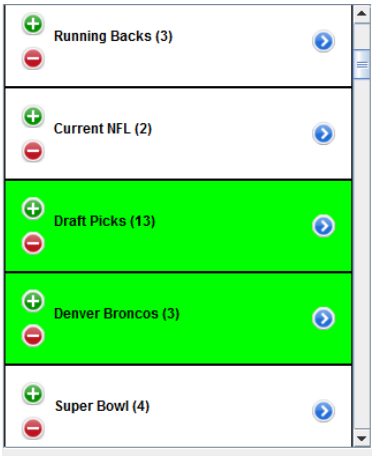


Figure 24. Two selected cluster labels “Draft Picks” and “Denver Broncos”

So the user selects “Denver Broncos” because they like this team and they select “Draft Picks” since they would like to see who the Denver Broncos are going pick in this upcoming draft. When the user goes into the “Denver Broncos” cluster they see news about the team, where to buy tickets and so on but they see nothing about the draft. So then they look in the “Draft Picks” cluster and they see a bunch of web pages that relate to the upcoming draft but they see nothing about the Denver Broncos. Separately these two clusters are different but when you combine the two meaning of the cluster label you get a whole new different topic. To solve this problem we can use query expansion. Earlier we talked about how we can use cluster labels as terms for query expansion. To solve this problem we will just do that, combine the two cluster labels “Denver Broncos” and “Draft Picks” and the pages return will be about the Denver Broncos and information about their upcoming draft. Query expansion allows users to reformulate the query and combine the two cluster label terms and resubmit the query. The newly refined query will return a more specific result that is tailored to the terms and phrases that were chosen by the users. After query expansion users should be able to select the top returned result and find exactly what they are looking for.

Another problem with clustering that query expansion helps with is viewing web pages from a small cluster. Many times clusters that may be relevant to what users are looking for are too small and only have a few web pages. These clusters may be small because the original query that was used for the initial clustering may have little similarity and co-occurrence with the small clusters. With query expansion users can select the cluster label that is relevant and use it as a query expansion term. The returned result will be a list of web pages that relate to the cluster label. So to the user it does not look like they are using query expansion. It just looks like they selected a small cluster and turned it into a big cluster. An example of this can be if the user clustered the term “Tahoe”. After the list of clusters are returned the user decides that they are interested in the cluster “North Star” because this is the ski resort they want to look at. The user sees that the “North Star” cluster label only has three web pages in it. With query expansion the user can use the cluster label “North Star” and reformulate and resubmit their query. The user will then get a whole list of web pages that relate to “North Star” ski resort that is a lot more than three web pages from the cluster. This is perfect for clustering because now we never have to worry about getting a cluster that is too small. In Figure 25 the first screen to the left shows the list of cluster labels for the query “Tahoe”. As you can see the cluster label “North Star” only has three web pages in its cluster. The second screen to the right is the result after using the cluster label “North Star” as the query expansion phrase.

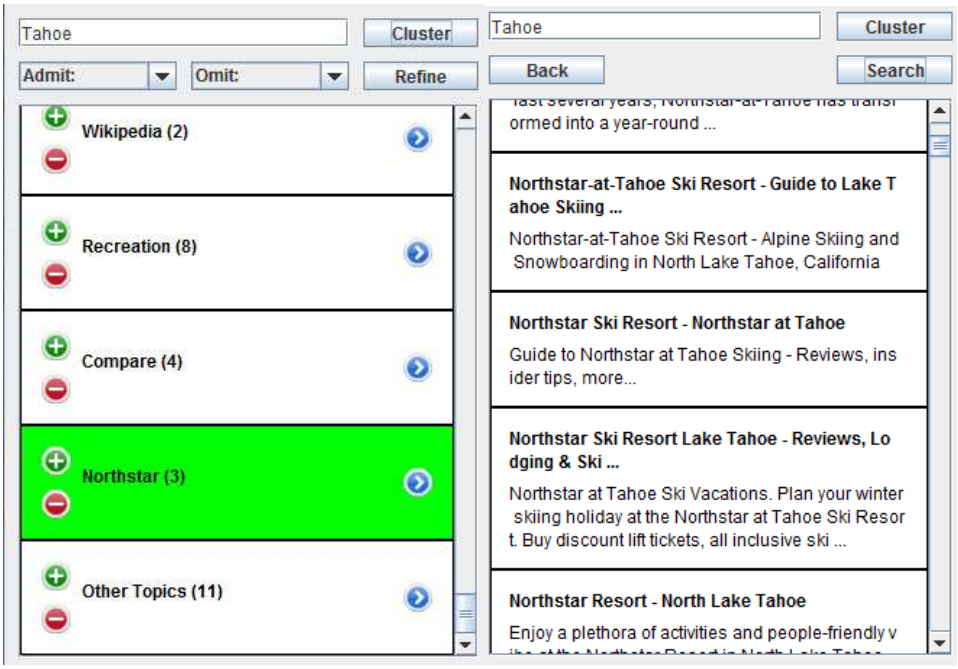


Figure 25. This is an example of doing query expansion on a small cluster.

Query expansion can be used with clustering for term and phrase filtering. A lot of times users submit a query and are bombarded with results that have a lot of noise and are irrelevant. With query expansion user can select web pages that are irrelevant and select the terms and phrase they want to remove from their results. They filter the noise that is returned by the result so they do not have to look through the irrelevant pages. Imagine clustering the query “Tahoe” again. The user looks through the cluster labels and decides that they want to look for some “Lake Tahoe Lodging” but they do not want to stay in “South Lake Tahoe”. So using query expansion they select the cluster label “Lake Tahoe Lodging” to expand their query and they chose “South Lake Tahoe” to be excluded from their result. The resulting page is a list of Lake Tahoe lodging in the North Lake Tahoe region. Since “South Lake Tahoe” was a phrase that was chosen to be excluded from the result we do not see any lodges that are from South Lake Tahoe in our list of results. All lodges that are in South Lake Tahoe most likely have the address “South

Lake Tahoe" in their title and snippets so we can be pretty sure the results return is anywhere in Tahoe but South Lake Tahoe. The first screen to the left in Figure 26 shows the list of cluster labels for the query "Tahoe". As you can see the cluster label "Lake Tahoe Lodging" is highlighted green because it will be use for query expanding and the cluster label "South Lake Tahoe" is highlighted red because the terms will be used to filter the result and all the pages that have the phrase "South Lake Tahoe" will be marked as irrelevant and excluded. The second screen to the right is the result after using query expansion for the two cluster labels. Note that the first top three pages returned are lodging for North Lake Tahoe and not South.

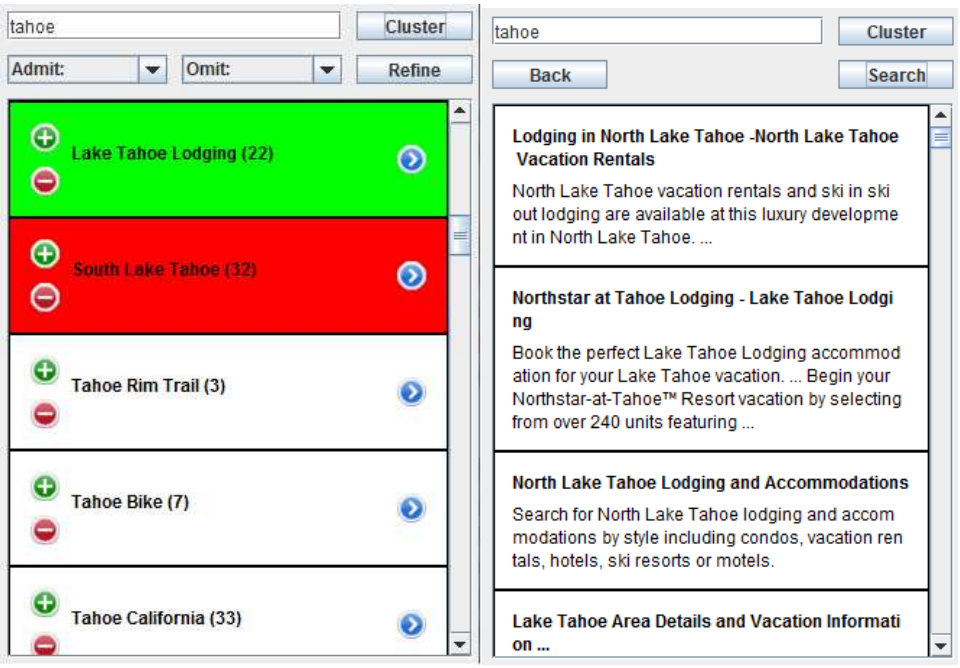


Figure 26. An example of using filtering in query expansion

In these three scenarios we are using cluster labels for query expansion terms and phrases. We are also using explicit relevance feedback because the user is using an iterative process in where they select relevant and irrelevant terms and phrase for their query expansion.

Because the user has the option of what phrases and terms are included and excluded from their results they are eliminating noise that is usually caused by query expansion. With little noise users are getting higher precision and recall and are getting relevant web paged faster and efficiently. They no longer have to sift through numerous pages of irrelevant web pages because of the time and effort put in removing the noise.

Now that we talked about how query expansion can help clustering it is also true that clusters can help query expansion. Many times query expansion consists of too much computation overhead and not worth it. The time spent finding the relevant and irrelevant phrases and terms is time consuming and the user is better off doing a regular search. When we use clustering to get the cluster labels for query expansion we cluster the different web pages returned to us. If the user sees a cluster label that is exactly what they are looking for and the cluster has enough web pages they do not have to use query expansion. So rather than using query expansion on the cluster labels the user should just be able to select a cluster label and view the web pages inside. When the user initially submitted the query to the clustering engine to get cluster labels for query expansion the cluster engine returns the cluster labels and the web pages that associated with them. Instead of using the cluster labels for query expansion the user can just view web pages in the cluster to save time. An example of this is showed in Figure 27 where the user clusters the term “iPhone” and decides that they like the cluster label “iPhone Accessories”. “iPhone Accessories ” has 11 relevant web pages and the user decide to browse through the 11 web site shown to the right.

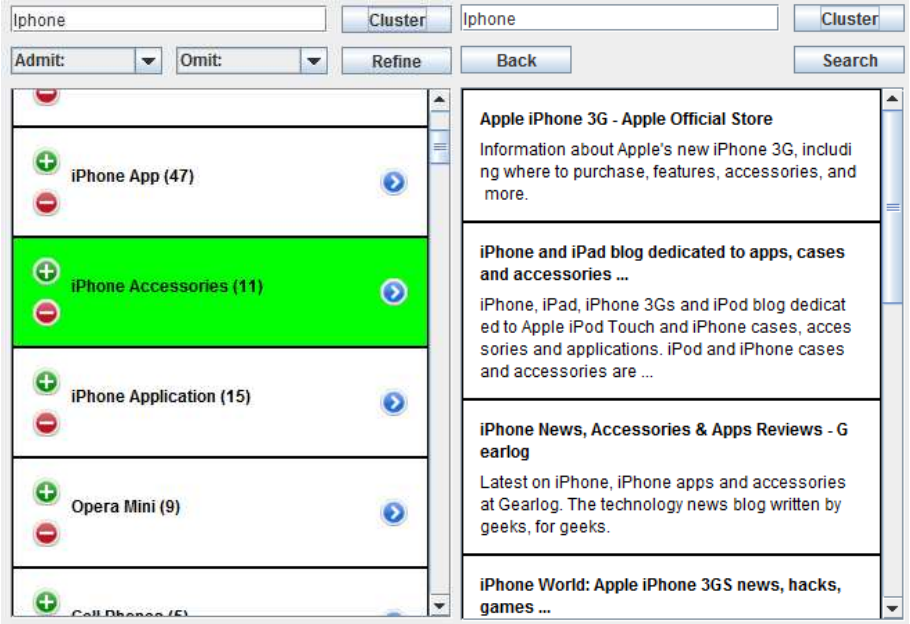


Figure 27. An example of using the clustering method

Many times viewing web pages in clusters are preferred over query expansion since users do not want to view too many pages.

Clustering is also good for query expansion when users do not know much about the subject. Because clustering will show the different topics that relate to the initial query users can scan through them and get high overview of the query topics. Users can skip over irrelevant clusters and only explore the clusters that they feel are important. The different cluster labels act as a summary of the different subjects of the query.

4.0 EVALUATION

Before actually doing any kind of evaluation on my mobile search engine I first need to decide on which search engine, cluster algorithm and settings I am going to use. These three options will greatly affect the outcome of my evaluation so I must look at them carefully.

4.1 Search engine

The three major search engines that I have been evaluating for my mobile search engine are Yahoo!, Google and Microsoft Live. These three search engines are very popular and all have java api's that make it easy for developers to query against their databases and get back web page titles, snippets and URL. The first thing I did to test which search engine I was going to use was to see which search engine returned the most results. I want a lot of results so the clustering algorithm can generate more cluster labels so I can do query expansion. To get the maximized search results returned for the three search engines I must change the number of results setting. After observing the search engine I notice that search engine api's do not allow their databases to return more than 1000 results. So I changed the results setting to 1000 and do eight queries against the three search engines. The graph in Figure 28 show the different number of web pages returned for the eight queries that I ran against Google, Yahoo! and Microsoft Live. The last graph shows the average web pages return for all eight queries.

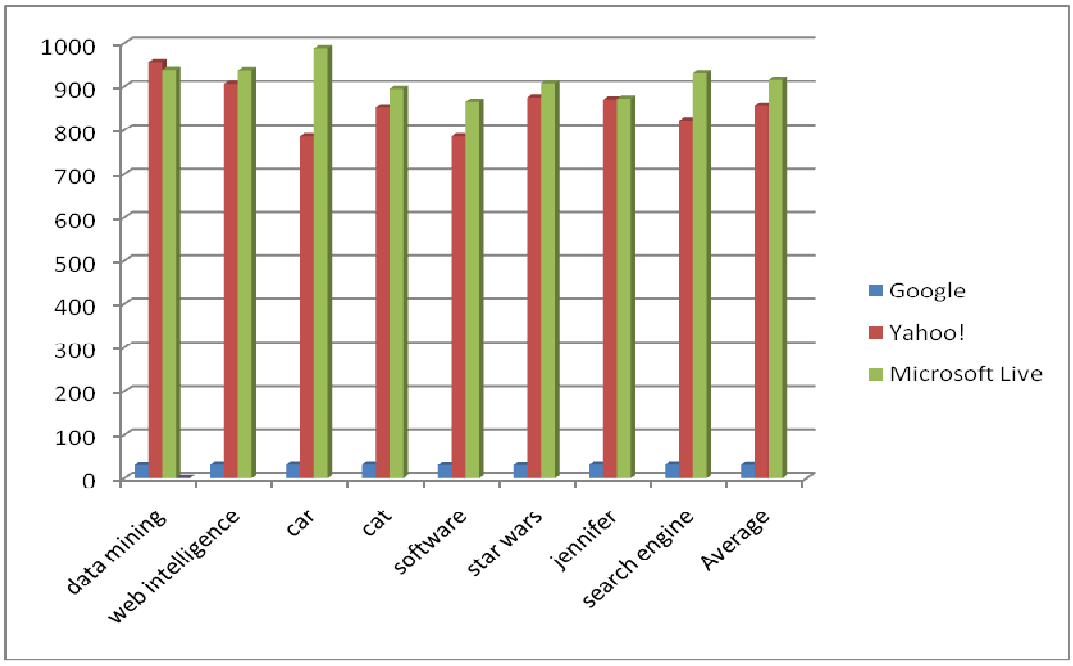


Figure 28. Number of result return from Google, Yahoo! and Microsoft Live

The average number of result return by the eight queries can also be seen in Table 2. As you can see from the graph and the table Google returns the lowest average amount of results. By looking at the graph you can see Google never returns more than 32 results and the average result returned is 31.6.

Search Engine	Average
Google	31.625
Yahoo!	856.875
Microsoft Live	917.375

Table 2. Average result return for the eight queries on the different search engines

This is a problem because with only 32 results the different clustering algorithm will only return 6-12 different cluster labels. This may be enough clusters if we were just using clustering for our mobile search engine, but because we are also doing query expansion this is not enough terms and phrases for users to scan through and refine their query. They need more options so they can mark relevant and irrelevant to get result back that is specific and do not contain noise. For general clustering the user wants a low amount of clusters because the whole idea about clustering is to try to limit the amount of information the user has to browse in order to find what they are looking for. If the number of clusters is too high the user might be better off just looking at the actual web pages instead of browsing through cluster labels. Because my mobile search engine is using both clustering and query expansion I must find a right balance that will allow users to look at clusters or use query expansion when the cluster do not provide the answers. Because using Google will only return 6-12 different clusters we will not use Google as our search engine. This is really unfortunate because Google is the most popular search engine and it is what I use personally. To get more than 32 results Google requires that developers sign up for keys. These keys are limited and determine the maximum number of queries per day and

maximum number of fetched results per query. I was only personally able to get one key which was not enough for me to do my evaluation on the Google search engine. Google limits the numbers of keys they give out to prevent traffic on their databases. So now that we eliminated Google we are left with Yahoo! and Microsoft Live. From Table 2 you can see that Yahoo! and Microsoft Live have on average 857 and 917 web page results returned respectively. For my mobile search engine I will only cluster 200 web pages and use query expansion to get 200 web pages so these two search engine return more than enough results for me. To determine which search engine to use I had 10 users try the two search engine on my mobile search engine for 5 minutes each search engine. They were allowed to search anything they wanted using any method they wanted. After 10 minutes using the two search engine I ask users which one they liked and why. Most people did not have a reason for selecting one over the other but some said the results returned from one were more accurate and relevant than the other. Table 3 has the result after 10 users evaluated the two search engines.

Search Engine	Preferred
Yahoo!	60%
Microsoft Live	40%

Table 3. Users choice between Yahoo! and Microsoft Live search

After asking the 10 different users which search engine they preferred 6 out of 10 chose Yahoo!. For this reason I decided to use the Yahoo! search engine for my mobile device.

4.2 Clustering Algorithm

The next thing that I needed to decide on was which clustering algorithm, STC or Lingo, I should use for my mobile search engine. The algorithms are very different from each other and hence have different quality and performance characteristics. The amount of time I spent understanding the algorithms will help me decide which algorithm to pick. As mentioned before my mobile search engine uses both clustering and query expansion so I need to balance the number of cluster labels that I generate. I need the option of increasing the number of cluster labels allowed without losing quality of the cluster labels. To see which clustering algorithm most fits that description I clustered the results I get from Yahoo! and counted the cluster labels that had more than two terms. This test assumes that quality of cluster labels depend on the number of terms. Because I want my cluster labels to be descriptive and specific I assume that cluster label that has more than two terms are high-quality. Before I start the test I need to first set STC so that it allows more than 15 clusters. Because Lingo defaults their "cluster count base" to 30 it usually returns about 40 different clusters depending on the query. That is why I need to change the "Maximum final clusters" for STC from 15 to 40 so the two algorithms can return the same amount of clusters. Figure 29 is a graph of the number of cluster labels that have more than two terms for STC and Lingo. The first eight graphs have the number of cluster labels for the different queries terms I used and the last graph is the average.

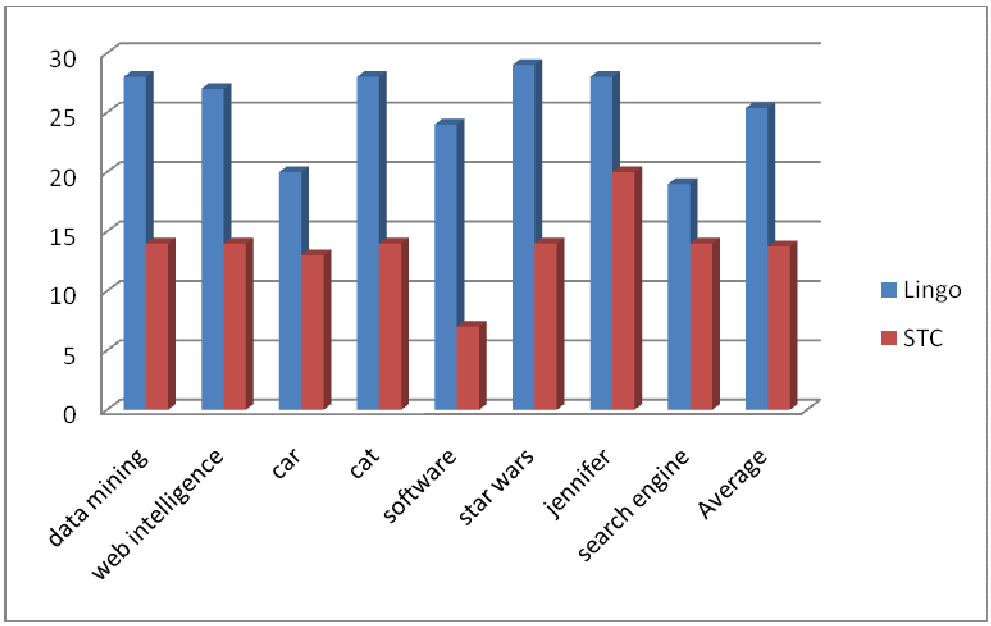


Figure 29. Number of cluster labels that have more than two terms for STC and Lingo

Table 4 has the actual average number of cluster labels with more than 2 terms for the eight queries that were clustered using Lingo and STC.

Cluster Algorithms	Average Number of Cluster Labels with more than 2 terms
Lingo	25.375
STC	13.75

Table 4. Average number of cluster labels with more than 2 terms for eight queries

As you can see Lingo returns higher quality cluster labels than STC. I even tried increasing the "Maximum final clusters" for STC and lowering the "Minimum base cluster score" to create more labels so that it will have a better chance against Lingo, but as the number of cluster labels increase the quality decreased and all the new cluster labels had one term. As I increase the number of clusters for Lingo the cluster labels that were generated still had more than two terms which is exactly what I need for my mobile search engine. Lingo cluster labels are better-formed and more diverse because of the term document matrix and SVD. Lingo cluster labels are better

formed and more diverse than STC. The reason Lingo returns long descriptive and diverse cluster labels is because it uses a term document matrix and SVD to find its cluster labels. The term document matrix allows clustering to be performed based on the different terms in the web page regardless of the order. When content is put in the term document the order of the terms is disregarded and not used for clustering. In STC phrases are used to cluster web pages and because phrases are order dependent only web pages that share terms in the exact same order will be considered a cluster. This becomes a problem because many times web pages that are similar may have words that are similar but are in different order. For instance imagine two web pages with the title “San Jose Sharks” and “Sharks San Jose”. We know by looking at the two titles that they are talking about the same thing but because the terms are in different order STC would not consider them a cluster. Lingo on the other hand would consider these two web pages similar and group them in the same cluster. To pick the cluster label to describe the cluster Lingo would just pick the phrase that had the most occurrences in the data. Another occasion where STC will not find clusters is when synonyms are used to replace certain terms in a phrase. STC only finds clusters when phrases are exactly the same, any variation of terms will not be picked up and these clusters would be disregarded. Lingo, on the other hand, will ignore these synonyms and will find other terms that these web pages have in common. Lingo is real diverse and is good at finding small outlier cluster regardless of poor co-occurrence. According to the Carrot2 developer manual the only problem with Lingo is it performance issue and it should not cause a problem as long as users are not clustering over 1000 results [13]. Because we are only clustering 200 results we will not have an issue with performance. When clustering over 1000 results STC is preferred because of it scalability and performance.

4.3 Lingo Settings

Now that we know we will use Lingo as our clustering algorithm we need to examine some of the settings that we will use for my mobile search engine. Because I know the details of the Lingo algorithm I am able to make changes to the settings and understand what I am doing. Table 5 has the Lingo settings I used for my mobile search engine.

Settings	Default	Min/Max	My setting
Cluster count base	30	2 /100	40
Size-Score sorting ratio	0.0	0.0/1.0	0.5
Cluster merging threshold	.7	0.0/1.0	0.5
Phrase Label Boost	1.5	0.0/10.0	2.0

Table 5. Settings for Lingo in Carrot2 that I used for my mobile search engine.

I mention the “cluster count base” earlier when I said it defaults to 30 but usually returns about 40 clusters. The “cluster count base” determines the number of clusters the algorithm returns. The value of this setting influences the candidate label threshold q that I talk about in section 2.3.1.3 Cluster label induction. As “cluster count base” increases the value of q will increase and this will cause more abstract concepts derived from the U matrix. Because the U matrix is dynamic and dependent on the different results that are being clustered and the rank of the original matrix we can never set the exact number of cluster labels generated. That is why even though when the “cluster count base” is set at 30 we can still get 40 cluster labels. Keeping this in mind I choose the “cluster count base” to be 40 for my mobile search engine which usually generates 50 to 60 cluster labels. I increase this number because I wanted more cluster labels to be generated so users would have more option for query expansion. Initially I had this value at 70 which generated 80 to 100 cluster labels and when I had users use my mobile search

engine they were spending too much time looking through the cluster labels. With that many cluster labels users were hesitant to use the cluster feature since there was just too many of them. Like I said before we have to find a balance for clustering and query expansion, with input from people trying out my mobile search engine and my testing I chose 40.

The next setting I looked at was the “size-score sorting ratio”. This setting is used to sort the cluster labels after they have been generated. The default value is 0.0 which means the cluster labels are sorted by the number of web pages in each cluster. This is the default value because Lingo is used for clustering and when users are browsing through clusters they want to see the biggest clusters first because that usually means that it is the most interesting and important topic. When the value of “size-score sorting ratio” is the max at 1.0 the sorting is done depending on the score of the cluster label and how relevant the cluster label is to the original query. This sorting is good to use for query expansion because we probably want to see the most relevant terms first so we can use them for reformulating our query. I chose to use .5 for my mobile search engine because it uses both query expansion and clustering so I thought splitting it down the middle would be appropriate.

The “cluster merging threshold” is the percentage of overlap between cluster’s web pages that is required for cluster to be merged. Having a low value would merge more clusters together which can means that irrelevant web pages could be group together. Having a high value could result in more clusters being generated that could be similar so we would have duplicate clusters. The default value of this setting is 0.7 and I changed it to 0.5 because I was seeing too many similar and duplicate cluster labels during my testing.

The last setting I changed was the “Phrase Label Boost” setting. This setting gave weight to multi-words labels against one-word labels. Higher value would favor multi-words and decrease the number of one-word labels. Because I am using cluster labels for query expansion and I want to be descriptive and specific I chose to change the default value from 1.5 to 2. The max number is 10 but when my value was too high the different cluster labels were too specific and too long which prevented users from combining cluster label for query refinement.

4.4 Evaluation Set-up

After selecting the search engine Yahoo!, the clustering algorithm Lingo and the above settings I had to modify my mobile search engine so that it can measure the amount of time it takes users to find relevant web pages using clustering, query expansion and searching. I already mentioned before that my mobile search engine was capable of getting standard ranked list results so I just need to compare that functionality with clustering and query expansion. To compare the different methods of information access I logged the amount of time when a search starts until the time the user finds what they are looking for. The timer starts as soon as the user either presses the “cluster” button for clustering or the “search” button for the standard search engine. It does not matter if the user uses clustering, query expansion or standard search engine my mobile search engine will always return a list of web pages in the last step. The timer stops when the user clicks on a web page and goes through the contents and then goes back to the search engine to mark the page as relevant three separate times. To mark the web page as relevant a popup box is displayed to the user when they click on a web page from the list. When the pop up box is displayed the timer is paused and the user has a chance to review the web page without feeling rushed since the timer is not running. After the user reviews the web page they go

back to the pop up box and mark “yes” for relevant and “no” for irrelevant”. If the user selects “no” the timer continues and they have to go on to the next web page. If the user selects “yes” I either stop the timer or continue depending on how many relevant web pages the user has already found. The reason I stop the timer after the third relevant web page is because people normally do not find everything they are looking for after one web page. It usually takes browsing through several web pages until users are satisfied, and that is why I decided on three relevant web pages before the timer can be stopped. To log all my times that I use to evaluate my search engine I use log4j.

4.5 User Study Results

Now that my mobile search engine is ready I will evaluate it with a standard search engine. To do this I have a user study of 16 participants. These 16 participants will answer a set of question that I will provide them and my mobile search engine will log how long it takes for them to answer my questions. For the first three questions the users are told to only use the search functionality of my mobile search engine. Then the next three questions they are told to only use the clustering functionality. For last three questions they are told to use the query expansion functionality. The sets of questions were mixed up into 4 different questionnaires so that the different questions will be used on all methods by different people. I did this so I can look at one question and see which method found the relevant pages the fastest. Table 6 and Figure 30 have the average time it takes users to find relevant web pages which was obtained from my user study.

	With Processing Time	Without Processing Time
Search	30.87s	26.91s
Refine	28.89s	20.34s
Cluster	23.16s	17.49s

Table 6. Average time to find relevant web pages using three different methods

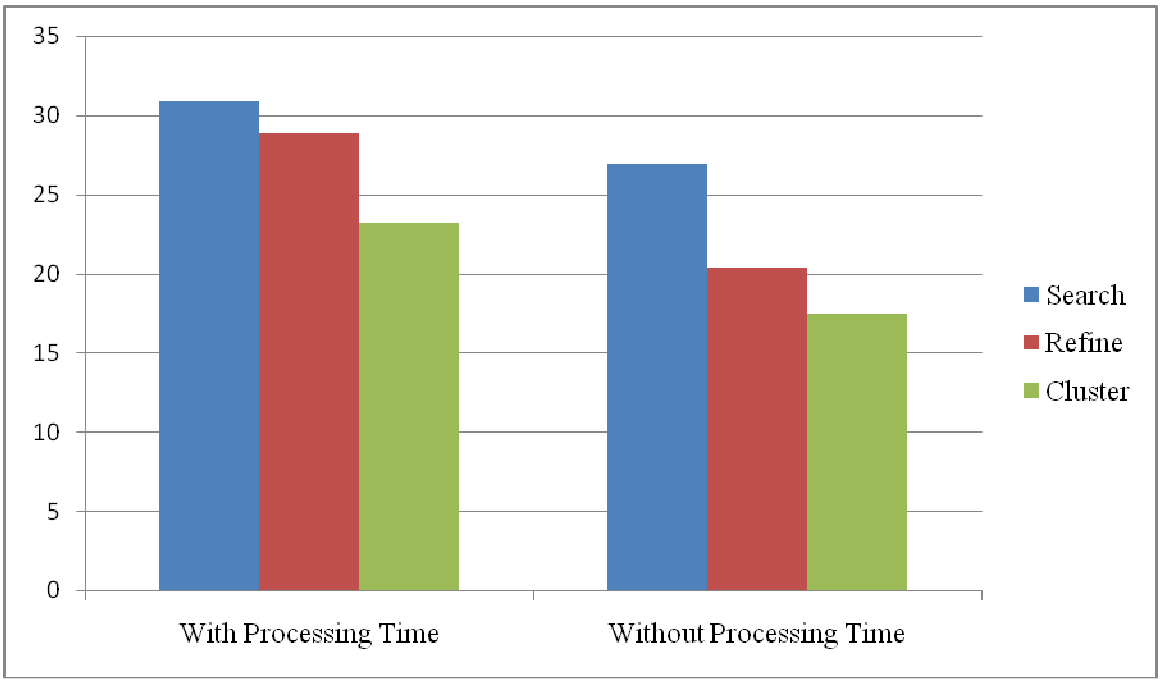


Figure 30. Average time to find relevant web pages using three different methods graph

I looked at the logs and calculated the average time to find relevant web pages with and without processing time. Processing time includes the amount of times it takes a search engine to return web pages and the time it takes the clustering algorithm to cluster the web pages. I discovered that the fastest method to finding relevant web pages was clustering. The next quickest method was query expansion and the slowest method was the standard search engine. I was very pleased from the results because it showed that even though there was overhead in clustering and query expansion the overall time it took to find relevant web pages was still significantly less than if the users did a regular search. My experiment reveals that both clustering and query expansion

can provide satisfying result for web query performance and even outperforming traditional search engines. Without looking at processing time my data showed that query expansion and refinement was 25% and clustering was 35% faster at finding relevant web pages than the standard search engine. With processing time query expansion was 7% and clustering was 25% faster than the standard search engine. The reason why query expansion speed was greatly reduced when I included process time is because of how much processing query expansion requires. Table 7 shows the average processing for the three different methods.

	Processing Time
Search	3.96s
Refine	8.55s
Cluster	5.67s

Table 7. Average processing time for the three methods

Query expansion and query refinement has the most processing time because it must first get 200 results from the standard search engine, apply the clustering algorithm on the results and then use the new query that was generated using query expansion to run against the standard search engine. Clustering must get the 200 results from the standard search engine and then apply the clustering algorithm and that is why it has less processing time than query expansion. The fastest processing time is the standard search engine because all it has to do is get the 200 results and display it to the user.

Another metric that I get from my study is how long it takes users to find relevant web pages without processing and overhead. Overhead for clustering is the time it takes the user to find the relevant cluster label and for query expansion its the time it takes to find the relevant and

irrelevant cluster labels. Table 8 and Figure 31 shows the average amount of time it takes users to find relevant web pages with and without overhead for the three methods. The last column of Table 8 is the average amount of overhead time that was used to find relevant web pages for the three methods.

	With Overhead	Without Overhead	Overhead Time
Search	26.91s	26.91s	0s
Refine	20.34s	6.09s	14.24s
Cluster	17.49s	5.79s	11.7s

Table 8. Average time to find relevant web pages with and without overhead

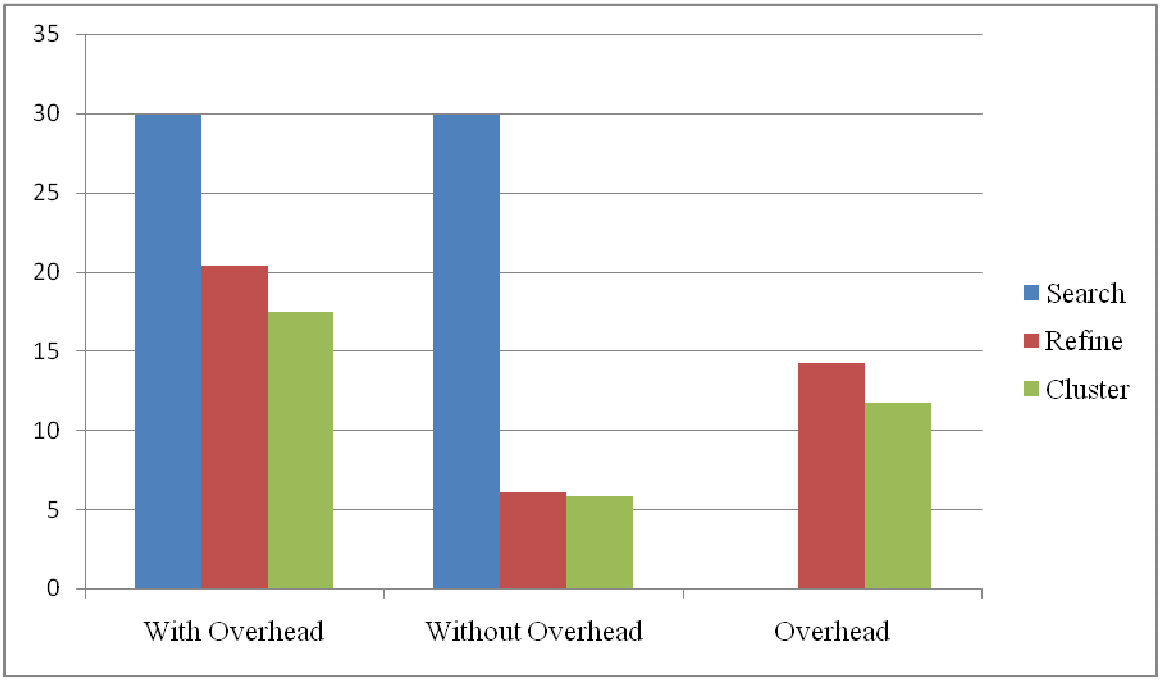


Figure 31. Average time to find relevant web pages with and without overhead graph

Obviously there is more overhead in query expansion than clustering and my data shows that. Clustering only involves looking for one relevant cluster label where query expansion involves looking for multiple relevant and irrelevant cluster labels. Traditional search engines do not require any overhead since they just need to press the ”Search” button. When looking at the average amount of time it takes the user to find the relevant web pages without overhead we are

calculating the amount of time it takes the user to find the relevant web page when they are given the list of results regardless of the method. Because query expansion and clustering spent so much overhead time they were able to find their web pages quickly when they were presented with the list of results. One thing that did not make sense to me at first was why it only took clustering 5.79 seconds to find relevant web pages without overhead when query expansion took 6.09 seconds. I thought query expansion was going get a much better time because of the amount of overhead put in to get a more specific result without noise. After looking at my results for each question I realize there were some questions that had a bad effect on query expansion.

Now I will talk about some of my questions and how they affected my results. The first question I would like to talk about is below.

With query “Shark” look up three different things about the animal.

This question caused the average time for finding relevant web pages for query expansion to increase and was one of the questions that I mentioned earlier. Query expansion’s main purpose is to get a more specific result with little noise and because the question is asking us to look at the different topics of the animal we have a problem. Users can see the different topics when they are browsing through the cluster labels looking for relevant and irrelevant terms but with query expansion they cannot view them all in the results. Users try selecting multiple topics hoping the query expansion will give them results from all the topics they selected but because my mobile search engine uses the “AND” option the query will only return web pages that relate to all of the relevant cluster labels that were selected. What the user think they are doing is using the “OR” option which will return all web pages that relate to one or more of the relevant cluster labels selected. Because the “AND” option is used the web pages returned are usually irrelevant

because the user just selected three different topics about sharks and is trying to find web pages that talk about all three topics. Users usually end up redoing the refinement looking at one cluster label at a time which is time consuming because they are using a lot of processing time and overhead. Other users just continue looking through the refine results browsing through the 200 different results. By looking at the data I can see that users spent a significant amount of time answering this question with query expansion. Users took longer on this question than if they just used a traditional search engine. For the traditional search engine users would spend a lot of time browsing through the 200 results looking for the three different topics about sharks but was faster than query expansion because there was no overhead. Clustering was the fastest method when answering this question because users could easily see the different topics about sharks and go into each one to view the individual pages. Clustering is perfect for multi-topic information queries.

The next question I would like to talk about is below.

Find websites to watch the show “Lost”.

The traditional search took a long time to find the relevant web pages because there is so much information and web pages that talk about “Lost” in the internet. Users had to sift through a many irrelevant pages in order to find what they were looking for. Major search engine do use clustering so they can get the different topics on the first page and this was shown in my results. When users was looking for web pages where they can watch “Lost” they often found the first web page fast but took longer for them to find the second and third web page. This is because the search engines do not use clustering to group web pages they just use it so they can get the different topics and make sure each topic is represented on the first page. To find more pages to

relate to the one the user just found they have to keep on looking through the different irrelevant web pages. For clustering this question was completed quickly since users can instantly see the cluster label "Watch Full Season with all Episodes" and this cluster had 60 web pages in it. From personal experience this is really convenient because when I am watching a show online I go through multiple pages trying to find the best quality and speed, and with clustering I can just click on the cluster and look through the 60 web pages that are in that cluster. Query expansion is also quick for this question but is slower than clustering because the extra processing time it takes to make the query refinement.

Another question I would like to talk about is below.

With query “NFL” look for latest news about the NFL that does not relate to the draft.

This question was nearly impossible for the traditional search engine because the draft was coming up and every NFL web pages was talking about it. Every web page the user would click on mentioned the draft. Many users were getting frustrated and wanted to give up. Clustering was a little better but still had problems because users would click on cluster label that had the team name like “Dallas Cowboys” thinking the team web site would not talk about the draft but they were wrong. Users usually had to scroll down to the “Others” cluster in order to find web pages that do not deal with the draft. Query expansion really shines on this question because of the filtering functionality. To answer this question efficiently which most users did they had to select a relevant cluster label like “Latest News” and filter out irrelevant labels like “Draft”. The refined results returned web pages that talked about the latest news but did not mention the word “draft” which was exactly what we were looking for.

After the user answered the nine questions and were familiar with the strengths and

weaknesses of the different methods for my mobile search engine I asked a few more questions. However, this time instead of restricting the user on which method they can use to answer the question I allowed them to use whichever method they preferred. As they answered the questions I took notes on what method they chose. Table 9 and Figure 32 shows the result of what method the user preferred on my mobile search engine.

	Search	Cluster	Refine
% Method Used	22%	45%	33%

Table 9. The preferred method users decided to use to answer questions

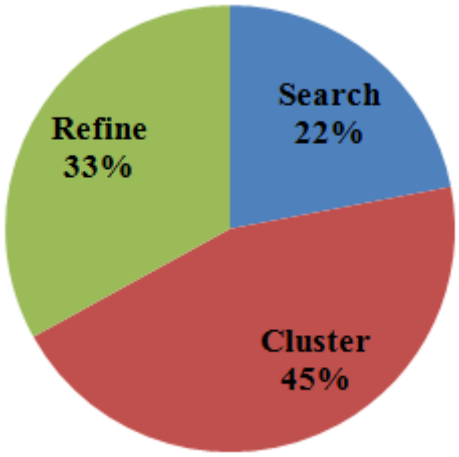


Figure 32. The preferred method users decided to use to answer questions graph

As you can see from the above questions clustering excelled on most of them and was probably why it was the method most used. Clustering is used more than query expansion because most users find what they are looking when they are browsing the cluster labels. There is no need to refine the query and waste overhead and processing time when the relevant web pages can be found in the cluster. I also notice that query expansion is the next method most commonly used and this is because of the questions that require filtering. Query expansion allows filtering and eliminating noise and this method is very effective when used correctly.

5.0 CONCLUSION

Implementing an unsupervised clustering algorithm which bases its cluster labels from frequently occurring words is challenging and has not had widespread commercial adoption for desktop application. That is why my mobile search engine combines query expansion with cluster labels so users can refine their search when clustering is not sufficient. Query expansion is good when users do not know what they are looking for and they can see other terms that relate to theirs. With my mobile search interface users are encouraged to use query expansion because instead of typing to refine their query they can select and deselect cluster labels that they believe are relevant and irrelevant. My results show that users will have no problem spending the time on refining their search because they get exactly what they want faster and efficiently without noise. Clustering and query expansion can help solve the issues that come with using the internet on mobile phones with small screen size and interaction limitations. Clustering and query expansion allows users to access specific and detailed information quickly and efficiently and thus is exactly what is needed for mobile phone users. The study conducted shows that my mobile search engine clearly was preferred by the participants over traditional ranked result list when it comes to finding relevant web pages.

6.0 FUTURE WORK

In the next section I will talk about how and why I want to integrate WordNet into my mobile search engine. I will also talk about how I want to evaluate my search engine in real life situations on real mobile devices.

6.1 Using WordNet for Query Expansion

I discuss some of the strengths and weaknesses of WordNet for query expansion but I was not able to get it into my search engine. WordNet allows more structure and certainty for query expansion, and I believe it can be used to help my mobile search engine. Cluster labels are good for query expansion but because they are generated directly from the query results they can be inconsistent. When the results are clustered the clustered labels are generated from frequently occurring words and phrases. If the results that were initially clustered did not have any relevant web pages than the cluster label will be useless. Irrelevant results usually occur when users have little understanding of the subject and cannot come up with high-quality initial queries. If WordNet is used people would see synonyms and different words that relate to the initial query that could help them. These words are generated manually in WordNet and can help the user understand the topic by letting them view suggestions. There may be times when cluster labels are not useful and looking at synonyms and related words can help.

To integrate WordNet into my application I would get the different terms return by WordNet and display them with the cluster labels after the user presses the “Cluster” button. The different terms would be have to be scored somehow and sorted with the list of cluster labels. Obviously we cannot put the number of web pages in parenthesis next to the term because there are no web pages and we did not do any clustering. Zero would be put in between the parenthesis to indicate that the terms were generated by WordNet. After the user presses the “Cluster” button and the list of cluster labels and WordNet terms are returned I would allow users to use the plus and minus button on the term for query expansion. Also I would allow users to press the blue arrow on terms from WordNet to query it against the search engine. Using the blue arrow makes

the user feel like their looking at a cluster when they are really looking at the term returned by WordNet. To the users the cluster labels are the same as WordNet terms. The user should not even notice that WordNet was use to generate the different labels that can be viewed or used for query expansion. WordNet can be an excellent tool for query expansion and its performance would strongly depend on the characteristics of the query so we must evaluate it carefully if we intend to use it.

6.2 Field Testing on Mobile Devices

For the sake of time my mobile search engine was tested on a desktop computer using Java. I only mimicked the limited screen size of a mobile device and its limited interaction methods. For me to fully evaluate my mobile search engine I need to implement it on a mobile device and allow users to use my search engine in real world situations. The questions that I ask in my questionnaire are ones that I come up with and may or may not be questions that are relevant to the mobile internet. For me to really be able to evaluate my mobile search engine users need to test out my application in real life situations. Mobile internet usage is different from desktop internet usage because users use their mobile internet while they are out and for purposes such as looking up restaurants, reading articles, looking up directions and much more. I may find out that my search engine is perfect for desktop searches but has efficiency and performance problems on mobile devices. It could be possible that users do not need clustering and query expansion on a mobile environment. These are the type of things I want to find out so I can improve my mobile search engine and get better results. By allowing users to use my search engine out in the real world I can track methods they use, queries they make and measure how long it takes for them to find relevant data. I need to see that my mobile search engine will

benefit mobile internet users and that the clustering and query expansion is used. By testing out in the real world I can also check if users are using my search engine correctly and efficiently and that my interface is intuitive. This field experiment could increase the validity of my search engine and further support my conclusion.

REFERENCES

1. Kolko, E. Beth. Studying Mobile Phone Use in Context: Cultural, Political, and Economic Dimensions of Mobile Phone Use. 2005 IEEE International Professional Communication Conference Precceding, 205-212, 2005.

2. Market Information and Statistics Division Telecommunication Development Bureau
International Telecommunication Union The World in 2009: ICT facts and figures. Retrieved from http://www.itu.int/ITU-D/ict/material/Telecom09_flyer.pdf
3. Kaki, Mika. Findex: Search Result Categories Help Users when Document Ranking Fails. PAPERS: Document Interaction, April 2005.

4. Carpineto, C., Mizzaro, S., Romano, G., Snidero, M. Mobile Information Retrieval with Search Results Clustering: Prototypes and Evaluations. Journal of the American Society for Information Science and Technology, 877–895, February 2009.
5. Church, K., Keane, M., Smyth, B. Towards More Intelligent Mobile Search. Smart Media Institute, University College Dublin, Belfield.

6. Jansen, B., Spink, A., Saracevic , T. Real Life, Real Users, and Real Needs: A Study and Analysis of User Queries on the Web. Information Processing and Management, 36(2), 207-227.

7. Smyth, B., Cotter, P. Personalized adaptive navigation for mobile portals. In Proceedings of the 15th European Conference on Artificial Intelligence, 608-612, 2002.
8. Church, K., Smyth, B., Cotter, P., Bradley, K. Mobile Information Access: A Study of Emerging Search Behavior on the Mobile Internet. ACM Transactions on the Web, Vol. 1, No. 1, Article 4, May 2007.

9. Crabtree, D., Gao, X., Andreae, P. Improving Web Clustering by Cluster Selection. Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, 2005.
10. Osinski, S., Stefanowski, J., Weiss, D. Lingo: Search Results Clustering Algorithm
Based on Singular Value Decomposition, Institute of Computing Science, Pozna´n University of Technology, ul. Piotrowo 3A, 60–965.

11. Nirkhi, S., Hande, K., Raisoni, G. Optimization of Context Disambiguation in Web Search Results. Conference on Computer Science and Information Technology 2008.

12. Heimonen, Tomi. Mobile Findex: Facilitating Information Access in MobileWeb
Search with Automatic Result Clustering. Hindawi Publishing Corporation Advances in Human-Computer Interaction, Volume 2008, April 2008.

13. Osinski, S., Weiss, D. Carrot2 Clustering Engine. Retrieved from http://search.carrot2.org/stable/search
14. Zamir, O., Etzioni, O. Web Document Clustering. Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998.

15. Zamir, Oren E. Clustering Web Documents:
A Phrase-Based Method for Grouping Search Engine Results, University of Washington
1999.

16. Branson, Steve. Clustering Web Search Results Using Suffix Tree Methods. Stanford University.
17. Carpineto, C., Osinski, S., Romano, G., Weiss, D. A Survey of Web Clustering Engines. ACM Comput. Surv. 41, 3, Article 17 (July 2009), 38 pages.

18. Mecca, G., Raunich, S., Pappalardo, A. A new algorithm for clustering search results. Data & Knowledge Engineering 62 (2007) 504–522, 2007.
19. Osinski, Stainslaw. An Algorithm for Clustering of Web Search Results. Poznań University of Technology, Institute of Computing Science, June 2003.

20. Osinski, S., Weiss, D. A Concept-Driven Algorithm for Clustering Search Results. IEEE Computer Society, May/June 2005.

21. Limbu, D., Connor, A., Pears, R., MacDonell, S. Contextual Relevance Feedback in Web Information Retrieval, Information Interaction in Context, 2006.

22. Jin, Q., Zhao, J., Xu, B. Query Expansion based on Term Similarity Tree Model. Institute of Automation, Chinese Academy of Science.
23. Gan, L., Wang, S., Wang, M., Xie, Z., Zhang, X., Shu, Z. Query Expansion based on Concept Clique for Markov Network Information Retrieval Model. Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008.
24. Custis, T. Al-Kofahi, K. A New Approach for Evaluating Query Expansion: Query-Document Term Mismatch. SIGIR 2007 Proceedings Session 24: Evaluation IIISIGIR'07, July 23–27, 2007.

25. Bordogna, G., Campi, A., Ronchi, S., Psaila, G. Query Disambiguation Based on Novelty and Similarity User's Feedback. 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2009.

26. Gong, Z., Cheang, C, Hou, L. Web Query Expansion by WordNet. LNCS 3588, pp. 166 – 175, 2005.

27. Leroy, G., Lally, A., Chen, H. The Use of Dynamic Contexts to Improve Casual Internet Searching. ACM Transactions on Information Systems, Vol. 21, No. 3, July 2003, Pages 229–253.

28. Fu, S., Pi, B., Desmarais, M., Zhuo, Y., Wang, W., Han, S. Query Recommendation and Its Usefulness Evaluation on Mobile Search Engine. 2009 IEEE International Conference on Systems, Man, and Cybernetics, October 2009