Clustering Wikipedia Search Results

Saravanan Thirumuruganathan
Saravanan.thiurmuruganathan@mavs.uta.edu
Department Of Computer Science, University of Texas at Arlington

ABSTRACT

The ranked query model used in current search engines does not work well for polysemes or incomplete or inexact queries which is quite common if the user is non-technical. One of the solutions is to do a clustering of top-N search results based on document and topic similarity and show a concept hierarchy to the user. The user can then drill down to the cluster which is closest to the concept he/ she had in mind . A good clustering search engine needs tweaking of multiple parameters and must have a good labeling algorithm. In this course project, I have attempted to create a clustering search engine based on Wikipedia data set which returns good clustering results and has an intuitive cluster labeling algorithm. The initial results are promising and the results appear equal to or better than the current popular clustering engines like clusty, mooter and webclust.

1 INTRODUCTION

Search engines have become the most common tool people use to search for information. When a user types in a query, he is given a list of millions of articles, of which he is assumed to make sense of. Typically, by good ranking by Search engines, users find what they want in one or two pages. For inexact queries or polysemes like Apple, Jaguar, Graph, Mccain etc the top results may not always be valid. This leads the user to modify his/her query and the process repeats till the user gets the result desired or is frustrated.

The reason is twofold: the search engine returns the result ranked based on its own internal algorithm without regard to user's semantic intentions and also that it only shows a fraction of the result set. In other words, search engine returns mostly unstructured data and the user is expected to make sense of it.

Example 1 : When a user queries for Jaguar , he might be looking Jaguar , the animal or Jaguar the car , or Jaguar the software . So giving a flat list is not suitable here.

Example 2 : When a user queries for say McCain , he might mean his senate life or his presidential campaign or about Sarah Palin or may be even his wife Cindy McCain. Again common search engines return a flat result here.

The most common solution for this problem is clustering of search results. Clustering solves the problem by creating groups of documents in search results based on topic similarity, and labeling each cluster intuitively based on its central concept. This allows the user to see a conceptual gist of many more documents (typically 100 or 200) and the user can directly jump to the concept closest to what he / she had in mind. The list of clusters can be considered as a concise summary of the search results for the query. Continuing the example, some of the clusters for Example 1 can be Car , Animal and Software while for Example 2 it can be early life, military career, presidential campaign, Cindy McCain , McCain Palin campaign etc

Although a clustering search engine is clearly beneficial to the user, there has not been lot of them. And most of the popular search engines don't have clustering built into them. Clusty seems to be the most popular one with decent results, although its labels are not the best that can be identified. In my project, I had tried to build a clustering searching for English Wikipedia. The clustering is based on classical K-Means algorithm. My major contributions are the many heuristics applied to make the search engine efficient and accurate. I had also developed a POS based Phrase label method that is superior to the current clustering search engines.

2 RELATED WORK

There are a handful of clustering search engines like clusty, mooter and webclust. Each of them has their own set of strengths.

The University of Washington had a clustering engine called Grouper which is described in [1]. That paper also discusses some of the challenges faced in a search results based clustering search engine. Although published in 1999, the points analyzed still remain valid. Grouper was based on Suffix trees.

Anton et al [2] compares clustering algorithms in Machine Learning and Information Retrieval.

Zeng et al [3] attacks the clustering problem from the opposite direction - Cluster labels are generated first via a phrase ranking mechanism and then the documents are assigned to each of these phrases. From literature, it produces good cluster labels. Lingo [4] employs a similar algorithm although it uses SVD as its core algorithm.

My label generation algorithm is quite similar to Lingo but is generated after creating clusters and uses a simple counter based Label selection instead of SVD. Another difference is in the concept of snippet. In this paper, a snippet for label generation is the entire document while in other algorithms it is the snippet given by the meta-crawler from search results.

3. PROBLEM STATEMENT

As per [5], the goal in hard flat clustering as follows. Given (i) a set of documents $D = \{d1, . . . , dN\}$, (ii) a desired number of clusters K, and (iii)
an *objective function* that evaluates the quality of objective function a clustering, we want to compute an assignment $\gamma : D \rightarrow \{1, . . . , K\}$ that minimizes (or, in other cases, maximizes) the objective function.

This paper deals with a reduced universe of the web as it takes does clustering only on the English Wikipedia site. The project , falls under a special type of flat clustering called "Search Result Clustering" which is subject to additional requirements which are detailed in [1] . The most important issues are listed here: Relevance, Browsable Summaries, Overlap, Snippet-tolerance, Speed, Incrementality.

Terminologies

Here are some the list of terminologies used in the paper.

Dataset : The latest xml dump of en.wikipedia.org download from [6]
Document : A Wikipedia article in the xml format. It is obtained by preprocessing the Wiki xml dump. It contains all the details of a Wikipedia article like title, content , categories, links, images , references etc.
Search Result clustering : A special type of document clustering where the documents are the results of some query in a search engine.
Non-Exhaustive Clustering : A document need not always be assignment to a cluster. As against exhaustive clustering where all documents *has* to be assignment to *nearest* cluster.
Soft Clustering : A document can simultaneously belong to multiple clusters.
N : The total number of documents considered for search result clustering
K : The number of initial clusters.
M : The size of the vector space.
$\mathbf{d_i}$: ith document
$\mathbf{R_i}$: Rank of ith document.
$\mathbf{C_j}$: The number of documents in the cluster j.

4 IMPLEMENTATION

This section describes the architecture and other implementation details of clustering search engine.

The project employs the following clustering techniques : Search result clustering, non exhaustive clustering and soft clustering. The various steps involved are detailed below.

4.1 Preprocessing

The aim of this stage is to convert the Wikipedia contents to a format easier to analyze and also to do other document enriching techniques like identification of categories. Preprocessing is necessary because the project uses a mostly static document repository which lends itself to this optimization step. Also calculation during the runtime phase will be very expensive.

The following are the steps done in this stage : Stripping of Wiki-Markup, identification of redirects and stub articles and marking them, identifying list of links, categories, reference, see also links. Additionally the document is also parsed, removed of stop words , stemmed via a simple porter stemmer and the result is stored in a field. The list of words which are links to other Wikipedia articles are also stored. So an xml file after preprocessing stores more information about the document. This also contains atleast three versions of the document content: The content without markup, content without markup , stemmed and devoid of stop words and the one which is a superficial abstract of the Wikipedia page which contains only the link text in the page.

4.2 Runtime Phase

This section describes the implementation of the search engine from the time user enters a query to the time user sees a clustered result.

4.2.1 Obtaining Ranked Search Results

Once the user enters the search query we contact a search engine with the query and get the top 100 results. Parameters to avoid adult and hate content are specified. The top 100 documents are treated as a valid , high probable , representative subset of the total n documents returned by the search engine. Since this is an online environment, compromise had to be made between accuracy and speed and hence the clustering of top 100 documents alone.

4.2.2 Search Results Preprocessing

Once the search results are obtained, the urls are checked against a local database to weed out redirects and stub articles. The articles marked whose neutrality is disputed are also removed. But the talk , category and disambiguation pages of a Wikipedia page are not removed as they are usually a rich source of labels. Articles not present in the local disk but present in search results are also discarded. The processed document data for each article is retrieved. A vector space consisting of all words with occurrence greater than minimum occurrence limit is created. The document vector is based on TF and logarithmic IDF.

4.2.3 Applying K-Means Algorithm

The initial Centroids are selected randomly. K-Means algorithm is applied on those documents. In each of the iterations, the similarity of each document with every Centroid is calculated. A document is assigned to all the clusters which are “close enough”. This is done as a document typically has multiple topics. The stopping criterion is based on the number of iterations and number of documents switching Centroids.

4.2.3 Document Similarity

The document similarity is based on two different set of parameters. One is based on TF-IDF values calculated in the preprocessing step. The other is the set of document attributes which determines topic similarity : Common categories, Common templates, Common images, Common “See also links” ,Common links in Wikipedia pages. So to be considered similar two documents must exceed the threshold for both set of parameters. If a document is relevant to multiple Centroids, it is assigned to each of those clusters. If a document is not close enough – i.e. if the document similarity is less than a threshold then it is assigned to a *Miscellaneous* cluster.

4.2.3 Post Processing of Results

Once the list of clusters and their constituent documents are obtained, they must be post processed to make results more user friendly.

One of the steps is to rank the clusters based on its size and its rank. The rank of the cluster is calculated by the summation of difference between each document’s rank and total number of documents selected. The result is then normalized .For a cluster n,

Cluster Rank = $\sum (N - R_i)/ C_n$

Each cluster in the search result is sorted based on its rank. Each document within a cluster is sorted based on its search result ranking.

4.2.4 Cluster Labeling

In this subsection, the label generation logic for the clustering search engine is described.

Each of the clusters must be labeled according to the central concept of it. Selecting most commonly occurring word does not work always as, the query term is usually the most occurring word and is not the best label. For example in a query based on Obama, it is not good to have several clusters labeled Obama. Again applying the same algorithm for all words other the query word is also good because the resulting cluster labels feel incomplete. Continuing the Obama example, a cluster label title of Senator or President Elect is also not perfect. The user will expect a label of senator Obama or President Elect Obama.

A better method is to use a Phrase based labeling. Each of the document content is passed to POS tagger and the nouns in it are obtained. For each of the nouns, we create a set of 1-4 n-grams both before and after the noun. If one of the words contains a stop word, additional words are added to the phrase.

Each of the list of phrases are ranked based on their occurrence in the cluster. A good label must be very common within a cluster but not so very common in the entire cluster. The frequency count of a phrase is not done based on the entire phase but by the individual words that make it. For example a cluster label *Closures in computer science* is evaluated by the occurrence of each of the words – closures, computer and science. One good side effect of this method is that longer phrase labels are automatically pushed higher in rank. Each of the cluster is labeled by its top ranked phrase while also ensuring that the label is not already selected.

After all these steps, the result is displayed to the user for his consumption.

5. REFINEMENTS

Although the basic functionality provided by above algorithm works , there are multiple refinements possible to make it run faster and produce better results. This section will list of some of those refinements.

5.1 Selecting a Search Engine

Since this project is on clustering and not ranking , it utilizes the results of an alternate search engine for its top-n results. Multiple search engines were explored. The Wikipedia search api uses a very naïve keyword matching algorithm and does not ranks authorities higher.

Google search API gives excellent results , but has a limit for 10 results in each API call. Calling API 10 or 15 times increases the total time taken for clustering. Yahoo API has a simple and flexible search API and it was ultimately selected for clustering. It has a limit of 100 results per API call which suits the project fine.

5.2 Handling New Articles

Wikipedia is a constantly evolving website with many new articles added every day. So it will be unfair to the user to show only the old pre processes results. If the system finds that it does not have the article returned by the search result , it spawns a new thread to get the Wikipedia article content as xml feed via [7]. It is parsed and converted to a usual document feed format and then it participates in the clustering algorithm.

5.3 Augmenting Search Results

The search results obtained are cleansed of irrelevant articles as discussed in section 4.2.2. This results in lesser than 100 articles for clustering. So to make up for the shortage, the system selects the top 150 articles, so that the number of articles even after cleansing remains sufficiently high.

5.4 Better Heuristics for selecting Initial Centroids

One of the most sensitive parameters for K-Means algorithm is the initial Centroids. The following heuristic is used to select good initial points. 40 % of Centroids are selected from the top 20% results. And the other Centroids are selected from next 70% of results. No Centroids are selected from bottom 10% .

The rationale for the choices are as follows : If a search query has n concepts associated with it, then the most common of them will be bubbled up in the search results via the search engine. No documents are selected in the bottom 10% to avoid the negative effect of outliers.

The Yahoo search API also returns the size of the document as a result parameter. It is used in the selection of initial Centroids. The rationale is that key documents in a cluster are typically large. And the short articles are typically stubs or disambiguation or some article that talks about a very narrow topic.

5.5 Maintaining Optimal Clusters

In each of the iterations, all the clusters are evaluated. If a cluster has more than 30 % of the articles as its member it is split to two clusters. The rationale is that probably the cluster is about a very generic topic and it is beneficial to split it into clusters talking about specialized topics.

The system also merges clusters with less than 3% of membership with the nearest cluster . If none exists, it will be merged with *Miscellaneous* cluster. The rationale is that either the topic of the cluster is too narrow due to wrong Centroid selection or we selected an outlier as an Centroid.

5.5 Handling Miscellaneous Cluster

Once the K-Means algorithm completes, we are left with some clusters and a special cluster – *Miscellaneous* containing all the documents which did not fit into other clusters. Typically the documents talk about a variety of topics.

Instead of giving an un assorted cluster, the system breaks the *Miscellaneous* cluster into small groups of 2 or 3 articles which talk about same or similar topic and that topic is used as the cluster label. The rationale here is that it is better to split the cluster and give many small and narrow clusters instead of giving an giant , un assorted list.

5.6 Handling Vector Space Size

The complexity of K-Means is dependent upon number of iterations, number of clusters, number of documents and the size of the vector space. Only vector space size is amenable to optimization.

A simple optimization is stripping stop words and stemming from the documents. This typically leads to 30 % reduction of word size. Another idea was not to add any word that occurs less than 2 times in each document. This results in another 20% reduction.

A much larger reduction is obtained by ignoring template expansion but using it as a high weight document similarity factor. This yields another 20 % reduction.

Another heuristic that is valid if a document is large is to use the list of links in the document as a valid abstract of the document. The cumulative result is that for a big article we get a 80% reduction of vector size. For a small article , a reduction of 60% is usually obtained.

6. EXPERIMENTS

This section discusses the experimental setup and the various evaluations done based on the results on the current system and three other clustering search engine.

6.1 Experimental Setup

Wikipedia does not provide any details on the most frequently searched words. Instead they provide only statistics like most visited pages and most edited pages. Due to this reason, I had to use the following as test cases for my clustering engine. I am using a unique combination of words from four sources : Google zeitgeist 2007 [8] , Google trends [9] , Wikipedia statistics [10], WikiRage [11] .This set of words is used as a test set as this best approximates the way users search.

6.2 Experimental Results

The top 200 search results from the above list were used for evaluating the project. I had distributed the project to my friends to evaluate it. I am including here the results of clustering for two common query terms :

Polysemes : The classical touchstone for clustering algorithms had been the search query Jaguar.

The results of this query in this system are : Jaguar Racing , Jaguar Mark , V Engine , Daimler Sovereign , Fender Jaguar , Atari Jaguar , Jaguar S-type ,Panthera Onca , Hybrids , Archie Comics

The results of this query in clusty are : Games, Atari Jaguar , Racing, Formula One, Engine , Jaguar Cars Production, Lion, Leopard , Air Force, Squadron ,Automobile manufacturer

In exact queries : McCain is considered as a sample query for this case because almost always user expected some specific concept like his senate life or his presidential campaign.

The results for this query in my search engine is as follows : Talk John Mccain , Presidential Campaign , Early Life and Military Career , Climate Stewardship Act, Mccain Palin Campaign ,Endorsements Mccain , Republican Party Presidential Primary , Mccain Democratic , Joe Mccain , Republican National Convention , Cindy Hensley Mccain, Father John S. , President , Debates Wikipedia , Social Security , Template John Mccain and Foods.

In Clusty the results are Presidential, Election, USS, John S. McCain, Edwin McCain , Act, McCain-Feingold, Foods, Canada, GOP Senator

John McCain, New Brunswick, McCain Detainee Amendment, Committee, USA Senate.

We can see that the clusters generated by my project has more intuitive labels and better search results. Also the documents in each cluster has a higher affinity to the concept than that of Clusty's.

Part of the reason for this better result is imposing a threshold for cluster membership and splitting catch all clusters. Another reason is the phrase based labels. Again the focus of the algorithm to minimize false negatives than false positives leads to better results.

7 CONCLUSION

In this project , I had designed a simple clustering engine with multiple heuristics to optimize the results. I had also introduced a phrase based labeling algorithm. The initial results are very promising although many more optimizations need to be done.

Future Work

Multiple areas of this project can be enhanced in a variety of ways. The first is a more formal way to evaluate the results of the algorithm. A simple way to analyze is to use Amazon's Mechanical Turk. Other improvements include providing an API for the search results, evaluating the use of LSI in improving result analysis, evaluating co-occurrence matrices .

More involved changes include : a) Using multiple search engines to obtain the top-N results instead of relying on a single search engine. This will enrich the cluster results and also avoid bad clustering results. b) Augmenting search results using similar words in Wiktionary when the search results for the original query terms is less than some threshold.

Acknowledgements

I would like to thank Dr.Chengkai Li for providing me with an opportunity to work on this exciting project. I also thank my friends who gave lot of good suggestions to improve this project. Special thanks to Wikipedia folks for not only creating the best encyclopedia but also have made it available for data mining analysis. I also thank the many open source projects like Apache Commons , Apache Lucene, Stanford POS tagger and Cern Colt library.

References

[1] Zamir O., Etzioni O. Web Document Clustering: A Feasibility Demonstration, Proceedings of the 19th International ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'98), 46-54, 1998.

[2] Leouski A. V. and Croft W. B. An Evaluation of Techniques for Clustering Search Results. Technical Report IR-76, Department of Computer Science, University of Massachusetts, Amherst, 1996.

[3] Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma. Learning To Cluster Search Results. The 27th Annual International ACM SIGIR Conference (SIGIR'2004), July 2004

[4] Jerzy Stefanowski, Dawid Weiss . Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition [http://www.cs.put.poznan.pl/dweiss/site/publications/download/iipwm-osinski-weiss-stefanowski-2004-lingo.pdf]

[5] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

[6] XML dump oof English Wikipedia content : http://download.wikimedia.org/enwiki/latest/

[7] Url to export a Wikipedia article to XML : http://en.wikipedia.org/wiki/Special:Export

[8] Google Zeitgeist for year 2007 : http://www.google.com/intl/en/press/zeitgeist2007/index.html

[9] Google hot trends for recent times : http://www.google.com/trends/hottrends

[10] Statistics about Wikipedia : http://wikistics.falsikon.de/2008/wikipedia/en/

[11] Most common edited Wiki articles : http://www.wikirage.com/