Google Search Result Clustering Using Suffix Tree Clustering Algorithm

Fatma Gülşah Kandemir

Computer Engineering Department,
Middle East Technical University,
06531, Ankara/Turkey
e1448802@ceng.metu.edu.tr

Abstract

As the information available on the web increase inevitably and the technologies in the web domain improve fast, user of web become more and more demanding about gathering correct information they are looking for. Current search engines like Google and Bing try to meet the demand of web searching of users. But, for example Google presents search results in an unordered way, which makes users to sift through lots of irrelevant documents. This is waste of time for users, especially the ones whose search query words include synonyms.

Therefore, clustering of web search results is essential to make web users find what they are exactly looking for. This paper tries to explain one of the methods used for clustering web documents which is *Suffix Tree Clustering (STC) algorithm.* This algorithm makes clustering based on the *snippets* which are small definitions of web documents included in the search results. Snippets are widely used in web document clustering systems because they yield to good results in most of the methods.

Introduction

Today's search engines like Google, Bing and AltaVista show search results as a list of plain documents which is hard for users to find relevant documents of their choice. The aim of this work is to make search results easy to browse for users. This can be done by clustering the search results according to the topics they are sharing.

Early clustering algorithms were relying on the offline data gathered from the search engines. But today's users are not so patient to get results even in a minute. Since technology is speeding up, clustering must be done very quickly.

Zamir et. al. in [1] had come up with several key requirements for web document clustering systems:

1. **Relevance:** The documents should be grouped correctly, namely the user should see relevant results together.
2. **Ease-of-browsing:** The users need to understand the content of a cluster at a glance.
3. **Overlap:** Some documents may occur in more than one cluster, since topic may overlap.
4. **Snippet tolerance:** The method should produce *reasonable* results even only by accessing ***snippets*** of documents. Because downloading the whole documents is very time-consuming.
5. **Speed:** The method should return results in seconds.

To accomplish these requirements [1] had proposed *Suffix Tree Clustering – STC* algorithm. STC uses suffix trees to identify sets of documents that share common phrases. The detailed method will be explained later in this document, but to briefly summarize the method, the steps will be like as follows:

1. Retrieve the search result snippets
2. Clean the snippets
3. Construct suffix tree
4. Identify base clusters
5. Combine base clusters to find final clusters
6. Show the final clusters and their documents to user.

I used Google as the search engine, because Google has the widest resource in the web and is the most common used engine nowadays.

This document will give information about the background of web document clustering concept, then explain the STC method in detail, after showing some results and comparisons with today's technology conclude the work done.

Related Work

In middle 80s document clustering algorithms appeared to be in literature. The most popular algorithm used then was Agglomerative Hierarchical Clustering (AHC). There were other algorithms used like Single Link *O(n2),* Group avarage *O(n2)* and Complete-link *O(n3),* but as *Etzioni&Zamirs'* experiments [1] show they are very slow compared to STC method.

Also there are linear time algorithms like K-means O(nkt) and the Single-Pass method O(nK).They are good in speed and comparable in final clusters. But they are treating documents as words not as a sequence of words so they lack sharing of the common phrases.

After 2000s hierarchical clustering methods are proposed and implemented. Some of these methods use only words for classifying, some of them use phrases. Thus, there have been 2 different approaches on the type of resulting clusters: flat vs. hierarchical and 2 different approaches on type of implementation: single-word vs. phrases. Table 1 shows a table of works based on these four approaches. Most of these approaches use snippets as the source for clustering except some of them.

	Flat Clustering	Hierarchical Clustering
Single word	Retriever, Scatter/Gather, WebCat	FIHC, CREDO
Sentence	Grouper, Lingo	SHOC, SnakeT, Tumba, WISE

Table 1. State of art clustering approaches.

Single words and flat clustering: RETRIEVER [3] uses a relational fuzzy clustering algorithm to flatly cluster web search results. SCATTER/GATHER [4] was one of the first web-clustering software but it was never tested in real web environment. It was using K-means algorithm like WEBCAT.

Sentences and flat clustering: GROUPER[2] is the work of *Etzioni&Zamir* and uses STC which I used in implementing my project. The original Grouper is no more available but an open source implementation of it, CARROT2[5], is available on web. Another work done using sentences, namely phrases in clustering is LINGO [6]. It uses Single Value Decomposition (SVD), and it is considerably slow when clustering large number of snippets.

Single-words and hierarchical clustering: FIHC [7] based their solution to construct hierarchical clusters, on the Frequent Itemsets Problem, whereas CREDO [8] uses a concept based on lattice on single words.

Sentences and hierarchical clustering: SHOC[9] uses the Suffix Arrays as in STC algorithm and achieves a hierarchical clustering using SVD approach. SNAKET [10] offers both hierarchical clustering and folder labeling with variable-length sentences. Again Tumba[11] uses STC algorithm but this time hierarchical clustering is done by subsumption algorithm developed by [20]. WISE [12] is another hierarchical clustering system but this time whole documents are used instead of documents. Wise is based on extraction of keyphrases in documents and then PoBOC algorithm is used to obtain hierarchy of clusters.

The commercial tool CLUSTY (*www.clusty.com*) product of VIVISIMO and YAHOO Search (*search.yahoo.com*) are in the final group and most of the techniques in this group try to achieve their performance.

The Method – Suffix Tree Clustering

I used the Suffix Tree Clustering (STC) algorithm, in my method, as proposed by *Zamir&Etzioni* in [1]. The STC algorithm is based on finding common phrases shared between web document snippets. This suffix tree algorithm used in STC method treats snippets as strings and words as suffixes, so the final suffix tree will be consisting of series of suffixes, in our case series of suffixes are phrases, and the documents including those suffixes.

The overall method has three main steps: first; document retrieval and cleaning, second; identifying base clusters, and third; constructing final clusters from base clusters identified in the second step.

Document Retrieval and Cleaning

In this step, the search query is sent to Google and resulting snippets are gathered using java html parser. Html parser cleans document from html tags. After that, punctuations and numbers namely all non-alphabetic characters are removed from the snippets.

Then the words are stemmed from their surface representations using the Snowball [13] stemmer library which is an implementation of Porter's Stemmer [14].

Document cleaning is important, because words having same origin may occur in documents in different surfaces, and this will disable the same originated words/phrases to be identified.

Identifying Base Clusters

Identification of base clusters is done in 2 steps. First step is constructing the suffix tree from all snippets. Second step is traversing down the suffix tree to find the base clusters.

Using suffix trees in document clustering is first introduced by *Zamir et. al.* in 97. Suffix tree is a data structure that presents the suffixes of given string. The suffix tree of a string *S* is a tree whose edges are labeled with strings such that each suffix of *S* corresponds to exactly one path from the tree's root to leaf. This maintains the uniqueness of phrases. As stated before, in this method *S* corresponds to the snippets, suffixes correspond to the words and phrases.

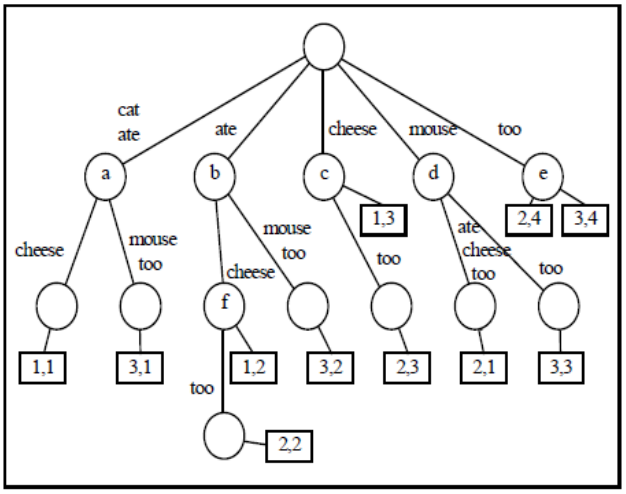

Figure 1. The suffix tree of the strings "cat ate cheese", "mouse ate cheese too", and "cat ate mouse too".

Here, the input contains a collection of snippets, so the suffix tree consists of all words or word groups contained in the whole collection. Each node keeps the information about the words/phrases it contains and the documents that the words/phrases are included in.

Figure 1 represents an example suffix tree constructed using "cat ate cheese", "mouse ate cheese too", "cat ate mouse too" set of strings. Nodes are represented with circles and the boxes attached to the nodes keep the information about the nodes' suffix and snippet information. The first number in the box represents the document id, in our case snippet id, and the second number represents the starting location of the suffix namely the phrase.

In the proposed implementation each node corresponds to a set of documents and the common phrase that is shared in those documents.

The base clusters are identified just after constructing the suffix tree of snippets. Each node having at least 2 documents related to it (the descendants and their related documents counted in the node also) composes a document group, in other words a base cluster. The name of the base cluster is the suffix that the node is labeled with, and also this label is the common phrase used in the document group.

Figure 2 shows the list of documents and phrases that

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

Figure 2. Base clusters of same strings.

are selected to compose the base clusters. In [1] they say they keep a list of internet specific words such as "mail, previous, java etc." and eliminate the base clusters having those stop words as labels. But I thought that eliminating these words from the snippets as early as possible, like when doing the document cleaning, would be more effective. Also they proposed that the more frequent a phrase is used in all documents the less it is likely to be a distinctive common phrase. Being stick to this concept, I added the *search query* to the stop word list. Therefore the final stop word list consists of:

1. Internet specific words
2. Language specific words (English)
3. Query word/phrase

Constructing Final Clusters

Final clusters are determined by combining the base clusters. The idea here is, combining the base clusters which share common documents. For each base cluster pair a similarity function is calculated. Let **A** and **B** be the base clusters that the similarity is to be calculated, |**A**| and |**B**| be the number of documents in these clusters, and the |**A**∩**B**| be the number of documents shared in these clusters, then the function of similarity would return true in the following condition:

$$|\boldsymbol{A} \cap \boldsymbol{B}| \geq \frac{|\boldsymbol{A}|}{\mathbf{2}} \; \boldsymbol{and} \; |\boldsymbol{A} \cap \boldsymbol{B}| \geq \frac{|\boldsymbol{B}|}{\mathbf{2}}$$

A graph table is constructed while checking the above condition. The nodes are the base clusters and there will be an edge between two nodes when the similarity function of the nodes return true. The connected components in this graph, forms a final cluster. For example Figure 3 shows the base cluster graph of the "cat ate cheese", "mouse ate

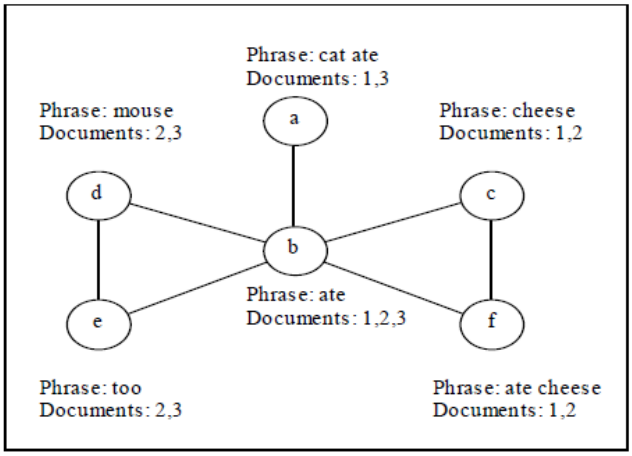

Figure 3. Final cluster graph of base clusters in Figure 2.

cheese too", "cat ate mouse too" set of strings.

This graph shows that all base clusters form one final cluster, namely they are all related to each other. Notice

Query: *Salsa*		Query: *Asus*		Query: *Art*	
Clusty Results	**My Results**	**Clusty Results**	**My Results**	**Clusty Results**	**My Results**
salsa dance (28)	danc (38)	eee pc (31)	pc, eee, eee pc (29)	art gallery (15)	artist (23)
club (17)	music, latin (22)	motherboard (19)	review (14)	museum (15)	museum (18)
pictures (15)	event (17)	downloads, driver (12)	netbook (11)	art prints (9)	galleri (17)
recipes (12)	danc lesson, lesson (11)	asustek computer inc. (6)	intel (10)	history, school (9)	paint, paint sculptur, sculptur (17)
definition (5)	photo, video (10)	linux (10)	laptop (10)	photography (9)	fine (12)
style, cuban (6)	club (9)	asus laptop (7)	comput (9)	visual (9)	collect (12)
texas (4)	guid, news, instructor (9)	arizona state university (6)	motherboard (9)	community arts (7)	exhibit, event (12)
musical genre (2)	recip (8)	computers (4)	intel atom n * (8)	exhibits (7)	feature (11)
meetup (3)	sauc, tomato, cilantro, chop, onion, pepper (7)			sculpture (6)	poster, print, satisfact guarantee, guarantee, fast (10)

Table 2. Some result comparisons between Clusty.com and my work. Notice that, even though they are not exactly same, in essence, they have similar logical clusters.

that, if "ate" was in the stop word list, there would be 3 final clusters formed.

Depth-First Search algorithm is used here when identifying the connected components. A final cluster's document list is the union of the documents that base clusters have.

After connecting base clusters and finding final clusters, the final clusters are ranked according to the number of documents they have and they are presented to the user in that order.

Results

A sample result page is shown in Figure 4, at the end of the document. The results are formed in real time and shown to user in at most a second. Thus the algorithm can be said to meet the users' speed expectations. The most crowded cluster is shown at the top, and the document snippets are listed below the name of the cluster.

Since there is no ground truth results of clustered documents in the web, I compared my results with the most widely used and trusted product Clusty, www.clusty.com.

First I, compared the top most level clusters' labels for the queries: "salsa", "art" and "asus". Table 2 shows the resulting labels of final clusters. The results mostly overlap with each other semantically.

	Top 5 Clusters	Top 10 Clusters	Top 15 Clusters
Average Precision	3.12	5.37	6.37

Table 3. The average precisions which are calculated by comparing top 10 Clusty.com clusters with top 5, top 10 and top 15 clusters of my results.

Second experiment conducted is, comparing the labels of clusters in the order of top 5, top 10 and top 15. The resulting table of this work is showed in Table 3. As the number of compared clusters from top to bottom increase, the number of clusters overlapping increases.

Conclusion

STC is an effective algorithm in clustering web search results. It uses the ease of snippets, so the program runs in real time and fast. It relevantly clusters the search results and the final clusters' names are informative, the user can understand the content of a cluster at a glance.

References

[1] O. Zamir and O. Etzioni, "Web document clustering: a feasibility demonstration," in *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 1998, pp. 46-54

[2] O. Zamir and O. Etzioni, "Grouper: a dynamic clustering interface to Web search results," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 31, no. 11-16, pp. 1361-1374, 1999.

[3] Z. Jiang, A. Joshi, R. Krishnapuram, and L. Yi. Retriever: Improving web search engine results using clustering. In Managing Business with Electronic Commerce 02.

[4] M. Hearts, and J. Pedersen, "Re-examining the Cluster Hypothesis: Scatter/Gather on Retrieval

Reuslts", in *Proc. of the 19th Annual International SIGIR Conf.*, Zurich, Switzerland, 1996.

[5] D. Weiss and J. Stefanowski. Web search results clustering in polish: Experimental evaluation of carrot. In IIS03.

[6] S. Osinski and D. Weiss. Conceptual clustering using lingo algorithm: Evaluation on open directory project data. In IIPWM04, 2004.

[7] B. Fung, K. Wang, and M. Ester. Large hierarchical document clustering using frequent itemsets. In $SDM03.

[8] C. Carpineto and G. Romano. Concept Data Analysis: Theory and Applications. John Wiley & Sons, 2004.

[9] D. Zhang and Y. Dong. Semantic, hierarchical, online clustering of web search results. In WIDM01.

[10] P. Ferragina and A. Gulli, "A personalized search engine based on web-snippet hierarchical clustering," *Software: Practice and Experience*, vol. 38, no. 2, pp. 189-225, 2008.

[11] B. Martins, and M. Silva, "Web Information Retrieval with Result Set Clustering", in *Proc. of Natural Language and Text Retrieval Workshop,* 2003.

[12] Campos, R., Dias, G., Nunes, C. and Nonchev, B., 2008. Clustering Web Page Search Results: A Full Text Based Approach. In International Journal of Computer and Information Science Vol 9(4). pp 29-40.

[13] Snowball, http://snowball.tartarus.org/

[14] Porter, http://tartarus.org/~martin/PorterStemmer/

Figure 4. A sample output page from my work, for query word *allergy*.