Cluster Algorithm for Search Engine Collator

Dixit Rashmi K.
Assistant Professor
Department of Computer Science & Engineering
Walchand Institute of Technology, Solapur, Maharastra, India

Abstract - Users of Web search engines are often forced to sift through the long ordered list of document "snippets" returned by the engines. The IR community has explored document clustering as an alternative method of organizing retrieval results, but clustering has yet to be deployed on most major search engines. Our project aims to provide an organized search result, which will help the users to map into their intended results. This minimizes the irrelevant results and for achieving this, the implementation of core clustering engine is a very efficient approach.

The core Clustering Engine technology is called document clustering, which is the automatic organization of documents into spontaneous meaningful groups. Document clustering methods never need to touch or know about the larger collection from which search results are taken, or undergo any other pre-processing steps. Organizing the search results occurs just before a user is shown the long list of search results.

The final output is a hierarchy (or tree) on the left of a split screen with the search results on the right. A search result is not constrained to fit within a singletree location, since individual search results could reflect many themes

Keywords: Tagging, Clustering, STC, Algorithm,

I. Introduction

Clustering products help enterprises organize information from anywhere, any time, in any language without the endless cost and complexity of building information taxonomy. This technology is on display for web searching and for searching corporate, government, news, etc. Clustering Engine does not crawl or index a document collection. It organizes the outputs of other search engines: URLs, titles, summaries, and meta-data if available and desired.

Data clustering is a common technique for statistical data analysis. Clustering provides partitioning of a dataset into subsets of similar objects or data clusters. Before actually using a clustering technique the first task one has to do is to transform the problem at hand into a numeric representation that can be used by clustering algorithms.

In our case, the goal is first to provide a similarity measure among keywords and then to run clustering techniques on the keyword space represented like this. The key in building an effective exploration space seems to be able to group and show related items

II. Motivations

The success of tagging services like 1Flickr[1], del.icio.us[2] and technorati[3] has shown that tagging is a great collaboration tool. Tagging seems to be the natural way for people to classify objects as well as an attractive way to discover new material. Tagging services provides users with a repository of tagged resources (tag space) that can be searched and explored in different ways. People tag pictures, videos, and other resources with a couple of keywords however; looking for information in the tag space has a number of hard limitations. The difficulty comes from the fact that several people usually use different tags for the same document. In fact, even a single user `s tagging practice may vary over time. Usually, this variability is compensated by looking at many users tags ; which is only possible when the page has been tag many times. However for less popular pages the problem remains .The majority of today's Web search engines (e.g., Excite, AltaVista) returns a very large list.

Let us imagine that you would like to tag the word "".A person not aware of this classification of'' would tag this word with any combination of the different tags. There is problem is rooted in the language ,words are often related and do not stand in isolation .Such relation among word are often called lexical isolation .

Document clustering algorithms attempt to group documents together based on their similarities. Document clustering can be performed, in advance, on the collection as a whole, but post-retrieval document clustering has been shown to produce superior results. This is due to the fact that the clusters are computed based on the returned document set; the cluster boundaries are drawn to appropriately partition the set of documents at hand. In contrast, pre-retrieval clusters might be based on features that are infrequent in the retrieved set, as many non -retrieved documents influence the cluster formation.

[1] http://www.flicker.com, now part of Yahoo!
[2] http ://www.del.icio.us, now part of Yahoo!
[3] http://www.technorati.com

One way to cluster Web search results is to do so on the search engine. But search engines are under severe resource constraints and dedicating enough CPU time to each query might not be feasible. The clusters, therefore, would have to be pre-computed. We assume the clustering interface is independent of the search engine, i.e., it can reside on a meta-search engine or on a client's browser.

- Coherent Clusters
- Efficiently Brows able
- Speed

In earlier work we introduced Suffix Tree Clustering (STC) – a fast, incremental, linear time clustering algorithm that produces coherent clusters. Using traditional IR metrics (e.g., average-precision) we compared STC to other fast clustering algorithms (including k-means, Buckshot and Fractionation) on the task of clustering results returned by Web search engines, and showed it to be both faster and more precise than previous algorithms. We used the results to argue that post-retrieval clustering of Web search engine results a-la-STC is feasible

In this paper, we present Grouper – a clustering interface, which uses STC as its clustering algorithm.

III.WORKING OF CLUSTERING

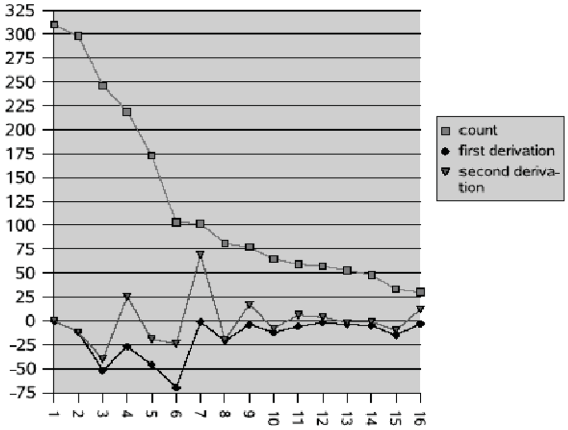
The algorithm is based on counting the number of co-occurrences (keywords that are used for the same page) of any pair of keywords and a cut-off point is determined to decide when the co-occurrence count is significant enough to be used. This results in a sparse matrix that represents keywords, so that the value of each element is the similarity of the two keywords. Say a user keywords an article about African trees that is written by an XHTML expert with the following keywords: xhtml, standard, trees, biology, Africa, to read, USA. Then (xhtml, standard) and (xhtml, trees) would each get one count as co- keywords. After processing the whole keyword space, we use the frequency counts of all the co-keyword pairs and attempt to identify the significant co-keywords. In order to do that, we determine the pairs of keywords that co-occur significantly more frequently then expected. We look for a cut-off point above which the co-keywords are considered strongly related. Let's look for example at the keywords related to keyword RSS

Table 1: Keyword “RSS”and their count

Keyword	Count	Keyword	Count
Feed	310	Web2.0	77
Blog	298	Home	65
Feeds	246	Wikipaedia	59
Search	219	Blogs	57
News	173	Biography	53
Google	103	Preview	48

In the table and the graph, we see that "feed" occurred 310 times together with RSS. In the graph, the y-axis is the count i (how many times a certain keyword is used together with RSS), as well as its 1st and 2nd derivative, the x-axis is i. In the mentioned example about African trees, the keyword combination xhtml, Africa is just accidental and related to this example and thus would not be selected.

Figure : Graph

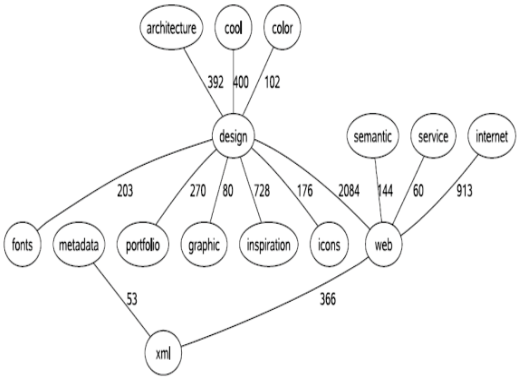


We see a clear change in the plot for the count i, and to determine this cutoff point, we consider the 1st and 2nd derivative of the count.

Once we get the cut-off it means that the keywords having the hit value more than the cut-off value are eligible for clustering and are strongly related to the keyword "RSS".

If we do this for every keyword in the keyword space we obtained an undirected graph G (V, E, W) consisting of nodes V, a set of edges E and a weight matrix W. Each vertex vi of the graph corresponds to a keyword keywordi. There is an edge between vi and vj if the keyword keywordi relates strongly to keyword keywordj or vice versa according to the described algorithm. The weight wi1i2 corresponds to the number of times keywordi1 occurred together with keywordi2 within the same item.

Figure: part of cluster design



The clusters were computed regularly and result seems to be fairly stable, that is there was no tendency towards one big cluster. Some cluster seem to big, i.e. the cluster above should split into a \design and \web

IV.CLUSTERING ALGORITHM

The input for the keywords clustering algorithm consists of:
Keywords ti, i = 1 ….I
Users uj, j = 1…J
Keyword resources (web resources were used in our experiments) rk, k = 1… K

A 3D tensor A belongs to R^(I*J*K) of Boolean values. The tensor A contains keyword information: if a user ui got a keyword from a resource rk with a keyword kj then Aijk = 1, otherwise Aijk = 0.

Our goal is to partition the set of keywords into non-intersecting groups of semantically related keywords. We show here how a graph is built from the input of the keyword-clustering algorithm. Let G (V, E, W) be an undirected weighted graph consisting of nodes V, the set of edges E, and a symmetric weight matrix W belongs to R^(I*I), where I is the number of vertices. Each vertex vi of the graph G corresponds to a keyword ti. We compute matrix B belonging to R^(I*K), collecting the keyword information from all users. The rows of the matrix B correspond to the keywords, while the columns of the matrix B correspond to the keyword resources. Thus, if a resource rk has a keyword ti by some user, then Bik = 1.

The weight wi1i2 of the edge between the vertex vi1, corresponding to the keyword ti1 and the vertex vi2, corresponding to the keyword ti2 is the number of resources, key worded by both keywords ti1 and ti2. Thus we take the rows i1 and i2 of B, and calculate the number of resources shared by the keywords ti1 and ti2: wi1i2 = Bi1 (logical AND) Bi2.

V.Advantage: Making the Clusters Easy to Browse

As mentioned in the introduction, it is not enough for a clustering system to create coherent clusters, but the system must also convey the contents of the clusters to the users concisely and accurately. The system is most useful when the user can decide at a glance whether the contents of a cluster are of interest.

One approach of describing a cluster includes presenting words that appear frequently in the documents of the cluster (the cluster's centroid) as well as presenting the titles of selected documents. As STC creates clusters based on phrases that are shared by many of their documents, Grouper can also use these phrases to describe the cluster. We have found these phrases to be extremely useful in characterizing the cluster's content.

Given a set of phrases that define an STC cluster we present them in descending order of coverage (the percent of documents in the cluster that contain the phrase) and length (number of words in the phrase, not counting stopped words nor words appearing in the query). Since each cluster can contain many phrases, we display at most the first six phrases because our goal is to create a compact cluster summary. We have found that in many cases some of the displayed phrases are quite similar and therefore redundant. This reduces the conciseness and clarity of the summary, as well as prevents additional informative phrases from being displayed. Such redundant phrases are therefore not displayed.

There are three heuristics we use to identify redundant phrases:

1. *Word Overlap:*

If a phrase has more than 60% of its non-stopped words appearing in another phrase of higher coverage, it will not be displayed. In Table 1, phrase 7 is not displayed as 75% of its words appear also in phrase 6, which also had a higher coverage.

2. *Sub- and Super- Strings*:

Often, we find that STC identifies phrases that are sub-strings of other phrases (we call these sub-phrases and super-phrases). This happens both when the two phrases are independent phrases in there own right (such as "united states" and "president of the united states") and when the phrase was truncated in the snippet received from the search engine (such as "president of the united states" and "president of the united…"). A sub-phrase will always have a higher coverage than its super-phrase, but the super-phrase will be more specific and therefore more informative. To balance this tradeoff, we determine, for each phrase, whether it has a sub-phrase or a super-phrase among the other phrases of the cluster. If it doesn't have a sub-phrase, it is designated as a most-general phrase; if no super-phrase exists, it is designated as a most-specific phrase. We will consider phrases redundant if they are not in either of these categories

3. *Most-General Phrase with Low Coverage*

The purpose of displaying short, general phrases is to achieve high coverage. In cases were a most-general phrase adds little to the coverage of its corresponding most-specific phrase, we will not display it. Thus, we define a minimal coverage difference (we found 20% to be satisfactory) that must be attained in order for the most-general phrase to be displayed. As mentioned earlier, documents are processed before they are inserted into the suffix tree (step 1 of STC). This processing improves the identification of common phrases, however the readability of the text is reduced. To deal with this problem we keep the original (unprocessed) text of the documents; instead of displaying a processed, and less comprehensible, phrase, we display the original text that corresponds to it. For example the phrase "post retrieve document cluster" might actually correspond to the string "post-retrieval document clustering", which might be more meaningful to the user.

In addition to describing the cluster using its phrases, we also use the standard technique of identifying and presenting single words that appear frequently within the documents of the cluster. We display up to five such words for each cluster. Words will not be shown if they appear in less than 40% of the documents, or in the stop list, or any of the displayed phrases. A semi-colon separates these words from the phrases of the cluster. There is one major difference between the phrases and these frequent words that should be emphasized. A phrase indicates that all the documents containing it are in the cluster, because it was a basic blocks on which the cluster was constructed. Documents containing a frequent word, on the other hand, might not all appear in the cluster.

Another design decision in Grouper was how to order the documents in a cluster. This affects both the order of the documents on the cluster's page, and the choice of the three documents that have their titles displayed in the cluster's summary on the main results page. Two options were considered: sorting the documents based on their relevance to the query or sorting them based on the number of the cluster's phrases each contains (this is similar to the option of sorting the documents of a cluster based on their similarity to the query or based on their similarity to the cluster's centroid, a). We chose the second approach – sorting by the number of the cluster's phrases each document contains – as we believe this result in more informative cluster summaries.

VI.USING CLUSTERS TO FIND SEMANTICALLY RELATED TAGS

Related tags can also help the user by suggesting interesting tags while tagging, searching, exploring or subscribing. For example a user subscribing to the tag music would be suggested to try the also add the tag mp3 to the subscription. Here the technique to automatically discover related tag based on the cluster.

The algorithm works as follows:

- For each tag ti that is frequent enough in the tagspace:
- Build a graph
- Partition a graph to different number clusters using the algorithm.
- Increase the similarity count for each pair of tag tj tk
- Sort all the pairs of tags
- Select the top N similar tags

VII. COMPARISONS

Different methods for clustering

- *K-means*

It is direct technique, the desired no of clustering are given as an input. Cluster is represented as geometrical center of gravity in the given subset. The main drawback is the complexity of K-mean in single iteration is O(N2).

- *PAM*

Random agents are selected and assigned to one of the initial medoids. Next the cost function of the current configuration is calculated. The main drawbacks are Complexity and high branching factor

VIII. CONCLUSIONS

With the help of this we intend to show that by clustering the results, obtained from a search engine, an ample amount of time and complexity is reduced. Also the experimental results have shown that clustering a result isolates the irrelevant and unintended results i.e. a distinction is made in between the relevant and the irrelevant results.

We can point out additional benefits of clustering which are more difficult to quantify but are of clear value an enhanced user experience with gains for the image and reputation of the intranet site. Our model doesn't include the doubled timesavings when a user consumes the time of other employees in order to solve his information need. The ability for a user to learn about company operations by viewing the taxonomy (or hierarchy) that is returned as the output of every search. For example, doing a search on a specific corporate customer may turn up taxonomy of the interactions with that customer that are recorded in the intranet's document base. This easy mechanism of knowledge diffusion is probably the biggest benefit

REFERENCES

[1] K.Bielenberg and M.Zachera. Groups in software: Utilizing tagging to integrate individual context for social navigation .1005

[2] A.Pothen, H.D.Simon, and K.P.Liou. Partitioning sparse matrices with graphs

[3] http://www.flickr.com, now part of Yahoo

[4] www.vivisimo.com

[5] www.google.com

[6] Database System Concepts by Silberschatz, Korth & Sudarshan.