World Wide Web Metasearch Clustering Algorithm

Adina LIPAI
Academy of Economic Studies, Bucharest

As the storage capacity and the processing speed of search engine is growing to keep up with the constant expansion of the World Wide Web, the user is facing an increasing list of results for a given query. A simple query composed of common words sometimes have hundreds even thousands of results making it practically impossible for the user to verify all of them, in order to identify a particular site. Even when the list of results is presented to the user ordered by a rank, most of the time it is not sufficient support to help him identify the most relevant sites for his query. The concept of search result clustering was introduced as a solution to this situation. The process of clustering search results consists of building up thematically homogenous groups from the initial list results provided by classic search tools, and using up characteristics present within the initial results, without any kind of predefined categories.
***Keywords**: search results, clustering algorithm, World Wide Web search.*

Introduction

Trying to keep up with the continuous growth of World Wide Web (WWW) the searching tools are engaged in a permanent race for ever faster development in order to reach better performances.

In the initial stages the general trend of development was concentrated on bigger data bases, bigger document bases, which to store the web pages accordingly. When the document storage reached considerable sizes the problem of better indexation was addressed. The bigger the storage capacity it became the more performing the indexing algorithm had to be in order to keep the web pages properly ordered. But the WWW was still growing with increasingly speed, so the crawler module had to be developed to reach higher speed in finding and downloading new pages.

For many years it was believed that the bigger the data base of the search engine is, the more performing it will be. The more and more efficient crawler was downloading pages at a ever higher speed and proper indexer algorithm was constructing the permanently increasing document base. As a result, the modules were the focus of search engine development for many years. But when document bases reached billions and tens of billions of documents, and the crawlers were downloading new documents at a speed of hundreds, or even thousand a day, a new problem appeared. With such big quantity of pages, the indexer was retrieving and presenting to a user longer and longer list as result to a query. The simpler and more common the query is the more results will be returned, rendering the user unable to check all of them in order to identify the web pages that best fit his needs.

Thus another efficiency criterion was introduced: easy retrieval of the information needed within the results provided by the search tool. The “easy retrieval” is evaluated both from the speeds perspective (the fastest the user finds what he is looking for, the better) and from the relevance of the results (did he find exactly what he needs, or just almost).

A fist solution implemented to solve this problem was ordering the returned results in a list based on some sort of relevance criteria (the more relevant the result was, the higher in the list it would be displayed). Even so, the required result is sometimes hard to find because it is not in the first 20 – 50 results displayed. The algorithm for clustering search results presented in this paper address this issue.

Clustering web metasearch results

Clustering search tools results means *grouping* them into *object classes* which are

constructed *using the search results characteristics*, with the purpose of simplifying the users work to retrieve the information it needs, helping him to find faster better quality results. *Metasearch* means the use of more search tools simultaneously for the same query.

Organising search results in clusters is not meant to replace the classical way of presenting results in ranked lists ([Brin, 98]). Its purpose is to provide supplementary organisation for those results. The clustering method will provide a series of search results clusters, with the property that the pages inside one cluster are similar to each other, and the pages belonging to different clusters differ from one another. Inside each cluster the initial ranking order provided will be preserved.

To justify its implementation, a search result clustering module has to meet the following requests:

- It has to simplify the user's work in finding the results it needs in the search result list, at a faster speed;
- The module must not over crowd the search tool's resources;
- It must not over extend the time needed to find the proper web page (e.g. if Google offers results for a query in few milliseconds, an clustering process that will take tens of seconds will be unsatisfactory ([Leouski, 1996]).

In order to make the user's job more easy the clustering algorithm will divide the list of results in homogenous groups. This way, the user will have to identify the cluster in which his pages are most likely to be and only search through that cluster, ignoring the other clusters. In the image below we have represented schematically the basic principles of how clustering method operates. The initial results $R_1, R_2, ..., R_n$ form a list, which after clustering will become part of one cluster $C_1 \dots C_n$. The cluster can differ in size accordingly to what web pages meet the subject criteria of that cluster.

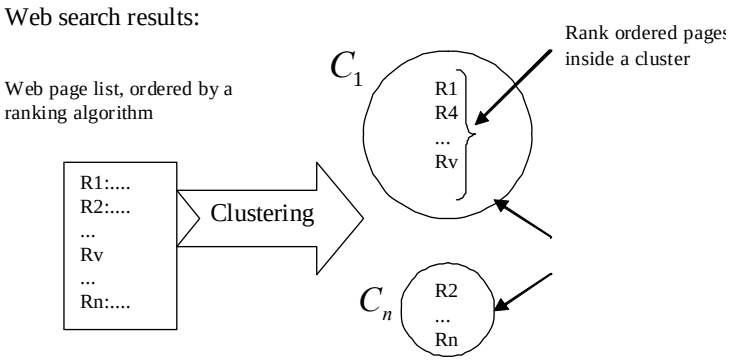

Fig.1. Web search result clustering

Steps in web search results clustering process

1. Obtaining the web page list
2. Document pre-processing
3. Transforming the documents into vector representations
4. K-means processing, with adequate adaptations for web pages
5. Constructing cluster like representation for final results

1. Obtaining the web page list

The initial web page list is obtained by reuniting all results from all search tools used in one centralised list, which will be used to cluster the results. In order to obtain the web page list the next steps have to be done:

a) **Elimination of multiple links:** because the results are obtained from more search tools it's more than likely the same site will be returned by more than one tool.

b) **Calculating the new rank:** the formula below will be used

$Rank_{new} = I_M * NrRez + I_{Rez}$, (1)

$Rank_{new}$: the new rank calculated for each web page

I_M : the index of the search engine which provided this page
NrRez : total number of results after the duplicates were eliminated
I_{Rez} : the index of the page in the I_M search tool

c) **Duplicate web pages** will receive the rank from the search tool with the lowest index

2. Document pre-processing

The clustering algorithm uses information from the pages in order to determine its subject or characteristics. Most document clustering algorithms use the whole document for this process, but such approach would slow our web page clustering algorithm to much. Therefore the algorithm will only use the snippets provided by the initial search tools. Previous work has showed that snippets provide good quality description of web pages, so justifying their use ([Leuski, 2000]). The title of the web page will also be used, if there is one. Title words will become then more important for classification than snippet words. Web pages processing is made up by the following operations:

a) **tag cleaning:** eliminating portions of the web document which are strictly related to text formatting

b) **lexical analyse:** the purpose of the analyse is to identify distinct words. The process implies eliminating useless characters such as comma, punctuation marks, sometimes numbers, or characters like: #, $...

c) **word root extraction:** the goal is to obtain an homogenous one-word description of similar but not identical words. The word obtain in the end doesn't have to have a meaning, nor be grammatically correct, but it will contain the description and similarity with all the other words that it represents. For example: "implementing", "implementation", "implemented" will be described by the root word "implement".

d) **stop word elimination:** a stop word is a word that does not have an informational value. In all languages there are a series of words which are considered stop words. For example: "on", "and", "the", "in", etc.

e) **establishing index words:** an index word, is a word that is representative in the context of the document.

In the pictures below we have represented the document pre-processing steps:

```
Course <- word
and <- stop word: eliminated during pre-processing
data <- word
mining <- word
next <- eliminated during pre-processing
knowledge <- index word
courses <- transformed into "course" root word
```

a) Lexical analyse

```
1. knowledge
2. Course
3. data
4. mining
```

b) Final list after all steps

Fig.2. Web pages pre-processing

3. Transforming the documents into vector representations

In order to use the k-means clustering algorithm we need to transform each document into a vector. The vectors will have the same size, turning our result list into an M*N matrix. Each line represents one web page and each column represents one word. The N dimension represents the total number of words that will be processed, from all documents. The dimension M represents the remaining number of web pages after the duplicate eliminations, the final search result list.

Supposing we have:

- M web pages: d_1, d_2,... , d_M and
- N indexed words from 1 to N

Than, one web page will be represented in vector space using the following formula:

$$d_i = [w_{i1}, w_{i2}, \ldots, w_{iM}] \quad (2)$$

where w_{iM} represents the weight of word j from document d_i

Example:

Table 1. Vector representation of web page

Document (title represented)	Word 1: automatic	Word 2: computer	Word 3: date	…..	Word N: analyse
d_1 = Data mining course	0.4	0.1	0.8	….	0.7
d_2 = Software development	0.01	0.9	0.3		.5
….					

The words 1..N are usually the index words, and they are obtained after the pre-processing the documents. Assigning each word a weight it's crucial in order to distinguish the more relevant words from the less important ones. In the end we will have quantified the importance that each word has for a given web page.

Transforming the documents into vector space representations requires the following steps:

I. Index word vector identification: it implies joining together in one vector all the index terms from all the documents.

II. Establishing the document vector matrix: for each document and each index term we will count the number of times one word appears in the same page.

For example, using as search query the words "data mining" we will have the next vector space representation for the results:

Table 2. Vector space representation of search results

Document / Words	course	Data	Mining	content	Discovery	knowledge	base	…	automatic	computer
d1	2	3	3	1	1	1	1		0	0
d2	0	2	2	0	0	0	0		1	1
d3	1	3	3	0	1	1	1		0	0

III. Weight word determination for each document

In order to calculate each words weight we will use the next formula:

$w_{ij} = fT_{ij} * \log(n / fd_j)$ (3) [Salton, 89]

w_{ij}: the weight of word t_j from d_i document

fT_{ij}: the frequency of word t_j in d_i document

fd_i: total number of web pages in which the word t_j is present.

Table 3. Word weight of vector documents

Document / Words	course	Data	Mining	content	Discovery	knowledge	base	…	Auto matic	computer
d1	0.176	0	0	0.477	0.176	0.176	0.176	…	0	0
d2	0	0	0	0	0	0	0	…	0.477	0.477
d3	0.176	0	0	0	0.176	0.176	0.176	…	0	0

The term $\log(n / fd_j)$ is also called *inverted document frequency*. The words that have a higher frequency in a page will most likely be a better description than the words that have low frequency. But also, the terms that have high frequency in all documents, will not make good description for differences that appear between pages. The term inverted document frequency intends to minimise the importance of the words that appear frequently in all documents. ([Chi, 2004]).

In the following table we have the weights calculated for the example above.

The initial query words ("data", "mining") will receive weight 0 because they are present in all the results initially retrieved by the search tools, therefore they do not bring any informational gain.

4. K-means processing, with adequate adaptations for web pages

Classical K-means clustering algorithm

K-means algorithm uses numerical input to build up distinct clusters. It splits the total data set into exclusive clusters using a measure called "distance". The distance can be calculated using many formulas, but basically having the meaning of metric distance.

K-means algorithm is as follows:

Step 1: *The number of clusters is provided as input: k parameter.*

Step 2: *K points are randomly selected. They will be the first "cluster centres": C1, ..., Ck*

Step 3: *The algorithm will place each instance in the cluster to which centre is closest according to the distance measure used.*

Step 4: *After each instance is distributed, for each cluster, the cluster "centroid" is re-calculated using all the instances that are now part of that cluster.*

Step 5: *The new centre of the cluster is the calculated centroid.*

Step 6: *We start the process from step 2, with the new centres.*

The clustering process is stopped when the same instance is repeatedly distributed to the same cluster. At this point we say the clusters are stable.

K-means adapted for web search result clustering has the following demands:

- High processing speed
- The ability to create high quality clusters
- It has to provide as output a description and a label for each cluster. This label and description will enable the user to identify the right cluster for his search.

Implementing the algorithm for search result clustering

Just as in traditional k-means clustering, a cluster will be represented by its centre or centroid ([Baeza, 99]). For web page clustering this centroid will represent a weighted word document within the vector space documents. This centroid will be called "the representative of cluster" k noted R_k. In order for R_k to be equivalent with the centre of clusters from the traditional k-means clustering, it must meet the following conditions:

- Each document d_i from C_k has joined words with R_k
- The words in R_k also appear in the most documents in C_k
- Not all the words from R_k have to also appear in all the documents from C_k
- The weight of words t_j from R_k it is calculated as an average weight of all the words present in all documents C_k ([Chi, 2004]).

After determine the representatives of the cluster, **the web metasearch results clustering algorithm** is as follows:

Input data:

- D_N *web pages (or documents) (provided by the initial search)*
- *K – number of clusters (accordingly to k-means traditional algorithm)*

Output data:

- C_k *clusters with documents (the clusters can overlap, meaning the same page can be assigned to more than just one cluster, for each distribution there will be an weight assigned to the document)*

Step 1*: K documents from the web page list are randomly selected*

- *This k documents will form the starting representatives of the clusters*
- *We will have:*
  - $C_1, C_2, ..., C_k$ *clusters*
  - $R_1, R_2, ..., R_k$ *clusters representative*

Step 2*: for each web page from D:* $d_i \in D$ *and each cluster* C_k, *k = 1, ... K*

- *We will calculate the similarity between the web page and the representative of the cluster:* $S(d_i, R_k(C_k))$

Step 3*: if the similarity is bigger than a given threshold* $S(d_i, R_k(C_k)) > \delta$, *than:*

- *The document* d_i *will be assigned to cluster* C_k *with a weight attached, the weight being*

calculated based upon the similarity value:

$$m(d_i, C_k) = S(d_i, R_k(C_k))$$

Step 4*: for each cluster the representative is re-calculated taking into account the new documents that were assigned to that particular cluster*

Step 5*: the process is re-started from step 2 until the new modifications are under a given threshold*

Step 6: *for each web page d_n that was not assigned to any cluster:*

- *The closest neighbourhood will be calculated $V(d_u)$, the neighbourhood must not contain documents that have similarity measure zero*
- *The d_n web page will be assigned to the cluster to which the neighbourhood $V(d_u)$ belongs to*
- *The weight for d_n assignation to C_k is calculated*

$$m(d_n, C_k) = m(V(d_u), C_k) * S(d_n, (V(d_u)))$$

Step 7*: for each cluster C_k the representative of the cluster R_k is recalculated*

Similarity calculation is done using the next formula:

$$S(x, y) = \frac{\sum_{i=1}^{t} x_i y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2}}, \quad (4)$$

also named Salton's cosine coefficient ([Steinbach, 2000])

5. Constructing cluster like representation for final results

a. Label extraction: for each clusters the sequence of words with the highest frequency will be assigned as label for that class. If a sequence of words is not found than a single word can be used

b. Creating the cluster structure: determining which cluster will follow which will be done using both the size of the clusters and the weight of the pages inside the cluster. The bigger the cluster and the higher the weight of the documents inside it, the higher the cluster will be in the cluster structure hierarchy. Also a measure of cluster quality can sometimes be calculated.

c. Cluster result presentation: in the pre-processing phase we extracted the root for the similar words and we eliminated the stop words, and some other modifications took place in order to make the document suitable for clustering. In order for the user to understand what a page is about it must be presented with a short relevant description of that web page. In almost all cases the initial search result list provides a short description (the snippet). That description can be kept, and re-presented to the user after clusters are made, but it can also be enhanced with the particular characteristics of the cluster assignment.

Search result clustering integration in user search process

The clustering module will present to the user a simple interface which is used to provide to the tool the search query and all the search preferences. The clustering module will take this words and options and automatically feed it to all the search tools that are used in the web search process (e.g. Google, Yahoo, MSN search, etc.). For our example we have used local clustering. All the processing needed for clustering will be made on the local computer after all the search tools used have provided their own list of results. After the search and clustering is finished, the user is presented with the final cluster structure, each cluster having attached to it a label and a description, and inside the cluster each web page having its own description and link to the page. In picture 3 we have represented the place of the web metasearch clustering module.

The interface provided by the clustering module must provide the user tools necessary to refine its search, like:

- Language support
- Ability to add / delete the search engines used for the metasearch
- Flexible presentation of results
- Possibility to ask a certain number of clusters
- Special filters: for blog sites, adult sites, etc.

Distinct clusters of subject different web pages

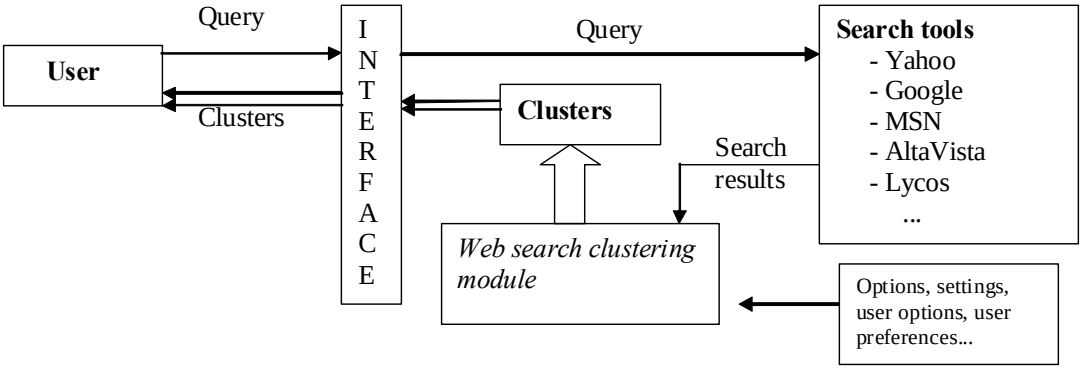

Fig.3. The place of the web search clustering module in online user search process

Conclusion

The paper presented a web page clustering algorithm and the solution to integrate it in user query web search process. The paper started by presenting the circumstances in which the need for web search result clustering algorithm appeared and it presented the characteristics that a clustering module needs to meet in order to be practically efficient.

After that we presented a classical approach of k-means clustering algorithm, and the modifications it undertook in order to be adapted to a web page clustering process. The web page clustering process was than described in detail starting with the initial document processing steps, and moving to the clustering process itself. In the end it was presented the representation requirements that the clustering module must meet in order to be efficient in a web search.

Even if clustering web search results is only a soft solution for enhancing a user's work to find the web pages it needs, it can greatly improve the results of a web search, in both time and quality. The idea to cluster web search results appeared for fist time in 2002, but the solution did not became widely popular even to this day due to the lack of understanding of its mechanisms and improvements that brings to a web search. This paper presented the user benefits given by such a tool compared to the efforts needed to implement it.

Bibliography

[Brin, 98] **Brin, S., şi Page, L**., *The anatomy of a large-scale hypertextual Web search engine*, In Proceedings of the Seventh International World WideWeb Conference, April 1998.

[Chi, 2004] **Chi, N., L.,** A tolerance rough set approach to clustering web search results, Master thesis in computer science, index 1891191, Faculty of Mathematics, Informatics and Mechanics, Warsaw University, 2003.

[Salton, 89] **Salton, G**. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[Steinbach, 2000] **Steinbach, M., Karypis, G., and Kumar, V.** *A comparison of document clustering techniques*. In KDD Workshop on TextMining (2000).

[Leouski, 1996] **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. (document available at: http://citeseer.ist.psu.edu/leouski96evaluation.html,)

[Leuski, 2000**] Leuski A. and Allan J**. *Improving Interactive Retrieval by Combining Ranked List and Clustering*. Proceedings of RIAO, College de France, pp. 665-681, 2000. (document available at: http://citeseer.ist.psu.edu/298378.html)

[Baeza, 99] **Baeza-Yates, R., Ribeiro-Neto, B.,** *Modern Information Retrieval*, Publisher: Addison Wesley; ACM Press, New York. (May 15, 1999)