Yahoo!Clusty - Adding real-time clustering functionality to the Yahoo! web search engine

Giuseppe Narzisi
Courant Institute of Mathematical Sciences
New York University
email: narzisi@nyu.edu

1 Objective of the project

Yahoo!Clusty\(^1\) is a Clustering Meta-search Engine (MSE) that allows users to send queries to Yahoo!. The returned snippets are grouped into homogeneous groups by topic.

The objective of this project has been to create a flexible MSE for the Yahoo! web search engine. The purpose is to present the results returned to a query in a more structured format which will allow the user to easily explore them by category.

The basic idea, which is has been recently become a focus of attention in the information retrieval community [6, 7], is to consider only the snippets of the returned web pages as a consistent representation of each page and grouping them in homogeneous clusters by means of clustering and categorization algorithms. The processing must be done on the fly at run-time, so it requires efficient implementation and design of technologies and algorithms in order to minimize the latency between the issuing of the query and the presentation of the results.

Many different approaches have been presented in the last 10 years (Copernic, Dogpile, iBoogie, Kartoo, Mooter, Vivisimo, etc.) and many academic prototypes have been explored as well. A recent example is given by the Armil\(^2\) [1] meta-search engine.

Given the limited amount of time and the complexity of the project, the goal is not to develop a sophisticated MSE that can outperform all the previous MSEs but to create a flexible platform for testing various clustering algorithms and labeling techniques on snippets and show that all this can be achieved in a one semester project. Moreover the system has been developed in such a way that it can be easily extended with more functionalities.

\(^1\)http://cims.nyu.edu/~gn387/websearchengines/yahoo!clusty.html
\(^2\)http://armil.iit.cnr.it
Figure 1: Architecture of the Yahoo!Clusty MSE.

2 Architecture

The architecture of the software together with the steps involved in answering a query are illustrated in figure 1.

After the user specifies the query using the web interface, the query is sent to the Yahoo! web search engine using the services offered by the Yahoo! SDK. The list of results is returned in the form of snippets (title, address/link, summary). Each snippet is then processed for stop words removal and stemming (stemming is performed using the well known Porter stemming algorithm [8]). After this text processing phase, each snippet is transformed into a bag of words and it is ready to be used for the calculation of the inverted index. Each word is assigned a (normalized) weight based on the standard TF/IDF score. At this point we have all the information for comparing documents using the standard cosine measure. Clusters are then computed and sent back to the web interface for visualization.

It is important to note that, except for the Yahoo! SDK calls, all the steps involved in answering the query are computed locally on the client machine of the user, which clearly reduce the overhead of any server side application. Each user runs locally a web Java applet that takes care of connecting to Yahoo!, downloading the results and cluster them in real-time.

The software is composed of four major parts as described in the following subsections (the order in which the part of the system are listed reflects the order in which they have been developed).

2.1 Web user interface

This component deals with the creation of the layout for query and results. It is a simple user-friendly interface where the user can type in a query. The results are displayed both in list-form and hierarchical form (clusters). Figure 2 shows the typical result of a query.

The web user interface has been be developed in HTML and JavaScript, while the rest of the software has been developed in Java in the form of a
signed-licensed web-applet. From a technology point of view it is necessary to make Java, JavaScript and HTML able to communicate between each other during the execution of the query and the processing of the results. This has been achieved using the Java Plug-in technology\(^3\), included as part of the Java Runtime Environment, Standard Edition (Java SE). This technology allows to establish a connection between popular browsers and the Java platform. This connection enables applets on Web sites to be run within a browser on the desktop. The feature which is particularly useful for the development of the project is that Java Plug-in supports Java-to-JavaScript communication with the Common DOM (Document Object Model) API\(^4\) through the use the Java JSObject\(^5\). Using this feature it is possible to write the results of the query and the clustering algorithm directly back into the HTML page using Java methods.

2.2 Text-processing software

This component takes care of elaborating the snippets, removal of stop-word, stemming and inverted index calculation. After the results of the standard Yahoo query have been collected, each result is transformed internally in a “bag of words”. Both the text contained in the summary and the title of the snippet are processed for stop-words removal and stemming. The remaining words after this sequence of steps are used later for the creation of the inverted index. The standard TF/IDF measure is used to compute the weight of each word in the vector representing the document.

2.3 Clustering component

This component includes software and algorithms for mapping the snippets into a vector space endowed with a distance function (metric) and group the snippets into clusters. The following two algorithms have been implemented: K-means\(^4, 5\) and Furthest Point First\(^2, 3\) (FPF). Different distance metrics have been developed in the past years for comparison in the document vector space. In this project the standard cosine distance has been implemented, which is the natural approach when the TF/IDF measure is used. The system has been structured in such a way that can be easily extended to use any other distance metric. Through the web interface the user can decide which algorithm to use and the number of final cluster generated. Figure 2 shows a snapshot of the typical outcome of the system to a user query.

\(^3\)http://java.sun.com/products/plugin/
\(^4\)http://www.w3.org/DOM/
\(^5\)http://java.sun.com/products/plugin/1.3/docs/jsobject.html
2.4 Labeling component

This component includes software for extracting from the set of snippets assigned to each cluster a sequence of words which well describes the group of items. Clearly the quality of a label depends on its descriptive power and discriminative power. For this component, a very simple approach has been used: the label of a cluster is made out of the best 5 words (according to their TF/IDF value) which belong to the centroid of the cluster. Even if no sophisticated labeling technique is used, the performance of the centroid approach work fairly well in general and, in the specific case of real-time clustering, requires minimal computational effort. Moreover, the user is typically able to identify the major topic of the cluster just looking at few words describing it. In addition to the 5-words label, each cluster is assigned its size (number of pages in the cluster) and a cohesiveness value that represents the measure of the cluster similarity. One method for computing the cohesiveness is to use the weighted similarity of the internal cluster similarity. It can be shown that this corresponds to the squared length of the cluster centroid [5].

3 Some query examples

Although no systematic evaluation of the MSE has been computed, different query examples have been tested in order to empirically evaluate the quality of the two clustering approaches (k-means and FPF). As experimentally observed in other papers in the literature, the FPF algorithm typically shows...
a faster convergence and a overall better quality of the results. On average the FPF produces less misclassified points (page belonging to the wrong cluster) than kmeans. Also the FPF algorithm suffers less than kmeans of the random initial selection of the points: from run to run the clusters generated by the FPF algorithm are way more stable than for kmeans. As an example of the Yahoo!Clusty behavior, in the following it is reported the list of the clusters generated by the MSE for the query “Jaguar” using the FPF clustering algorithm with a request of 5 different clusters; the number close to each web-page represents the actual rank in the ordered list returned by Yahoo!.

The MSE is clearly able to identify the clusters related to many of the different meanings for the word “Jaguar”: cars, music instruments and bands, apple mac os x, animals. Particularly interesting is that for the same query “Jaguar” kemans algorithm is never able to generate the small cluster related to music, while the FPF approach always returns a cluster composed of the only two pages related to music instruments and bands among the Yahoo! results (cluster number 2 in the list).

1. car,review,price,part,quot (46) (cohesiveness = 0.044)
   73 Niello’s Jaguar Sacramento—Sacramento Jaguar Dealer Minutes from Roseville, Elk Grove & Folsom
   2 One World Journeys — Jaguar: Lord of the Mayan Jungle
   8 Jaguar US - Home
   9 Jaguar UK - Jaguar Cars
   10 Jaguar Cars - Wikipedia, the free encyclopedia
   ...

2. music,funkgroov,latin,orient,jagar (2) (cohesiveness = 0.538)
   27 JAGAR — Funk
   41 ::: Fender.com :::.

3. appl,featur,product,inform,ford (20) (cohesiveness = 0.074)
   77 Apple - Mac OS X Leopard
   1 Jaguar
   4 CSG Species Accounts: Jaguar (Panthera onca)
   7 New Jaguar Cars Research All Models - Edmunds.com
   13 Jag-lovers - here to provide everything for the Jaguar enthusiast!
   ...

4. xj,type,owner,forum,discuss (11) (cohesiveness = 0.128)
   82 Jaguar
   21 Jaguar Models - Main Page (resin model kits)
   31 Ford Motor Company Vehicle Brands - Ford
   35 Jaguar
   66 Compare Prices and Read Reviews on Jaguar Cars at Epinions.com
Another interesting query used for tasting the MSE has been “Hepburn”. Inspecting the output of many different runs, the typical structure of the results consists of two major clusters corresponding to the two famous Hepburn actresses (Katharine and Audrey), plus a number of additional clusters that do not clearly reflect any other specific category. If the user specify exactly 2 as the number of generated clusters \( k \) than the FPF algorithm always returns two clusters, one for Katharine and one for Audrey, while kmeans typically mixes pages of the two actresses. For this specific query it would have been more useful to have same form of hierarchical clustering such that, under each of the two major clusters for Katharine and Audrey, the pages were organized by topics like photos, music, movies, etc. In the following the clusters generated by Yahoo!Clusty using the PFP algorithms and \( k = 5 \) for the query “Hepburn” are reported.

1. katharin,photo,movi,video,link (26) (cohesiveness = 0.069)
   81 Katharine Houghton Hepburn
   9 The New York Times Movies
   11 TASCHEN Books: Katharine Hepburn
   16 Katharine Hepburn Photos - Katharine Hepburn News - Katharine Hepburn Information
   29 Katharine Hepburn Quotes

2. audrei,biographi,film,ladi,new (30) (cohesiveness = 0.06)
   94 Box Office Report - Revenue Database - Audrey Hepburn
   6 Audrey Hepburn
   7 YouTube - Audrey Hepburn Tribute
   8 YouTube - Audrey Hepburn
   12 TASCHEN Books: Audrey Hepburn

3. gt,new,com,pictur,myspac (18) (cohesiveness = 0.08)
   73 Mia Shoes.com: Hepburn
   4 Welcome to Audrey Hepburn.com
   14 MySpace.com - Hepburn - BOSTON, Massachusetts - Rock
   17 The Katharine Hepburn WebRing
4 External software & Web resources

In order to communicate with the Yahoo! web search engine and collect the results, I have used the Yahoo! Search Web Service SDK\(^6\) (vs. 2.0.2). Yahoo! SDK allows to access Yahoo! content and services using an application ID. Like a browser’s User-Agent string, the Application ID uniquely identifies your application and has no effect on rate limiting. Initially I was planning to use the respective tool for the Google services (Google SOAP Search API\(^7\)), but apparently Google has decided to stop issuing new API keys for the SOAP Search API and the new available platform (Google AJAX Search API\(^8\)) allows to retrieve only 8 results per query! The Yahoo! SDK is a library written in Java that allows to connect to Yahoo! and download the results in the form of WebSearchResult objects. Then it is possible to extract title, address and snippet of each result respectively using the methods getTitle(), getUrl() and getSummary().

Particularly useful and of inspiration for the Java implementation of the system, has been the information retrieval software developed by Professor Raymond J. Mooney\(^9\) at University of Texas at Austin. This code supplies ”miniature” pedagogical Java implementations of information retrieval, spidering, and other IR and text-processing software. It is being released for educational and research purposes only under the GNU General

---

\(^6\)http://developer.yahoo.net/
\(^7\)http://code.google.com/apis/soapsearch/
\(^8\)http://code.google.com/apis/ajaxsearch/
\(^9\)http://www.cs.utexas.edu/users/mooney/
Public License, and it was developed for an introductory course on "Intelligent Information Retrieval and Web Search".

5 Future Works

The main goal of the project has been the development of a flexible and extensible Meta Search Engine for snippet clustering called Yahoo!Clusty. In the following many possible extension of the system are listed:

- Improving the web user interface with new enhancements (like an advanced search for expert users).
- Implementing other clustering algorithms: agglomerative hierarchical clustering, bisecting kmeans, one-pass clustering, suffix tree clustering, etc.
- Systematic evaluation of the clustering algorithms: after many other algorithms will be available in the system, a systematic evaluation of their performance will be possible using well known cluster quality measures.
- Clusters merge operation: all the clusters that share the same signature should be merged. This way we avoid to ask to the user to select the parameter $k$ for the number of output clusters.
- Extending the retrieval of the results from Yahoo! to other well known web search engines (Google, Ask, etc.).

References


http://www.cs.utexas.edu/users/mooney/ir-course/

