Noodles: A Clustering Engine for the Web

Giansalvatore Mecca, Salvatore Raunich,
Alessandro Pappalardo, Donatello Santoro

Dipartimento di Matematica e Informatica,
Università della Basilicata, Potenza, Italy

Abstract. The paper describes the Noodles system, a clustering engine for Web and desktop searches. By employing a new algorithm for document clustering, based on Latent Semantic Indexing, Noodles provides good classification power to simplify browsing of search results by casual users. In the paper, we provide some background about the problem of clustering search results, give an overview of the novel techniques implemented in the system, and present its architecture and main features.

1 Background

Web and desktop search services are nowadays essential tools for any Internet user. However, keyword-based, boolean-style search engines like Google usually fall short when asked to answer rather broad queries – those that are often posed by less-experienced users – like, for example, to find documents about the term "*power*" or the term "*amazon*". The poor quality of results in these cases is mainly due to two different factors: (*a*) polysemy and/or synonimity in search terms (*b*) excessively high number of results returned to the user. As a consequence, less skilled users are often frustrated in their research efforts.

The Semantic Web promises to solve most of these problems by adding semantics to Web resources. In fact, there have been some proposals in the literature towards a semantic Web search engine [4]; these proposals assume that documents can be classified based on their RDF or OWL annotations, and therefore are not immediately applicable. As a consequence, at the moment the most promising efforts along the way to Semantic Web search services are the so-called *clustering engines.*

The idea of clustering search results is not new, and has been investigated quite deeply in Information Retrieval, based on the so called *cluster hypothesis* [10] according to which clustering may be beneficial to users of an information retrieval system since it is likely that results that are relevant to the user are close to each other in the document space, and therefore tend to fall into relatively few clusters. Several commercial clustering engines have recently emerged on the market. In fact, even Google has experienced for a while with forms of clustering of their search results [8], and has recently introduced a "Refine Your Query" feature[1] that essentially allows to select one of a few topics to narrow a search. Similar experiments are also being conducted by

1 http://www.google.com/help/features.html#refine

Microsoft [9]. Other well known examples of commercial clustering engines are Vivisimo [11] and Grokker [3]. These systems share a number of common features with research systems introduced in literature, mainly the Grouper system [12] [13] and Lingo/Carrot Search [6] [7]. We summarize these features in the following.

First, these tools are usually not search engines by themselves. On the contrary, when a user poses a query, the clustering engine uses one or more traditional search engines to gather a number of results; then, it does a form of post-processing on these results in order to cluster them into meaningful groups. The cluster tree is then presented to the user so that s/he can browse it in order to explore the result set. It can be seen that such a technique may be helpful to users, since they can quickly grasp the different meanings and articulations of the search terms, and more easily select a subset of relevant clusters.

Being based on a post-processing step, all of these clustering engines work by analyzing *snippets*, i.e., short document abstracts returned by the search engine, usually containing words around query term occurrences. The reason for this is performance: each snippet contains from 0 to 40 words, and therefore can be analyzed very quickly, so that users do not experience excessive delays due to the clustering step. However, snippets are often hardly representative of the whole document content, and this may in some cases seriously worsen the quality of the clusters.

2 The Noodles System

The Noodles system, a snapshot of which is shown in Figure 2.1, is a clustering engine for Web and desktop searches. It is based on a novel algorithm for clustering search results [5]. One of the main features of the system is that it is not based on snippets, but it requires access to the whole document contents and uses a different compression techniques to improve performance.

A key objective of the Noodles clustering technique is to achieve a high level of quality in terms of its ability to correctly *classify* documents, i.e., to dynamically build a bunch of clusters that correctly reflect the different categories in the document collection returned by the search engine. Similarly to [1], we believe that this ability may assist less-skilled users in browsing the document set and finding relevant results.

The clustering algorithm that has been developed is called *Dynamic SVD Clustering* (*DSC*) [5], and it is based on Latent Semantic Indexing [2]; the novelty of the algorithm is twofold: (*a*) first, it is based on an incremental computation of singular values, and does not require to compute the whole SVD of the original matrix; (*b*) second, it uses an original strategy to select k, i.e., the number of singular values used to represent the "concepts" in the document space; differently from other proposals in the literature, our strategy does not assume a fixed value of k, neither a fixed approximation threshold.

In [5], based on experimental results, we show that the algorithm has very good classification power; in many cases it is able to cluster pre-classified documents collections with very good accuracy; it is worth noting that the

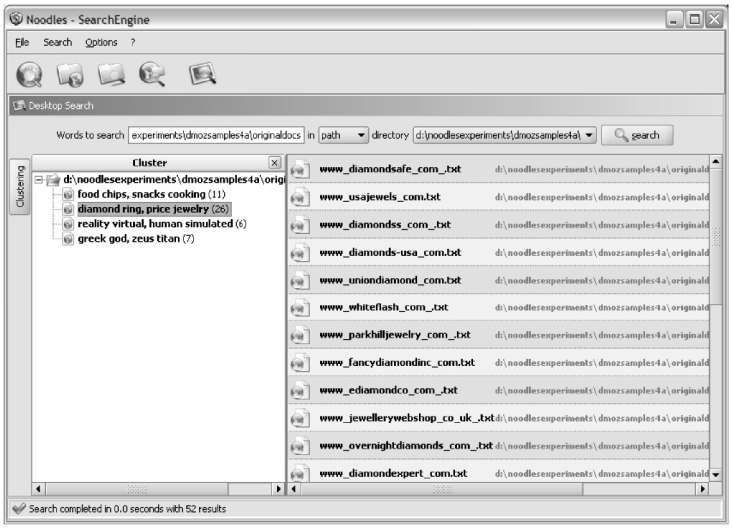

Figure 2.1: A Snapshot of the Noodles Desktop Search Engine

quality of the classification severely degrades when snippets are used in place of the whole document content, thus providing further evidence that snippets are often too poor and not sufficiently informative.

Note that such good classification power has a cost in terms of computing times, since the SVD-based computation has higher complexity with respect to typical phrase analysis algorithms based on snippets. However, the algorithm has been designed in such a way that in practice it has good performance, and lends to a very natural clustering strategy based on the minimum spanning tree of the projected document space [5].

3 Architecture of the System

The system is fully written in Java. It comprises several modules and some well-known Java libraries, as shown in Figure 3.1.

Spring[2] has been adopted as a dependency-injection framework. This greatly helped the development in several respects; on the one side it improved the overall system modularity; on the other side, it helped to experiment different strategies for various aspects of the clustering algorithm. For the development of the desktop user interface, we adopted the Spring Rich Client Platform, which is tightly coupled to the dependency-injection framework, and provides further ease of configuration and an effective binding framework for forms and wizards.

The system supports both Web and desktop searches. Web searches use Google via its web service API. To simplify the experimental phase, it also incorporates a module to sample categories in the DMOZ directory (dmoz.org).

Desktop searches use Apache Lucene (lucene.apache.org) as an indexing

2 http://www.springframework.org

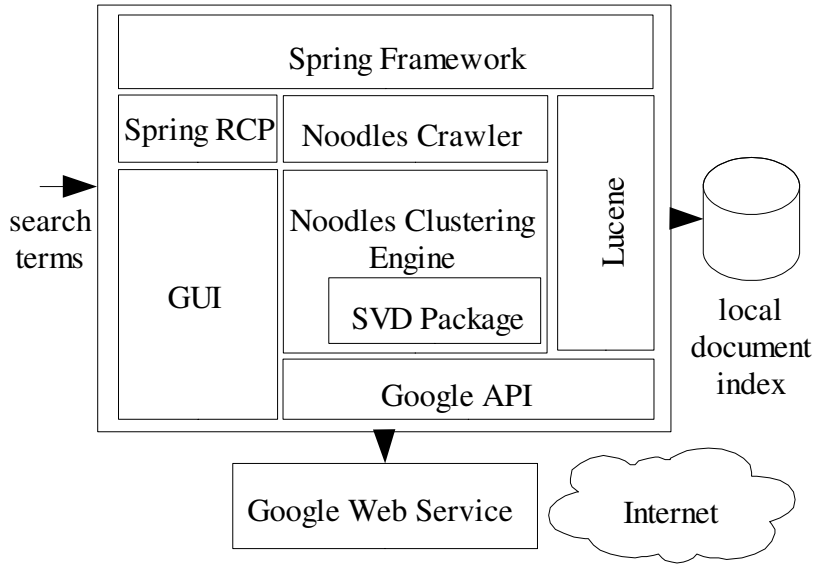

Figure 3.1: Architecture of the System

engine. The Noodles crawler runs as a daemon and incrementally inspects the local disk to feed files to Lucene for indexing purposes. Its behavior is fully customizable: users may select what portion of the local disk they intend to index.

At the core of the system stands the Noodles clustering engine, which implements the dynamic SVD clustering algorithm. In order to perform SVD computations, COLT (dsd.lbl.gov/~hoschek/colt) was selected as a matrix manipulation package, although we are performing further experiments using JScience (www.jscience.org), the newest math package for the Java language; in our experiments, JScience – which is based on a high performance library called Javolution (javolution.org) – showed significantly better performance both in terms of computing time and heap usage with respect to COLT and other similar packages. Unfortunately, JScience currently does not offer support for SVD, although this might be added to future version. Nevertheless, we believe that JScience might significantly improve computing times once support for SVD is implemented.

In implementing the system, special care was devoted to reduce computation times. For example, we developed our user interface in such a way to minimize the latency associated with the display of clusters. More specifically, when a user runs a query, the system very quickly returns the ranked search results. At the same time, it starts to cluster top-ranked results on a background thread, and shows a "Clustering" button on the screen. It the user wants to see the clusters, s/he has to select this button. Considering typical user reaction times, the net effect of such an organization of the user interface is that, in the user perception, the clustering is almost immediate.

4 Description of the Demonstration

During the demonstration of the system, we will show its features by running several searches, both on the Web and on the local disk. More specifically:

- we will perform searches based on typical polysemic terms, like "power", "amazon", and "life", and show how the system produces meaningful clusters that correspond to the most frequent meanings of these terms, and correctly classifies documents with respect to these meanings;
- we will perform searches based on pre-classified documents, sampled from DMOZ, and show how the system is typically able to correctly reproduce the DMOZ classification;
- we will perform free searches both on the desktop and on the Web and try to assess the quality of the clustering based on user feedbacks.

Bibliography

1. C. Chekuri, P. Raghavan. Web Search Using Automatic Classification. In *Proceedings of the World Wide Web Conference*, 1997.
2. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Sciences* (1990) 41(6):391-407.
3. The Grokker Search Engine. http://www.grokker.com.
4. R. Guha, R. Mc Cool, E. Miller. Semantic Search. In *Proceedings of the World Wide Web Conference*, 2003.
5. G. Mecca, S. Raunich, A. Pappalardo. A New Algorithm to Cluster Search Results. *Data and Knowledge Engineering* () To appear:doi:10.1016/j.datak.2006.10.006.
6. S. Osinski, J. Stefanowski, D. Weiss. Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition. In *Proceedings of the International Conference on Intelligent Information Systems (IIPWM)*, 2004.
7. S. Osinski, D. Weiss. A Concept-Driven Algorithm for Clustering Search Results. *IEEE Intelligent Systems* (2005) 20(3):48-54.
8. Web 2.0 - Exclusive Demonstration of Clustering from Google. http://www.searchenginelowdown.com/2004/10/web-20-exclusive-demonstration-of.html.
9. The SRC Search Engine. http://rwsm.directtaps.net/.
10. C. J. van Rijsbergen. *Information Retrieval, Second Edition*. London, Butterworths, 1979.
11. The Vivisimo Search Engine. http://www.vivisimo.com.
12. O. Zamir, O. Etzioni. Web Document Clustering: A Feasibility Demonstration. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1998.
13. O. Zamir, O. Etzioni. Grouper: A Dynamic Clustering Interface for Web Search Results. *Computer Networks* (1999) 31(11-16):1361-1374.