# The Anatomy of a Hierarchical Clustering Engine for Web-page, News and Book Snippets *

Paolo Ferragina Antonio Gullì
Dipartimento di Informatica, Università di Pisa, Italy

Current search engines return a ranked list of web pages that contain the keywords of the user query together with some *contextual information*, in the form of a page excerpt, the so called *(page or web) snippet*. The key difficulty in the searching process relies in what is *"relevant"* to the user. The same set of keywords may abstract different user needs that may also vary over the time according to the context in which the user is formulating his/her own query. Recently, there has been a surge of commercial interest in novel IR-tools that help the users in their difficult search task by means of novel ways for reporting the query results. Two success stories are represented by Vivisimo and Dogpile that were elected as the best meta-search engines of the last three years 2000-03 by a jury composed by more than 500 web users, as reported by SearchEngineWatch.com. Other commercial tools are Mooter, Copernic, iBoogie, Kartoo, and Groxis. These IR-tools add to the flat list of query results a *hierarchy of clusters* built on-the-fly over the snippets. Each node of this hierarchy is properly *labeled* via a meaningful sentence that captures the "theme" of the snippets (and, thus, of the corresponding web pages) contained into its cluster. As a result, users are provided with a small, but intelligible, picture of the query results at various levels of details. So that the user has no longer to browse through tedious pages of results, but may navigate through labeled folders.

In this paper, we investigate the *web snippet hierarchical clustering* problem in its full extent by devising an algorithmic solution, and a software prototype called SnakeT (accessible at http://roquefort.di.unipi.it/), that: **(1)** draws the snippets from 16 Web search engines, the Amazon collection of books a9.com, the news of Google News and the blogs of Blogline; **(2)** builds the clusters on-the-fly (ephemeral clustering [10]) in response to a user query without adopting any pre-defined organization in categories; **(3)** labels the clusters with sentences of variable length, drawn from the snippets and possibly missing some terms, provided they are not too many;

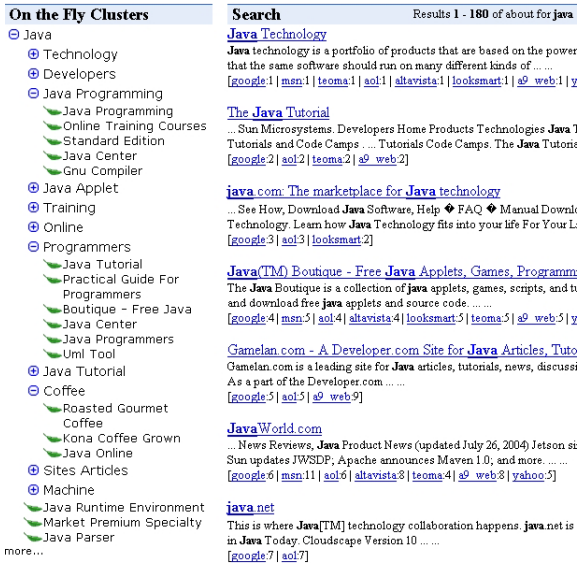


**Figure 1. SnakeT's Clusters for "*Java*".**

**(4)** uses some ranking functions which exploit *two knowledge bases* properly built by our engine at preprocessing time for the *sentences selection and cluster-assignment process*; **(5)** organizes the clusters into a hierarchy, and assigns to the nodes intelligible sentences in order to allow post-navigation for query refinement. Our clustering algorithm possibly let the clusters overlap at different levels of the hierarchy.

We remark that the problem we have in our hands is different from the canonical clustering which is persistent since "in normal circumstances, the cluster structure is generated only once, and cluster maintenance can be carried out at relatively infrequent intervals" [11]. Moreover this problem is made difficult by the fact that, for efficiency issues, the labeling must exploit just the snippets and they are less informative than their documents because consist of about 100 words. In summary, our result is innovative in two respects: it introduces novel algorithmic ideas, and it is the first in the literature to address the ephemeral clustering of three kinds of heterogeneous collections such as the Web,

*Partially supported by the Italian MIUR projects ALINWEB, ECD, the "Italian Grid Project", "Distributed high-performance platform", and by the Italian Registry of ccTLD.it. Contact: {ferragina,gulli}@di.unipi.it

catalogues of books and rapid evolving collections of news and blogs.

# 1 The software architecture

Figure 2 illustrates the software architecture of SnakeT. There are two types of computations: an off-line computation that builds two knowledge bases at preprocessing time; and an on-line computation that is executed at query time and groups the snippets into clusters, extracts meaningful sentences to label them and produces a hierarchy. An extended description is in [3].

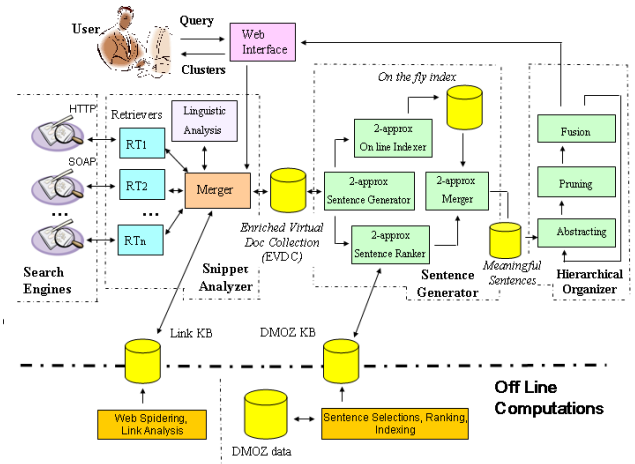


**Figure 2. The architecture of SnakeT.**

## 1.1 The two knowledge bases

**The Anchor Text and the Link Database.** Several search engines exploit the hyperlinks among Web pages as a source of information for ranking and retrieval. We use this information to cluster the snippets by modelling each Web page $p$ as a *virtual document* consisting of two sets of terms: The set $A(p)$ formed by the terms contained in $p$, and the set $B(p)$ formed by the terms contained in the (anchor) text surrounding each hyperlink that points to $p$. Our experiments showed that $B(p)$ often provides a precise, yet concise, description of $p$. Unfortunately, common search engines do not return $B(p)$. Hence, we deployed the *Nutch's open source spider* to collect more than 200 millions Web pages selected among the top-cited ones. Hereafter, this index is denoted by $\mathcal{A}_{\texttt{link}}$.

**The Semantic Knowledge Base.** We introduce this KB to *rank* a set of candidate sentences, and select the most meaningful ones for our labeling purposes. There are many proposals in the literature to rank individual terms, either based on precomputed archives (e.g. ontologies, dictionaries and lexical databases, as WordNet and AppliedSemantics) or on frequency information, e.g. `TFxIDF`. To exploit the positive features of both approaches, we designed our clustering engine based on a *mixed* strategy. We indexed DMOZ, a high-quality directory freely available on the Web, and developed a *ranking engine* upon it, hereafter denoted by $\mathcal{R}_{\texttt{dmoz}}$. This engine exploits more than three millions single terms and implements an ad-hoc *DMOZ-category*-centered `TFxIDF` measure, taking into account single terms, pairs of terms, longer sentences and their positions in the DMOZ tree. As a result, we are implicity exploiting a whole Web directory, but unlike [2], we are using it only for *ranking* the candidate sentences which, we remark, are created on-the-fly from the snippets at query time.

## 1.2 First module: The Snippet Analyzer

The task of this module is to forward the user query to a set of selected, remote search engines and to retrieve and merge the returned snippets. Currently we request an average of 40 Web snippets to each search engine using async I/O. The merged snippets are enriched with information retrieved from $\mathcal{A}_{\texttt{link}}$ and processed by a *Linguistic Analyzer* which filters a *stop list* in 12 different languages; stems the result by using a variant of the Snowball stemmer; and finally extracts Part of Speeches and Named Entities. The result is the Enriched Virtual Document Collection (EVDC).

## 1.3 Second module: The Sentences Generator

The task of this module is to create a set of meaningful labels as "approximate" sentences extracted from the EVDC. These labels will be then used to form and name the *clusters* and all nodes in the *hierarchy*. We are interested in long and intelligible sentences which are much more useful than a single term, but their extraction and management introduces some difficulties. For example, the sentences "John Fitzegerald Kennedy", and "Kennedy John" are syntactically different, but it would be desirable to consider them as the "same sentence". Grouper and the other academic tools [15, 17, 12, 4] did not deal with this problem because they treated sentences *formed by contiguous terms*. Nevertheless it is clear from the "future directions of research" of [15, 10] that the extraction of sentences involving not contiguous terms might boost the precision of the cluster labels and, in turn, the usefulness of the cluster hierarchy to humans. Vivisimo makes indeed use of approximate sentences as it is clear from its working. To the best of our knowledge, we are the first to design a clustering engine that tries to cluster together Web snippets containing *almost the same sentences*.

**First phase: Construction.** We generate "2-*approximate sentences*" by drawing all the pairs of (non contiguous) terms that occur within a proximity $d$ in each sentence identified by the Snippet Analyzer. Stop words are skipped. Of course, some of the term pairs could introduce artificial meanings. Hence, we apply a simple rule: We say that $(t_h, t_k)$ is a *valid* term-pair if either the two terms $t_h$ and $t_k$ appear contiguously in some other document of the

EVDC or they appear contiguously in $\mathcal{R}_{\texttt{dmoz}}$. Sentences not valid are discarded.

**Second phase: Ranking.** The 2-approximate sentences are ranked using both $\mathcal{R}_{\texttt{dmoz}}$ and the frequency information derived from the snippets themselves. Our ranking has two peculiarities: it exploits *"local"* information, via a set of statistics collected from the Web snippets, and *"background"* information, via a judgement on the importance of a 2-approximate sentence assigned according to $\mathcal{R}_{\texttt{dmoz}}$. 2-approximate sentences below a threshold are removed.

**Third phase: Merging.** The filtered 2-approximate sentences participate to a merging process whose aim is to form longer *$k$-approximate sentences*. Here, we exploit an index based upon a combination of bitmaps and inverted lists, and built *on-the-fly* on the filtered sentences.

### 1.4 Third module: The Hierarchical Organizer

Given the $k$-approximate sentences, we postulate that *"virtual documents sharing the same $k$-approximate sentences belong to the same cluster"*. According to this policy, a single document may occur in many clusters, being this consistent with the observation that a Web page can cover *multiple themes*. The result of this initial clustering step is a flat pool of clusters (leaves of our hierarchy), each one having assigned a $k$-approximate sentence as meaningful label. This label is hereafter called the *primary label* of that cluster. Primary labels indeed occur within all the (virtual) documents forming a cluster and they are the sentences reporting the largest rank. They are therefore of *high quality* but are often *too much specific* to generate the labels of the nodes in the higher levels of the hierarchy. Our next key choice has been then to associate to each (leaf) cluster also a *set of secondary labels* defined as follows: they are $k$-approximate sentences which have a *good rank* and occur in *at least* the $c$% of the documents contained in the cluster. If $c$ is sufficiently large the secondary labels provide a description for the cluster at a coarser level of detail (than the primary label), and are thus more useful for hierarchical formation and labeling. In order to manage primary and secondary labels efficiently, we concatenate them into a single string, separated by a special symbol \$, called the *signature* of the cluster.

The hierarchy formation process consists actually of three phases executed repeatedly until a few number of superclusters remain: *abstraction*, *pruning* and *fusion*. The abstraction phase aims at building another level of the hierarchy starting from a set of (super)clusters $SC_1, \ldots SC_z$, each having its own signature. We compare the signatures of all $SC_i$s and extract the common substrings which do not include the special symbol \$. If the signatures of $SC_{i_1}, \ldots, SC_{i_z}$ contain the substring $s$, then we form a new supercluster $SC'$ whose primary label is $s$ and whose documents are given by the union of the documents of $SC_{i_1}, \ldots, SC_{i_z}$. We can look at $SC'$ as the *father* of $SC_{i_1}, \ldots, SC_{i_z}$ in the hierarchy. The rank of $s$ is computed as a function of the rank of the (primary or secondary) labels having $s$ as a substring. We exploit the properties of the signatures to execute an all-against-all comparison among these signatures in almost linear time. Moreover we point out that, $s$ is not necessarily a contiguous substring of the original Web snippets since it is a substring of a $k$-approximate sentence. The final pruning phase aims at simplifying the hierarchy obtained so far, by removing the *redundant* superclusters. We formalize this as a *covering* problem and solve it via a greedy approach. Finally, we compute the secondary label of each remaining supercluster, by uniting the secondary labels of its children and by discarding those labels which occur in less than $c$% of its included documents. At this point we have re-established the inductive conditions on the superclusters at the current level, and we can repeat the above three phases.

## 2 Related Works and Experimental Results

Various papers have been recently published onto this challenging problem [7, 15, 17, 12, 4, 10, 5, 9, 16, 8, 13]. Grouper [15] uses *contiguous phrases* of variable length drawn from the Web snippets by means of a Suffix Tree data structure. SHOC [17] uses instead a Suffix Array and organizes the clusters in a hierarchy via an SVD approach. In Lexical Affinities Clustering [10], the indexing unit consists of a *pair of words* (not necessarily contiguous) that are linked by a lexical affinity (LA). All the above softwares are **no longer available** as the authors communicated to us. FIHC [4] uses Frequent Itemsets , WebCat [5] uses Transactional K-Means, Retriever [7] uses robust relational fuzzy clustering. These three approaches do not form any hierarchy and the labels are *single keywords*. [14] proposes to combine links and content in a k-means framework. The problem is that search engines don't provide an easy access to the links graph, as we discuss in section 1.1. [1, 2] use *static classification* and exploit statistical techniques to learn a model based on a labeled set of training documents. The model is then applied to new documents to determine their categories. We believe that using a predefined set of categories is not flexible enough to capture the different, fine themes of the pages answering to a given user query. Hearst [6] in fact suggested that ".. clusters [should be] created as a function of which documents were retrieved in response to a query, and therefore have the potential to be more closely tailored to characteristics of a query than independent, static, clustering.". The most recent results come from Microsoft and IBM. [16] extracts sentences of variable length but contiguous, via five different measures through regression. The approach is not hierarchical, and in fact the authors highlight the need of (1) a hierarchical clustering, and (2) external taxonomies for improving labels precision.

This is actually what we do with SnakeT. [8] proposes a greedy algorithm to build the hierarchy based on a minimization of an objective function similar to ours. However, their labels are contiguous sentences and usually consist of single words. We point out that [16, 8] **didn't provide us with an access** to their software, so that a fair comparison with them has been not yet possible to be performed.

A Web interface is instead provided by three software: CIIRarchies [9] extracts sentences using a pre-computed language model and builds the hierarchy via a recursive algorithm; Highlight [13] adopts lexical analysis and a probabilistic framework for hierarchy construction, and Carrot2 [12] being an open source implementation of Grouper. These softwares have been checked against SnakeT. We performed several user studies running our system on a Linux PC with a CPU P4 and RAM 1.5Gb. The first study was aimed at understanding whether a Web clustering engine is an useful complement to the flat, ranked list of results. We asked to 45 people, of intermediate web ability, to use VIVISIMO during their day by day search activities. After a test period of 20 days, 85% of them reported that using the tool "[..] get a good sense of range alternatives with their meaningful labels", and 72% said that the most useful feature is "[..] the ability to produce on-the-fly clusters in response to a query, with labels extracted from the text". Then, we selected 18 queries belonging to many different topics (*iraq, bush, data mining, bill gates, last minute, car rental, mp3, divx, sony, final fantasy, ipo, equity, google ipo, warterlo, second war, aids, allergy, nasa*, and asked three users to compare our results against those provided by Mooter, CIIRarchies, Highlight, Carrot2. For a large part of the queries users did not like Mooter, since it provides single terms clusters. Carrot2 often tends to create a number of clusters which exceed the number of snippets itself (!!), thus reducing the need of such IR-tool. Carrot2 also fails to cluster together similar labels such as "knowledge, knowledge discovery", "mining and knowledge", and furthermore labels the hierarchy paths with sentences which are one the substring of the other. Highlight obtains its first-level's labels by using classification, so that they are few and of little use. In many cases (e.g. "data mining"), the clustering process produces identical subtrees under different top level categories and a number of clusters which exceeds the number of snippets themselves (e.g 160 clusters for the "iraq" query). CIIRarchies provides good hierarchies but, as the authors themselves admit, they are often not compact, have large depth and contain some non content-bearing words which tend to repeat (e.g ""DivX download" occurs under 5 different fathers and at different depths). CIIRarchies and Highlight are significantly slower than SnakeT. Our third study was aimed at drawing a preliminary evaluation of our software against Vivisimo. We selected 20 students of the University of Pisa and asked them to execute the above queries on VIVISIMO and SnakeT. 75% of them were satisfied of the quality of our hierarchy and its labels.

In summary, no academic result has achieved a performance comparable to the one provided by Vivisimo, and no one proposed a unifying clustering system with a Web interface for both Web Search, Books, News and Blog domains. This is one of our main achievements with SnakeT. Moreover we claim that the architecture of SnakeT (Fig. 2) is simple and its blocks can be used easily by other researchers. In the light of the experimental results, some issues remain to be further investigated. We are: (1) extending and refining our users' studies on free queries and on a larger and more variegate set of users; (2) investigating the benefits carried out by each module of SnakeT; and (3) studying some forms of mathematical evaluation for the "quality of the labeled hierarchy of clusters". This is an interesting research problem as stated in [8].

## References


[1] G. Attardi, A. Gulli, and F. Sebastiani. Theseus: categorization by context. In *WWW99*.

[2] H. Chen and S. T. Dumais. Bringing order to the web: automatically categorizing search results. In *SIGCHI00*.

[3] P. Ferragina and A. Gulli. The anatomy of a clustering engine. Technical report, TR04-04 Informatica, Pisa, 2004.

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

[5] F. Giannotti, M. Nanni, and D. Pedreschi. Webcat: Automatic categorization of web search results. In *SEBD03*.

[6] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR-96*.

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

[8] K. Kummamuru. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *WWW04*.

[9] D. J. Lawrie and W. B. Croft. Generating hiearchical summaries for web searches. In *SIGIR03*.

[10] Y. S. Maarek, R. Fagin, I. Z. Ben-Shaul, and D. Pelleg. Ephemeral document clustering for web applications. Technical Report RJ 10186, IBM Research, 2000.

[11] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

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

[13] Y. Wu and X. Chen. Extracting features from web search returned hits for hierarchical classification. In *IKE03*.

[14] M. K. Yitong Wang. On combining link and contents information for web page clustering. In *DEXA02*.

[15] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. In *Computer Networks 98*.

[16] H. Zeng, Q. He, Z. Chen, and W. Ma. Learning to cluster web search results. In *SIGIR04*.

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