Università di Pisa
Dipartimento di Informatica

Technical Report: TR-04-21

A personalized search engine based on web-snippet hierarchical clustering

Paolo Ferragina
Antonio Gullí
Dipartimento di Informatica, Pisa

November 23, 2004

ADDRESS: via F. Buonarroti 2, 56127 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

A personalized search engine based on web-snippet hierarchical clustering*

Paolo Ferragina Antonio Gullí
Dipartimento di Informatica, Pisa

November 23, 2004

Abstract

Search engines provide the view of the Web, and their smart ranking algorithms are their point of view. To offer the best view, personalized ranking algorithms are currently flourishing. They focus on the users rather than on their submitted queries, by taking into account some contextual/profiled information.

In this paper we propose a personalized (meta-)search engine based on the web-snippet hierarchical clustering technology (a là VIVISIMO) that is fully adaptive and non intrusive both for the user and for the queried search engine(s). It works on the top of 16 commodity search engines and fetches 200 (or more) results from them per user query. Our engine is able to mine on-the-fly the fine and variegate "themes" behind these results and then organize them in a hierarchy of folders that offers, at various levels of details, an up-to-date picture of these results. Users can therefore browse the hierarchy, select the themes that best match the "intention" behind their query, and ask our engine to personalize on-the-fly those query results according to their choices. In this way lazy users are not limited to look at first ten results, but immediately acquire several points of view on a larger pool (about 200) of them!

We claim that *it does exist a mutual reinforcement relationship between ranking and web-snippet clustering* from which both of them may benefit. Our extensive experiments show that this form of personalization is very effective in informative queries, polysemous queries, and poor queries consisting of at most two terms (more than 80% of the Web queries are of this type!). In these cases, in fact, one theme might be so web-popular to unfortunately monopolize the top-ten results of link-based ranking algorithms.

1 Introduction

The purpose of a search engine is to retrieve from a given collection the documents deemed "relevant" for a user query. The key difficulty lies with the definition of *"relevant"*. In fact, the same set of keywords forming several user queries might abstract different needs that may also vary over the time according to the context in which users are operating. Personalization (e.g. EUREKSTER), user behavior profiling (e.g. GOOGLE, YAHOO), query term suggestion (e.g. ASKJEEVES) are some of the tools provided by third-generation search engines to better and better satisfy the "user requirements". Web-snippet clustering[1] is another approach to help users in their difficult retrieval task. The basic idea is that the *snippets* returned by a (meta-)search engine are clustered into *folders* which are *labeled* with sentences intelligently selected from the phrases contained in the snippet themselves. The folders are then organized in a *hierarchy*, whose internal nodes are also labeled with sentences. The labels are as up-to-date and fresh as the query results. Note that each label captures the "theme" of the snippets (and thus, of the corresponding web pages) contained into its associated folder. Users can therefore *navigate* through the labeled folders driven by the "intention" behind their query. These nice features have driven some authors to say that *"clustering technology is the* PAGERANK *of the future"*, and indeed there is some rumor about Google's research on web-snippet clustering. [2]

The web-snippet clustering problem *is* a challenging variant of the classical clustering problem because of four demanding requirements. (i) The folders must be formed on-the-fly from the snippets returned by the queried search

*Partially supported by the Italian projects ECD, the "Italian Grid Project", "Distributed high-performance platform", and by the Italian Registry of `ccTLD.it`.

[1] A Web snippet is a fragment of a Web page summarized by the remote search engine, where searched keywords are usually highlighted.

[2] http://www.searchenginelowdown.com/2004/10/web-20-exclusive-demonstration-of.html

engine(s), whereas canonical clustering is persistent since "the folder structure is generated only once, and folder maintenance can be carried out at relatively infrequent intervals" [25]. (ii) For obvious computational reasons, the clustering engine processes only the (short and thus poor) web snippets and not their corresponding (long and thus informative) originating documents. (iii) Every folder must be assigned with a meaningful sentence in order to reflect in an intelligible way the "theme" of the snippets (and thus corresponding documents) contained into it. (iv) Folders must be organized in a hierarchy and this must be properly labeled to allow the post-navigation of the user for query refinement.

Web-snippet clustering *is* also a sort of document organization into topical contexts. But we have neither *structural information* [23] associated to each web-snippet, nor we can use *static classification* [1, 5] for them. Web-snippet clustering indeed operates on the huge, heterogeneous, uncontrolled web and requires the mining of the (poorly composed) snippets returned by search engines. Thus, any "fixed set of category labels" would be not flexible enough to capture the variegate and fine themes behind them [13].

As far we know, we are the first to suggest the use of web-snippet clustering as a tool for personalizing the ranked list of results returned by a meta-search engine. Our tool, called SnakeT, is an effective add-in of any (meta-) search engine because it offers: query refinement, disambiguation, knowledge extraction and personalized ranking. This last specialty builds upon an issue that we wish to stress in this paper: *it does exist between ranking and web-snippet clustering a mutual reinforcement relationship* from which both of them may benefit. SnakeT is particularly useful when the users submit informative queries [3], polysemous queries, or poor queries consisting of at most two terms (more than 80% of the Web queries are of this type!).[3] We engineered and fully tested SnakeT so that we can safely state that it is the first open-source tool for web-snippet clustering being competitive against commercial proposals.

2 Our Contribution in Context

In recent years there has been an upsurge of commercial interest [6] on the web-snippet clustering problem. This problem was solved in a primitive form by Northernlight, which stopped working in 2002! It exploited a complete 17,000-node subject taxonomy built by humans and classifying a pre-defined set of Web documents by subject, type, language and source! Since then, this technology has improved in its structure and efficacy and employed by some commercial softwares such as Vivisimo, Mooter, Copernic, iBoogie, Kartoo, Groxis, Dogpile and Clusty. They do not use a predefined taxonomy but cluster on-the-fly the snippets returned by commodity search engines. The efficiency of the technology has been recognized with the "best meta-search engine award" from 2001 to 2003 by SearchEngineWatch.com. Last year, Dogpile was voted as the most innovative IR-tool by a jury of about 570 web users.[4] Unfortunately, very little information is available about the technology underlying these tools.

The scientific literature offers various solutions to the web-snippet clustering problem [15, 29, 31, 27, 9, 20, 10, 19, 30, 17, 28, 22, 4]. In the simplest case, these solutions use a "bag of words" representation in which the indexing unit consists of a single word and the clustering is flat. In the more sophisticated case, the labels consist of longer sentences and the clustering is hierarchical (like Vivisimo). We survey below these approaches by classifying them into a taxonomy of existing solutions.

1. *Single words and Flat clustering.* Scatter/Gather [13] was one of the first web-clustering systems on the top of an IR engine. It may be considered to belong to this class even if it was not tested upon a Web search engine. Recently, various other systems were proposed in this class. WebCat [10] uses Transactional K-Means to produce the flat clustering. Retriever [15] uses robust relational fuzzy clustering. The system of [26] retrieves the snippets from a search engine and then expands this base set with all the in-linking and out-linking pages to improve precision. But search engines do not provide a cheap access to the web graph thus making this retrieval efficient if limited to a local (partial) copy available at clustering engine site. Among these softwares only WebCat [10] is available on-line.

2. *Sentences and Flat clustering.* Grouper [29] was the first academic meta-search engine to address the web-snippet clustering problem and belongs to this class. It used phrases of variable length to label the folders, but these phrases were drawn as *contiguous* portions of the snippets by means of a Suffix Tree data structure. Lingo [22] uses SVD to find meaningful long labels. Recently, a research group at Microsoft [30] proposed a system that extracts (contiguous) sentences of variable length via regression on five different measures. However regression

[3] Informative queries should provide different results over the time. For instance, SnakeT suggests "John Kerry" and "Bush" for the query "President Elections"and users can personalize ranked results, accordingly.

[4] Dogpile has been granted licence to Vivisimo's technology.

Figure 1: VIVISIMO and SNAKET on query "*asthma*"

needs a training phase, and it is not clear how to do this on the whole heterogenous Web. The authors did not make the system available for testing, and asked for a hierarchical clustering and external taxonomies to improve the precision of their extracted labels. This is actually what we do with our SNAKET. Among the softwares of this class, it is available on-line only CARROT2 [27], an open source implementation of GROUPER. The original GROUPER is no longer available, as the authors communicated to us.

3. *Single words and Hierarchical clustering.* FIHC [9] uses an analysis based on the Frequent Itemsets Problem in order to construct the folders and to organize them in a hierarchy. CREDO [4] uses a concept lattice on single words, and is the only system available on line for this class. MOOTER is a commercial engine which belongs to this class.

4. *Sentences and Hierarchical clustering.* This is the most interesting class including systems that try to mimic VIVISIMO. LEXICAL AFFINITIES CLUSTERING [20] was the first system to propose this approach. It improves precision over recall by using a *pair of words* (not necessarily contiguous) that are linked by a lexical affinity (LA), i.e. a correlation of their common appearance. The clustering phase is implemented using a variant of the classical HAC approach. In [29] Etzioni proposed a simple extension of GROUPER to hierarchical clustering based on the size of folders overlap. SHOC [31] uses instead a Suffix Array for (contiguous) sentences extraction and organizes the folders in a hierarchy via an SVD approach. HIGHLIGHT [28] adopts lexical analysis and a probabilistic framework for hierarchy construction, but the authors do not provide any evaluation. CIIRARCHIES [19] extracts meaningful sentences from the snippets by using a pre-computed language model, and builds the hierarchy via a recursive algorithm. The authors admit that their hierarchies are often non compact, have large depth and contain some non content-bearing words which tend to repeat. Recently, a research group at IBM [17] proposed a system that constructs the folder hierarchy based on the minimization of an objective function similar to the one we use in our SNAKET. However, their labels consist frequently of single words, whereas in just few cases are contiguous sentences. The authors did not make this system available for testing! Surprisingly enough, the only systems of this class available for testing are HIGHLIGHT and CIIRARCHIES; the others are no longer available on line, as the authors communicated to us. Our SNAKET belongs to this class, it is highly engineered, available on-line and widely tested, and it aims at overcoming the limitations of the systems above by using non-contiguous sentences as labels, by adopting some special knowledge bases for ranking and selection of folder labels, and by using a novel

covering algorithm to select meaningful and compact hierarchies.

Name	Word Flat	Sentences Flat	Word Hier.	Sentences Hier.	Online/ Software
WebCat	+				+
Retriever	+				
Scatter/Gather	+				
Wang	+				
Grouper		+			
Carrot		+			+
Lingo		+			+
Microsoft		+			
FICH			+		+
Credo			+		+
IBM			+		
SHOC				+	
CIIRarchies				+	+
LA				+	
Highlight				+	+
SnakeT				+	+
Mooter			+		
Vivisimo				+	+

Figure 2: Taxonomy of current solutions.

Given this scenario, we compared SnakeT against the softwares of the fourth class available on line: CIIRarchies, Highlight. As commercial engines we compare Mooter and Vivisimo. We also tested Carrot2 for historical reasons: it offers the only available implementation of Grouper. We remark that we did not test the most recent results of [30, 17] because: they didn't provide us with an access to their software; we could not repeat their experiments because the original datasets are missing and querying the same search engines for snippet extraction gives now different snippets (see section 5.2); and finally, we could not implement by ourselves these systems due to the lack of algorithmic details in those papers.

	Anecdotal	User Survey Num Users	User Survey Queries	Measures
WebCat			10	F1
Retriever	+			
S/G	+	13	3	
Wang				Accuracy
Grouper		7	3	precision
Carrot	+	14	3	click seq
LA				entropy
Microsoft		3	30	P@N
FICH				F1
Lingo		7	8	
Credo	+			
IBM	+	20	10	coverage
SHOC	+			
CIIRarchies		8	49	EMIN
Highlight	+			click seq
SnakeT	+	20	77	P@N

Figure 3: Main evaluation measures

The literature of web-snippet clustering offers three different methodologies for comparing the systems: anecdotal evidence for the quality of the results, user surveys conducted on a (small) set of users for a (small) set of queries, and some mathematical functions (see Fig. 3). However, there is not a general consensus about the measure to use for evaluating a web-snippet clustering engine. Moreover, although there are many proposals for evaluating flat clustering [11, 21], it is still open the definition of a mathematical evaluation which takes into account the *expressiveness* of the labels within a folder hierarchy. In section 5 we evaluate SnakeT via some user surveys, anecdotal evidence on systems quality, and by extending the methodology of [30] with the goal of addressing the "label expressiveness" issue. In the same section, we also address the lack of a standard dataset for evaluating web-snippet clustering engines by proposing a snippet collection that has been obtained by querying a set of 16 search engines on 77 queries, selected among the top searched on Lycos[5] and Google[6] during 2004. This dataset has been used to evaluate SnakeT and all of its modules. It is available on line [7] and can be used freely by the research community either to reproduce our experimental results or to test any new web-snippet clustering engines.

[5]http://50.lycos.com/

[6]http://www.google.com/press/zeitgeist.html

In summary, no academic result has achieved a performance comparable to the one provided by VIVISIMO; no one proposed a unifying hierarchical clustering system with a Web interface for both Web Search, Books, News and Blog domains; no one performed a large and extensive testing on all the existing web-snippet clustering tools; no one proposed a publicly available dataset for future comparisons; and no one suggested to use clustering for realizing a form of Personalized Web Ranking. These are the main achievements of SNAKET, whose software architecture (Fig. 4) is simple and open source.

3 Our contribution: SnakeT

We address the web-snippet clustering problem in its full generality by proposing a novel IR-tool, called SNAKET, which is based on four key ideas.[7]

(1) For the *sentence formation process*, SNAKET overcomes the "contiguity limitation" of GROUPER's labels and generalizes the notion of LAs [20] to more than two words. Indeed labels are drawn as sentences of the web-snippets where few terms are possibly missing.

(2) For the *sentences selection process*, SNAKET exploits two knowledge bases (KB) built at preprocessing time. One KB offers some *anchor-text information*, derived from a small crawling of the Web, to enrich the description of the snippets. The other KB indexes the high-quality directory DMOZ, with the aim of providing a ranking engine for the sentences extraction process (unlike [5]).

(3) For the *folder hierarchy formation and labeling*, SNAKET improves the expressivity of the folder hierarchy by solving a special covering problem that ensures the assignment of the labels in such a way that over any hierarchy's path they are *not necessarily one the substring of the other*.

(4) We exploit the *"labeled folder hierarchy" framework* to improve the efficacy of modern search engines. Precisely, we show that SNAKET is an effective add-in of any (meta) search engine because it offers in *one unique tool*: query refinement, disambiguation, knowledge extraction and personalized ranking. Folder labels act as up-to-date and fresh *concepts* distilled on-the-fly from the web snippets. Users can then select the concepts that best match the "intent" behind their queries, and SNAKET can so *"re-rank"* (personalize!) the results according to these choices. Furthermore, the browsing of the hierarchy allows the user to acquire several points of view on the "query results" on a potentially larger pool of snippets (about 200 or more) without the effort of scanning all of them. In this way lazy users are not just limited to look at first ten results!

SNAKET is accessible at `http://snaket.di.unipi.it/`. The user can query 16 search engines (about Web, Blog, News and Books) and get a display of the results in a way similar to VIVISIMO (see Fig. 8). On the left, SNAKET shows the hierarchy of labeled folders. On the right, SNAKET shows a flat list of results ranked according to GOOGLE, if the result has been returned by this engine, otherwise the ranking function exploits a humanly-fixed weight for each other engine. Fig. 4 provides an overall picture of SNAKET's architecture.

For space reasons, we concentrate on the three most innovative modules of SNAKET: the 2 knowledge bases (built at preprocessing time and used at query time for labels formation and ranking), and the hierarchy-formation algorithm (operating at query time). The design of the other modules exploits known algorithm ideas (e.g. inverted lists, PoS, Named Entities, asynchronous I/O for search-engine query) and thus will be not detailed here. We nonetheless notice that their engineering was tricky to achieve current SNAKET's performance which are actually competitive against commercial products, and far better than any other available academic system.

The Anchor-text Archive. The (anchor) text surrounding the hyperlinks that point to a web page p are known to provide a precise, yet concise, description of p [12]. Many search engines, like GOOGLE, exploit anchor texts for ranking and retrieval. SNAKET uses this information for clustering the snippets. However, since common search engines do not return anchor texts, we used the *Nutch's open source spider* to collect more than 200 millions Web pages selected among the top-cited ones. We extracted from these pages the anchor texts and then stored them in a DB indexed by their pointed URLs. At query time, SNAKET searches this DB for each page p returned by the 16 search engines. In the case the URL of p is present, then p's snippet is enriched with the (anchor) texts of the hyperlinks pointing to it.

The Label-Ranking Engine. There are many proposals in the literature to rank individual terms, either based on precomputed archives (e.g. ontologies, lexical databases, as WORDNET and APPLIEDSEMANTICS) or on frequency information, e.g. `TFxIDF`. SNAKET adopts a *mixed* strategy based on the directory *DMOZ* which is freely available, controlled by humans and thus of high-quality. DMOZ classifies more than 3,500,000 sites in more than 460,000

[7] We point out that a description of a preliminary prototype of SNAKET has been presented in the short paper [8]. The SNAKET discussed in the present paper is an engineered and fully tested software, with the Personalized Engine available, which is competitive against the commercial softwares.

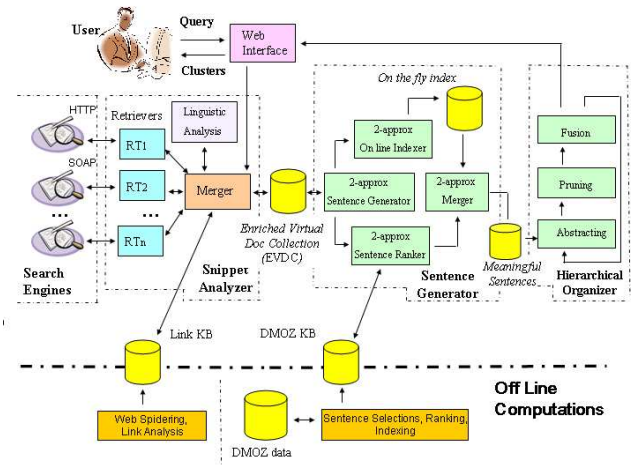

Figure 4: The architecture of SnakeT.

categories, and is used for ranking and retrieval by many Web search engines, like Google. We are the first, to the best of our knowledge, to use the whole DMOZ for Web snippets clustering.[8] We indexed the whole DMOZ and implemented a `TFxIDF` measure that is *DMOZ-category* centered.

How the labeled folder-hierarchy is built. SnakeT considers two snippet sentences identical even if some terms are missing or occur in different order. This way, the sentences "John Fitzegerald Kennedy", "Kennedy John" and "John F. Kennedy" are considered identical thus mimicking Vivisimo and improving the other academic tools [29, 20, 31, 27, 9]. To manage the case of "fake meanings" introduced by this dropping, SnakeT uses the two KBs previously mentioned.

We postulate that *"snippets sharing the same approximate sentence belong to the same folder"*. This produces a flat pool of folders labeled with a sentence called the *primary label* of the folder. If folders share many sentences, the primary label is the one that reports the *largest rank* among them. A snippet may occur in many folders, but this is consistent with the observation that a web page can cover *multiple themes*.

In order to generate the folder hierarchy SnakeT associates to each folder a *string signature* composed of the primary label and a *list of secondary labels*. The primary label are the approximate sentences associated to the folders: they are of *high quality* but often *too much specific* to generate the folder labels at the higher levels of the hierarchy. The secondary labels are approximate sentences which have a *good rank* and occur in *at least* the $c\%$ of the snippets contained in the folder. Secondary labels provide a folder description at a coarser level of detail (depending on c), and are thus more useful for the hierarchical formation and labeling. SnakeT forms the labeled hierarchy by merging folders into superfolders based on the substrings shared by their signatures. Namely, if the signatures of the folders $F_1, \ldots, F_z$ share the substring s, then SnakeT forms a new *candidate* super-folder SF having those folders as children, s as primary label, and secondary labels obtained by concatenating the secondary labels of $F_1, \ldots, F_z$. The rank of s is computed as a function of the rank of the (primary or secondary) labels having s as a substring. Notice that the substring s is not necessarily a contiguous phrase of some snippet because the signatures are approximate sentences. Among all the candidate super-folders SnakeT discards the *redundant* ones before repeating the process onto the next level of the hierarchy. The number of these redundant superfolders is not negligible and their deletion contributes to make the overall hierarchy more intelligible and compact. We formalize the discarding-problem as a *covering* problem that keeps into account three crucial factors for folder descriptions: label conciseness, accuracy and distinctiveness.[9] The covering problem is similar to the one proposed in [17], but here we use approximate sentences and secondary labels to obtain parent-child folders that share the fewest terms between their labels (see e.g. Fig. 7). Fig. 5 reports the time complexity of SnakeT and other engines. We notice that SnakeT's time complexity is $O(n \log n + m \log mp)$ by using a Suffix Array for comparing the signatures. Experiments showed that small values of m, p achieve excellent results.

[8] Unlike [5], we are using a whole web directory (DMOZ) only for the *ranking* of the candidate folder labels. And we remark that these labels are created on-the-fly from the snippets returned by the 16 search engines at query time.

[9] As suggested by Raul Valdez-Perez, President of Vivisimo.

Flat	Time Complexity	Hierarchical	Time Complexity
Retriever	$O(nk)$	LA	$O(n^2 \log n^2)$
Wang	$O(kn \log n)$	IBM	$O(kn)$
Grouper	$O(n)$	SHOC	$O(n)$
Carrot	$O(n)$		
Microsoft	$O(n)$	SnakeT	$O(n\ logn + m \log mp)$

Figure 5: Notation: n is the number of processed snippets, k is the number of desired folders, m is the number of extracted sentences/words, p is the number of labels extracted by SNAKET.

4 Personalizing Search Results

We investigate four ways of exploiting the "labeled folder hierarchy" framework for improving the efficacy of modern search engines. Precisely we show that SNAKET is an effective add-in of any (meta-) search engine that offers in *one unique tool*: query refinement, disambiguation, knowledge extraction and personalized ranking. This last specialty is innovative, as far as we know, and builds upon an issue that we wish to stress in this paper: *it does exist between ranking and web-snippet clustering a mutual reinforcement relationship* from which both of them may benefit. Indeed link-based ranking approaches tend to produce results which are biased towards the most popular meaning of an ambiguous query: "Jaguar" on GOOGLE does not get answers related to the mayan civilization in the first ten results. Conversely, SNAKET is able to distill from the web-snippets *few key concepts* (possibly some of low rank, see Fig. 6) that may be subsequently deployed by the user to *re-rank* (personalize!) the results produced by the underlying search engine. Of course, the better is the ranking, the better are the snippets from which SNAKET distills its key concepts, and thus the better should be the quality of the extracted labels and of the resulting folder hierarchy. SNAKET fetches about 200 snippets from the 16 search engines; this number is customizable.

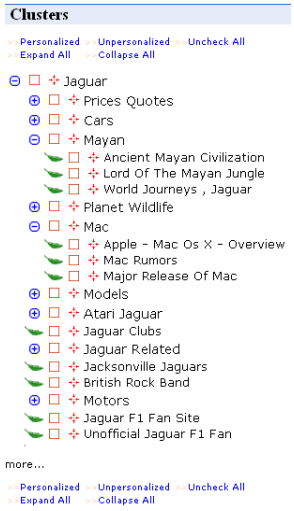

Figure 6: Knowledge extraction for "jaguar"

Hierarchy Browsing for Knowledge Extraction. Users can navigate over the hierarchy by expanding or collapsing on-the-fly its folders. The expansion is cheap since it occurs at the client side. The navigation can be seen as a form of *knowledge extraction* process that allows the user to acquire several points of view on the "query results" on a potentially larger pool of snippets (about 200 or more) without the effort of scanning all of them. This is useful for query disambiguation, in case of informative queries [3], and also because users frequently look at just the first top-ten results of the ranked-flat list. See Fig. 6 where a user learns from the hierarchy created for the query "jaguar" that this term refers to: an animal, a car, the mayan civilization, and Mac OS X.

Hierarchy Browsing for Results Selection. Users can narrow the ranked list of snippet results to those which generated a label l, by just clicking on l. This is pretty much similar to what VIVISIMO does, with the speciality that SNAKET does everything at the client side.

Query Refinement. Once the user looks at the folder hierarchy, she can decide to refine the query Q in two

different ways. Either she can deploy the folder labels for choosing *new keywords* for a *refined query* to submit to SNAKET. Or she can choose as *additional keywords* the terms of a label l, by clicking on it. In this latter case SNAKET submits automatically the refined query $Q' = Q \wedge l$ to the set of remote search engines, and then builds a new folder hierarchy for them. See Fig. 7 where the query "allergy" may be refined as "latex allergy" by clicking onto the label "Relief/Latex Allergy". This is a form of query expansion/suggestion used by many commercial search engines, here re-interpreted in the Web-snippet hierarchical clustering framework.

Figure 7: Hierarchy browsing for "Allergy"

Personalized Ranking. Users can select a set of labels $L = \{l_1, \ldots, l_f\}$ and ask SNAKET to *filter out* from the ranked list the snippets which do not belong to the folders labeled by L's labels. We think that this is the most useful feature offered by SNAKET's interface, in that *it allows to dynamically adapt the ranked list of (about 200 or more) results to the local choices made by any user*. The number of fetched snippets is customizable. As far we know, we are the first to suggest the use of web-snippet clustering as a tool for personalizing the ranked list of results returned by a meta-search engine. This feature turns out to be particularly interesting when the users submit informative queries [3], polysemous queries, or poor queries consisting of at most two terms (more than 80% of the Web queries are of this type!). See Fig. 9 where a user aiming at introductory material about the programming language "java", first formulates the query "java", and then selects the labels "Tutorials" and "Training" for getting *personalized results*.

Figure 8: SNAKET on the query "java"

Figure 9: Personalized SNAKET: the user clicks on "Tutorial" and "Training" and gets its *personalized* ranked list

It goes without saying that the literature offers many proposals for personalized ranking that exploit a collection of user profiles to customize the classical link-based ranking algorithms. However these proposals show two main limitations. (1) Although they exploit nice scaling techniques [14], they ultimately need to compute, for each Web page, a number of ranking values that is related to the number of user profiles! (2) Their application is limited to the knowledge of *up-to-date user profiles* which is a critical resource to be acquired and maintained. Currently GOOGLE [10] collects *category-based profiles* which have to be explicitly given and maintained by the users, and may refer only to a very small number of categories. YAHOO [11] requires an explicit `user login` and builds the profiles based on the users' activities.

Our proposal is instead *fully adaptive* and *non intrusive* both for the user and the queried search engine(s). SNAKET does not require neither an explicit `login` by the user nor a pre-compilation of a user profile, it also does not require neither a tracking of the user's past search behavior nor a modification of the underlying search engine(s) returning the web snippets. Specifically SNAKET works on the top of any un-personalized ranking algorithm (e.g. a meta-search engine, GOOGLE, ...) and allows the user to *personalize on-the-fly* the query results via a post-processing phase that involves the user interaction with the folder hierarchy (see above). This interaction implicitly distills the best "up-to-date user profile" since it is the same user that selects those folder labels L which best match her time-varying and subjective needs. Notice that the user can change the selected labels and thus modify on-the-fly her set of ranked results. The re-ranking is always done on all the 200 (or more) snippets returned by the queried search engine(s). Any choice of L labels changes the "view" of the users on all these snippets. Everything occurs at the client side, thus being computationally cheap. As said before this approach is intuitive and not offered by any known search engine!

Personalized Web Interface. We remark that SNAKET offers a lightweight Web-client interface that does not require to maintain any state on the server. For each query, the server performs the Hierarchical Clustering of the snippets returned by the queried engines and then sends to the Web client all the information needed to perform the above tasks via a one-time communication in XML. Folder expansion, browsing and personalization occur at the client side and thus result computationally cheap. Conversely, VIVISIMO requires a communication between the client and the server for each cluster refinement.

5 Experimental results

SNAKET runs currently on a commodity PC with Linux, a P4 CPU and RAM 1.5Gb. This is the system where we executed our tests. For space reasons we report just the most important results, whereas a more extensive testing is available on line, see [7].

In what follows we describe the experimental setting for SNAKET. See Fig. 10 for time figures. We remark that our evaluation deals with the three issues traditionally addressed in the literature: user surveys, SNAKET's performance test under various evaluation metrics, and anecdotal evidence of the overall efficacy of SNAKET and of its modules. This last evaluation deploys a unique (in the literature) dataset of snippets, enriched with clustering results, that we have collected from the 16 search engines and made available on-line [7].

[10] http://labs.google.com/personalized
[11] http://mysearch.yahoo.com/

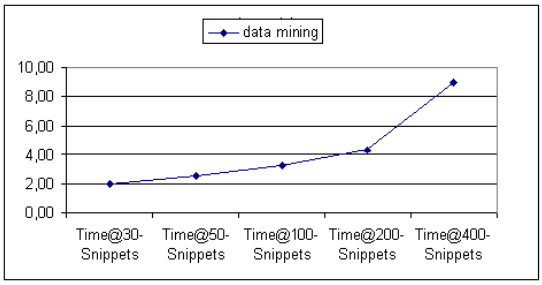

Figure 10: SnakeT's time for "data mining" over different number of snippets

5.1 Users Surveys

First Study: Is Web clustering beneficial? This study was aimed at understanding whether a Web-snippet clustering engine is a useful complement to the flat, ranked list of results offered by classical search engines (like Google). 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". This study confirms (!) the evaluation reported by SearchEngineWatch.com.

Second Study: SnakeT vs other academic systems. We selected 18 queries from our dataset 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 to three users to compare our results against those provided by Mooter, CIIRarchies, Highlight, Carrot2. For a large part of the queries the users did not like Mooter, since it provides folders labeled with single terms. Carrot2 often tends to create a number of folders which exceeds the number of snippets itself (!), thus impacting negatively onto the usability of such IR-tool. Carrot2 also fails to cluster together similar labels such as "knowledge, knowledge discovery", "mining and knowledge", and furthermore it labels the hierarchy paths with sentences which are one the substring of the other thus introducing few additional knowledge with the browsing. Highlight obtains its top-level's labels by using classification, so that they are few and of little use. Moreover, the clustering produces frequently identical subtrees under different top level categories and a number of folders which exceeds the number of snippets themselves (e.g 160 folders for the "iraq" query!). CIIRarchies provides good hierarchies but, as the authors admit [12], they are often not compact, have large depth and contain some non content-bearing words which tend to repeat. From the point of view of performance we remark that the two best available academic tools, CIIRarchies and Highlight, are significantly slower than SnakeT. We also remind to the reader that no other system is available on line for comparison, as discussed in section 2, and that our three-users survey is fair because of the objectivity of the negative comments these three users have drawn.

Third Study: SnakeT vs Vivisimo. We tried to draw a preliminary evaluation of our software against Vivisimo. We selected 20 students of the University of Pisa and asked them to execute the above 18 queries on these two engines. 75% of them were satisfied of the quality of our hierarchy and its labels. See Fig. 11 for details.

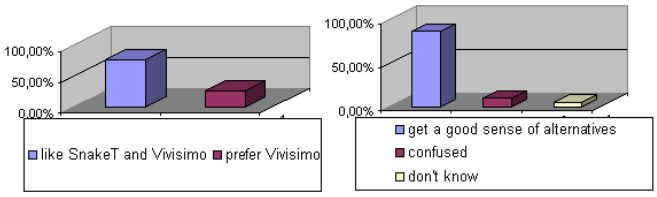

Figure 11: SnakeT vs. Vivisimo

[12] Pg 91-92 in [18] "A disadvantage present in both snippet and full text hierarchies is that they include uninformative topics. As the hierarchies become further developed, we will look for techniques to identify these uninformative topics."

5.2 SnakeT's dataset and anecdotal evidence

We have built a dataset consisting of the Web snippets collected by 16 search engines in response to 77 queries. It is available on-line at [7]. Queries are selected among the top searched on Lycos[13] and Google[14] during 2004. For each query we retrieved from 180 to 220 Web snippets. The dataset has been manually annotated by humans who judged the correctness of the results wrt the query. Moreover, the dataset has been enriched with the labels of the folders produced by SnakeT. This way the dataset can be used to infer anecdotal evidence about the quality of the folder hierarchy and to tune the software modules building up SnakeT (Figs. 6,7,8,9 are extracted from the dataset). As far we know, this dataset is the largest available on line and the only one retrieved by many search engines. It can be used freely by the research community either to reproduce our experimental results or to test any new Web clustering engine. This feature is crucial because the Web snippet clustering is a form of ephemeral clustering [20], and thus its results may change over the time as a consequence of a change in the list of answers returned by the search engines.

5.3 Evaluation of SnakeT

We run an extensive testing to tune up the different modules composing SnakeT. Because of the 10-page limit we are forced to report here few results, and thus we refer the reader to [24] for a full report. We evaluated SnakeT by using our dataset and a mathematical measure that extends the one adopted in [30] by taking more into account the folder hierarchy and its labels. We actually evaluate the precision at the first N labels associated to the top-level folders generated by SnakeT for each of the 77 queries. Precision at top N is defined as: $P@N = \frac{M@N}{N}$, where $M@N$ is the number of labels which have been *manually tagged* relevant among the N top-level labels computed by SnakeT. If a label l has been tagged as "ambiguous", we judge l relevant if the majority of its children labels are relevant (unlike [30]). We believe that P@N, specialized on the top-level folder labels, reflects the natural user behavior of considering these labels as the most important for hierarchy navigation. We use P@3, P@5, P@7 and P@10 since the lazy users does not like to browse a wider folder hierarchy.

Benefits of using DMOZ**.** The DMOZ index acts as a ranking engine for the approximated sentences to be selected and produces both a significative boost of P@N and an increase in the number of relevant top-level labels. In our experiments we noticed this phenomenon on a very large set of queries (see Fig. 12).

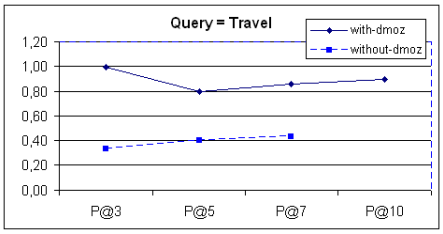

Figure 12: P@N using the DMOZ index

Benefits of using Anchors index. This index was introduced to enrich those sites which are indexed by search engines but do not have a good textual description. In this category fall sites with many images and almost no text, or those sites which belongs to the Web frontier reached by the spider. Moreover, we have here the top authorities which are described well by many other sites [16]. In Fig. 13 we report different values of P@N for the query "Yahoo", using or not using the anchors index. It may be observed a beautiful phenomenon common to many queries: The anchors increase P@N for the lower values of N.

Benefits of using multiple engines. Multiple engines offer a better coverage of the Web because of the low overlap of current search engines [2]. Usually this is seen as a limitation, rather than a resource, because of the difficulty in combining the multiple ranked lists. Fig. 14 reports a different view on this issue: the use of query results coming from many search engines produces a more detailed and meaningful hierarchy. This may better help the user in exploiting the various forms of personalization offered by SnakeT.

Benefits of using approximated sentences as folder labels. SnakeT considers two snippet sentences identical even if some terms are missing or occur in different order. These approximated sentences are useful when two related

[13] http://50.lycos.com/

[14] http://www.google.com/press/zeitgeist.html

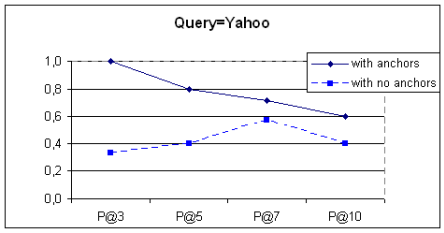

Figure 13: P@N using the Anchor index

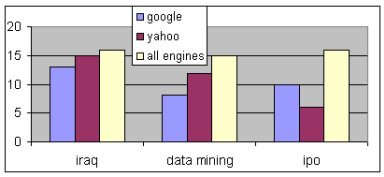

Figure 14: Number of top levels folders

words do not appear adjacent in the snippets collection. We then studied the distribution of the gaps for the most relevant pair of words in our dataset. Fig. 15 reports this distribution for the query "car rental", in logarithmic scale, and shows that the same word-pairs (plotted four word pairs) appear frequently within different text gaps either in order or reversed. Due to experimental observations, we then decided to adopt a maximum gap of four, for the words generating our approximate sentences.

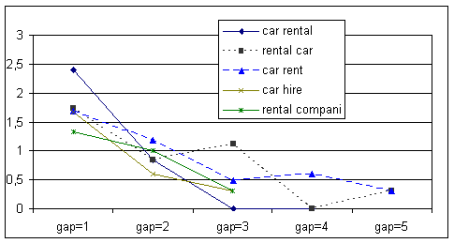

Figure 15: Distribution of word pairs

We are left with the study on the overall performance of SNAKET. Here, we comment on the most salient aspects and refer to Table 20 for details.

Label precision over our dataset. For all the 77 queries composing the dataset described in Section 5.2, we achieved an average precision over the labels of P@3=91%, P@5=83%, P@7=78% and P@10=79%, see Fig. 16.

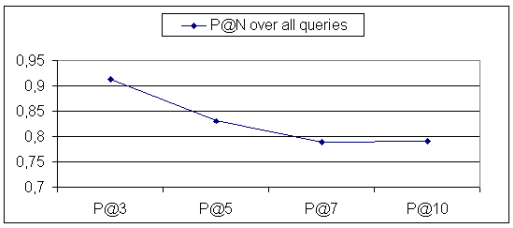

Figure 16: P@N on our dataset

Number of top-level labels. SNAKET was tuned to produce only meaningful top-level labels and to discard those below a fixed rank. Therefore not all of the queries produce ten labels (and hence ten folders) in the top level.

In Fig. 17 we report the exact number of generated top-level labels for all the 77 queries of our dataset. Notice that all queries produce at least three, and many of them produce up to ten, top-level labels.

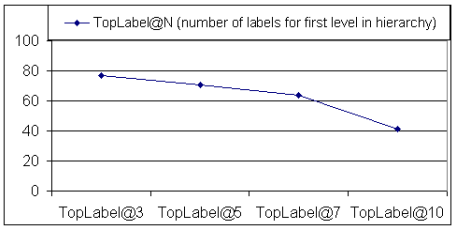

Figure 17: TopLabels@N for the Web Dataset

Precision over the personalized results. In sect. 3 we described the ranking adopted by SNAKET to aggregate the results of the 16 search engines, and in sect. 4 how SNAKET personalizes the search results. We studied how the precision changes when SNAKET's personalization is applied. We used *over the snippets* the following precision measure: $P@N_{snippets} = \frac{MS@N}{N}$, where $MS@N$ is the number of snippets which have been *manually tagged* relevant among the N top-snippets returned by SNAKET's personalization. We measured an increase in $P@N_{snippet}$ of about 22% averaged over N and our dataset of queries. In Fig. 18 we compare $P@N_{snippets}$ for (un)personalized results on query "divx".

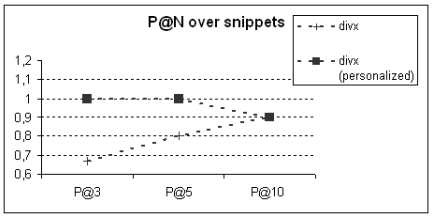

Figure 18: P@N over the snippets for "divx"

Number of web snippets contained into folders. The *weight of a folder* is the number of snippets it contains. A hierarchy is defined *weight balanced* if nodes at the same level have comparable weights. In Fig. 19 we report the distribution of the weights for the top-level folders generated for the query "data mining". For space constrains, we expanded only the top-folder "Software". Notice that SNAKET's hierarchy is balanced, and this phenomenon occurs on many queries of our dataset. We remark that a good balance is crucial for personalization since it enforces the folder hierarchy to equally detail all the concepts behind the query.

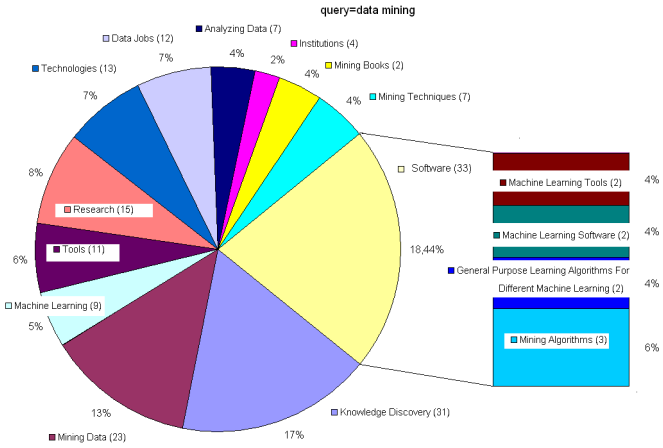

Figure 19: Covering Distribution for "data mining"

6 Conclusion

SnakeT is a unifying hierarchical web-snippet clustering system with a Web interface for Web Search, Books, News and Blog domains. Space constraints prevented us to describe the extension of SnakeT to the last three domains. Readers can nonetheless check on-line these features. Given the extensive experiments executed on SnakeT and the engineering of its modules, we can state that its time and quality performance are comparable to Vivisimo but SnakeT is open-source. In addition, SnakeT offers: query refinement, disambiguation, knowledge extraction and personalized ranking. The last specialty is innovative and builds upon an issue that we stressed in this paper: *it does exist between ranking and web-snippet clustering a mutual reinforcement relationship* from which they may benefit.

References

[1] G. Attardi, A. Gulli, and F. Sebastiani. Theseus: categorization by context. In *WWW8*, 1999.

[2] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *WWW7*, 1998.

[3] A. Broder. A taxonomy of web search. In *SIGIR Forum 36*, 2002.

[4] C. Carpineto and G. Romano. *Concept Data Analysis: Theory and Applications*. John Wiley & Sons, 2004.

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

[6] CNN.com. Better search results than Google? Next-generation sites help narrow internet searches. Associated Press, January 2004.

[7] SnakeT Dataset. URL: `http://roquefort.di.unipi.it/~gulli/listAllowed/testSnakeT/`.

[8] P. Ferragina and A. Gulli. The anatomy of a hierarchical clustering engine for web-page, news and book snippets. In *ICDM04*, 2004.

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

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

[11] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. In *JIIS*, 2001.

[12] T. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating strategies for similarity search on the web. In *WWW11*, 2002.

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

[14] G. Jeh and J. Widom. Scaling personalized Web search. In *WWW12*, 2002.

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

[16] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *JASM*, 1999.

[17] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *WWW13*, 2004.

[18] D. J. Lawrie. *Language Models for Hierarchical Summarization*. PhD thesis, University of Massachusetts, Amherst, 2003.

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

[20] 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.

[21] M. Meila. Comparing clusterings. Technical Report 418, University of Washington, 2002.

[22] S. Osinski and D. Weiss. Conceptual clustering using lingo algorithm: Evaluation on open directory project data. In *IIPWM04*, 2004.

[23] W. Pratt, M. A. Hearst, and L. M. Fagan. A knowledge-based approach to organizing retrieved documents. In *AAAI/IAAI*, pages 80–85, 1999.

[24] SnakeT Test Results. URL: `http://roquefort.di.unipi.it/~gulli/listAllowed/testing/`.

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

[26] Y. Wang and M. Kitsuregawa. On combining link and contents information for web page clustering. In *DEXA02*.

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

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

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

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

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

QUERIES	M@3	M@5	M@7	M@10
adsl	3	4	6	
aids	3	5		
airline flight tracking	3			
allergy	3	5	7	10
asthma	3	5	7	10
athens	2	2	3	
avril lavigne	3	5	6	8
bats	3	4	6	6
bible	3	5	6	8
britney spears	3	5	7	10
bush	3	4		
christina aguillera	3			
data mining	3	5	7	10
david beckham	2	4	6	8
divx	3	5		
dragon ball	3	3	5	8
dvd	3	5	7	10
dylan dog	3	3	3	5
eminem	3	5	7	
fbi	3	5	7	
final fantasy	3	5	7	10
final ipo	1	1	3	
firewall	3	5	7	
google	3	5		
grande fratello	3	5	7	8
guppy	3	3	4	
halloween	3	4	5	6
harry potter	3	5	7	10
hurricane	3	5	7	
ikea	2	2	2	
iraq	3	4	6	9
jaguar	3	4	6	
janet jackson	3	4	4	7
java	3	5	7	10
jennifer lopez	3	5	6	
kazaa	3	4	4	
las vegas	3	5	6	6
madonna	3	5	7	9
marjuiana	2	4	6	9
matrix	2	2	4	7
michelle vieth	3	4	4	
morpheus	3	5	7	9
mousetrap	3	3	5	6
movie	3	5	6	9
mp3	3			
music	3	5	7	10
napster	3	4		
nasa	2			
news	2	3	4	4
new york	3	5	5	6
nokia suonerie	2			
nostradamus	3	4	4	
pagerank	3			
perl	3	4	6	9
pink	2	3	3	
pisa	2	3	3	
pokemon	2	3	5	
retrovirus	2	3	3	5
sailor moon	3	5	5	5
samba	3	5	6	7
search	2	4	6	
seti	2	3	5	
shakira	3	4	6	
silvio berlusconi	2	4	4	
simpson	2	4	4	7
skateboard	3	5	6	9
sony	3	5	7	
spiderman	2	3	4	7
tattos	3	5	7	9
terrorism	3	5	7	10
time	3	5	6	8
travel	3	5	7	
vasco rossi	3	4	4	6
waterloo	3	5	6	7
winmx	2	2		
wtc	3	3		
wwf	3	4	6	7
	P@3	P@5	P@7	P@10
	91%	83%	78%	79%

Figure 20: M@N and P@N for SnaketT 's dataset