

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>



The European Journal for the Informatics Professional
<http://www.upgrade-cepis.org>

Vol. VIII, issue No. 1, February 2007

Publisher

UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by **Novática** (<http://www.ati.es/novatica/>), journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*, <http://www.ati.es/>)

UPGRADE monographs are also published in Spanish (full version printed; summary, abstracts and some articles online) by **Novática**

UPGRADE was created in October 2000 by CEPIS and was first published by **Novática** and **INFORMATIK/INFORMATIQUE**, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>)

UPGRADE is the anchor point for UPENET (UPGRADE European NETWORK), the network of CEPIS member societies' publications, that currently includes the following ones:

- **Informatik-Spektrum**, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
- **ITNOW**, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- **Mondo Digitale**, digital journal from the Italian CEPIS society AICA
- **Novática**, journal from the Spanish CEPIS society ATI
- **OCG Journal**, journal from the Austrian CEPIS society OCG
- **Pliroforiki**, journal from the Cyprus CEPIS society CCS
- **Pro Dialog**, journal from the Polish CEPIS society PTI-PIPS

Editorial Team

Chief Editor: Llorenç Pagés-Casas, Spain, pages@ati.es

Associate Editors:

François Louis Nicolet, Switzerland, nicolet@acm.org

Roberto Carniel, Italy, rcarniel@dgf.uniud.it

Zakaria Maamar, Arab Emirates, Zakaria.Maamar@zu.ac.ae

Soraya Kouadri Mostéfaoui, Switzerland,

soraya.kouadrimostefaoui@gmail.com

Rafael Fernández Calvo, Spain, rfcervo@ati.es

Editorial Board

Prof. Wolfried Stucky, CEPIS Former President

Prof. Nello Scarabottolo, CEPIS Vice President

Fernando Píera Gómez and

Llorenç Pagés-Casas, ATI (Spain)

François Louis Nicolet, SI (Switzerland)

Roberto Carniel, ALSI - Tecnoteca (Italy)

UPENET Advisory Board

Hermann Engesser (Informatik-Spektrum, Germany and Switzerland)

Brian Runciman (ITNOW, United Kingdom)

Franco Filippazzi (Mondo Digitale, Italy)

Llorenç Pagés-Casas (Novática, Spain)

Veith Risak (OCG Journal, Austria)

Panicos Masouras (Pliroforiki, Cyprus)

Andrzej Marciniak (Pro Dialog, Poland)

Rafael Fernández Calvo (Coordination)

English Language Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson

Cover page designed by Concha Arias Pérez

"Gaia gateway" / © ATI 2007

Layout Design: François Louis Nicolet

Composition: Jorge Llácer-Gil de Rames

Editorial correspondence: Llorenç Pagés-Casas pages@ati.es

Advertising correspondence: novatica@ati.es

UPGRADE Newslist available at

<http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

Copyright

© Novática 2007 (for the monograph)

© CEPIS 2007 (for the sections UPENET and CEPIS News)

All rights reserved under otherwise stated. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team

The opinions expressed by the authors are their exclusive responsibility

ISSN 1684-5285

Monograph of next issue (April 2007)

**"Information Technologies
for Visually Impaired People"**

(The full schedule of UPGRADE
is available at our website)

Monograph: Next Generation Web Search

(published jointly with Novática*)

Guest Editors: *Ricardo Baeza-Yates, José-María Gómez-Hidalgo, and Paolo Boldi*

- 2 Presentation. The Future of Web Search — *Ricardo Baeza-Yates, Paolo Boldi, and José-María Gómez-Hidalgo*
- 5 Efficient Sparse Linear System Solution of the PageRank Problem — *Gianna M. Del Corso, Antonio Gullì, and Francesco Romani*
- 12 Learning to Analyze Natural Language Texts — *Giuseppe Attardi*
- 19 SNAKET: A Personalized Search-result Clustering Engine — *Paolo Ferragina and Antonio Gullì*
- 27 The Multimodal Nature of the Web: New Trends in Information Access — *Luis-Alfonso Ureña-López, Manuel-Carlos Díaz-Galiano, Arturo Montejo-Raez, and M^a Teresa Martín-Valdivia*
- 33 Adversarial Information Retrieval in the Web — *Ricardo Baeza-Yates, Paolo Boldi, and José-María Gómez-Hidalgo*
- 41 GERINDO: Managing and Retrieving Information in Large Document Collections — *Nivio Ziviani, Alberto H. F. Laender, Edleno Silva de Moura, Altigran Soares da Silva, Carlos A. Heuser, and Wagner Meira Jr.*
- 49 Research Directions in Terrier: a Search Engine for Advanced Retrieval on the Web — *Iadh Ounis, Christina Lioma, Craig Macdonald, and Vassilis Plachouras*
- 57 Yahoo! Research Barcelona: Web Retrieval and Mining — *The Yahoo! Research Team*

UPENET (UPGRADE European NETWORK)

- 59 From **Novática** (ATI, Spain)
Informatics Profession
The Maturity of IT Professionalism in Europe — *Sean Brady*
- 68 From **Pro Dialog** (PTI-PIPS, Poland)
Graphical Interfaces
Portable Declarative Format for Specifying Graphical User Interfaces — *Zbigniew Fryźlewicz and Rafał Gierusz*
- 75 From **Novática** (ATI, Spain)
Next-generation Web
Blogs: On the Cutting Edge of the Next-generation Web — *Antonio Miguel Fumero-Reverón and Fernando Sáez-Vacas*

CEPIS NEWS

- 83 Harmonise Project: Building up to the Final Report—*François-Philippe Dragnet*
- 84 News & Events: European Funded Projects and News Updates

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

SNAKET: A Personalized Search-result Clustering Engine

Paolo Ferragina and Antonio Gulli

We propose a (meta-)search engine, called *SNAKET*, that queries 16 commodity search engines — specializing on the topics Web, blog, books and news — and then offers two complementary views on their returned results. One is the classical ranked list, the other one consists of a hierarchical organization of the results into folders labeled with variable-length sentences which are created on-the-fly at query time. These labels capture the "theme" of the query results contained into their associated folders. Users can eventually browse the labeled folder hierarchy with various goals: knowledge extraction, query refinement, or results personalization. This form of personalization is privacy preserving and non intrusive for the underlying search engines.

Keywords: Multi-Document Summarization, Personalized Web Ranking, Search Engines, Snaket, Web Snippet Clustering.

1 Introduction

Current search engines return a ranked list of web pages that contain the keywords of the user query together with some *contextual information* like the title and a text fragment summarizing the context of the searched keywords in each result page. This fragment is called a (*web-*)*snippet*. Recently, there has been a surge of interest in novel IR-tools that help the users in their difficult search task by means of novel ways for reporting the query results. "Search result clustering" is one of the most promising IR-tools [11], introduced in a primitive form by *NORTHERNLIGHT* and then made popular by *VIVISIMO*.

The problem solved by these tools consists of clustering the results returned by a (meta-)search engine into a *hierarchy of folders* which are *labeled* with variable-length sentences. The labels capture the "theme" of the query results contained into their associated folders. This labeled hierarchy offers a complementary view to the ranked list of results returned by current search engines. Users can then exploit this view by *navigating* the folder hierarchy driven by their search needs, with the goal of extracting information from the folder labels, or reformulating another query, or narrowing the set of relevant results. This navigational approach is especially useful for informative [1], polysemous and poor queries.

Search result clustering is a challenging variant of classical clustering because of two demanding requirements:

- The folder hierarchy and its labels must be computed *on-the-fly* in order to adapt themselves to the different themes of the results returned by the queried search engine(s). Canonical clustering is instead persistent since "hierarchical structure is generated only once, and folder maintenance can be carried out at relatively infrequent intervals" [12].

- The construction of the labeled folder hierarchy must deploy the heterogeneous, uncontrolled, and poorly composed information contained in the web-snippets returned by commodity search engine(s).

Many commercial systems implement nowadays the

Authors

Paolo Ferragina is Associate Professor of Computer Science at the University of Pisa. He got his PhD in Computer Science from the same University, and his Post-doc from the Max-Planck Institut für Informatik (Germany). His research is mainly devoted to the design, analysis and experimentation of algorithms and data structures for storing, compressing, mining and retrieving information from large amounts of data. His research results received one US Patent and some international awards from IEEE Vehicular Technology Society, EATCS and Philip Morris. He has served as PC member and co-chair of several international conferences. He has been plenary speaker at CPM '04 and SPIRE '05, and the (co)editor of two special issues on Theoretical Computer Science and Theory of Computing Systems. He is the author of more than 80 international publications, and he is leading various international projects, one of them is the first European Yahoo-Research academic grant. <ferragina@di.unipi.it>.

Antonio Gulli is the Director of Advanced Search Projects at Ask.com, a market leading Web search engine. Ask.com belongs to IAC/InterActiveCorp, a Fortune 500 company. In 2006 he got a PhD in Computer Science from the University of Pisa. In 1998 he founded "Ideare", a European Web search engine company, that has been sold to Tiscali S.p.A., a leading European ISP. His research is mainly devoted to Web Information Retrieval, Data Mining, Parallelism and efficient ways for Managing Terabytes of data. He has served as PC member of several international conferences including WWW2007, AIRWeb07, TextLink-2007, ECIR2007, LinkKDD-2006, and LinkKDD-2005. Currently he is the Head of first European R&D centre of Ask.com. This centre delivered the state-of-the-art Ask.com's Image Search Engine and has been involved in the development of the Blog and Feed search engines. In 2006, IAC/InterActiveCorp assigned the IAC Horizon Award for the achieved results. <agulli@ask.com>.

web-snippet clustering technology in their (meta-)search engines (see Figure 1): *VIVISIMO*, *MOOTER*, *COPERNIC*, *iBOOGIE*, *KARTOO*, *GROXIS*, *DOGPILE* and *CLUSTY*. *GOOGLE* and *MICROSOFT* seem also to be interested into this IR-tool [19] [18]. *ASK.COM* is adopting clustering as part of its *Zoom* technology. Unfortunately, very little information is available about such industrial software. On the other hand, the scientific literature offers several detailed solutions to the



Figure 1: VIVISIMO (left) and SNAKET (right) on the Query "asthma". Notice on top of SNAKET's window the buttons for invoking the personalized ranking.

web-snippet clustering problem, but their performance is far from the one achieved by VIVISIMO.

Another approach to help users in searching the Web is the *personalization* of the ranked lists of query results. Personalized ranking combines web-graph linking information with some contextual/profiled information with the goal of achieving *adaptivity* and *scalability* to the variegated user needs. Examples of industrial personalized services are offered by GOOGLE, YAHOO, ASK.COM and EUREKSTER. In the scientific literature the personalized ranking problem has been investigated by proposing interesting scaling techniques for the classical link-based ranking approaches [3] [8] [13]. However these approaches either allow profiles over a restricted (tiny) set of choices, or they need to maintain up-to-date profiles which are a critical and private resource.

The contribution of this paper¹ is twofold: (i) We propose a software for web-snippet clustering, called SNAKET, that achieves efficiency and efficacy performance close to VIVISIMO; (ii) we exploit the labeled folder hierarchy of SNAKET to design a form of personalized ranking that is fully-adaptive, privacy preserving, and scalable to an unbounded number of user needs. Overall this shows that between ranking and web-snippet clustering does exist a mutual reinforcement relationship from which both of them may benefit.

¹ This work was done while the second author was a PhD student at the Dipartimento di Informatica, University of Pisa. Part of this work appeared in the *Proceedings of WWW 2005*.

The rest of the paper is organized as follows. In Section 2 we describe SNAKET's architecture and comment the algorithmic specialties underlying its design and implementation. In Section 3 we discuss our proposal for personalizing the search results returned by a commodity search engine based on SNAKET's labeled folder hierarchy. Finally, in Section 4 we present some experimental results, and compare the performance of SNAKET against VIVISIMO.

2 The Anatomy of SnakeT

The scientific literature offers various academic and scientific solutions to the web-snippet clustering and the personalized ranking problems. We refer the interested reader to [4] and the literature contained within.

The name of our software, SNAKET, stands for SNIppet Aggregation by Knowledge ExtracTION. Its architecture is detailed in Figure 2 where all of its modules and their interrelations are illustrated. We can actually identify three main algorithmic phases in SNAKET's functioning: sentence selection and ranking, hierarchical clustering and labeling, and personalized ranking. The first two phases will be detailed in this section, personalization will be discussed in the next section.

2.1 Two Knowledge Bases

The Anchor Text Database: Several search engines exploit the hyperlinks among web pages as a source of information for ranking and retrieval. In this paper, we use this information for the *clustering and labeling of the snippets*. To achieve this goal, we model 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 surround-

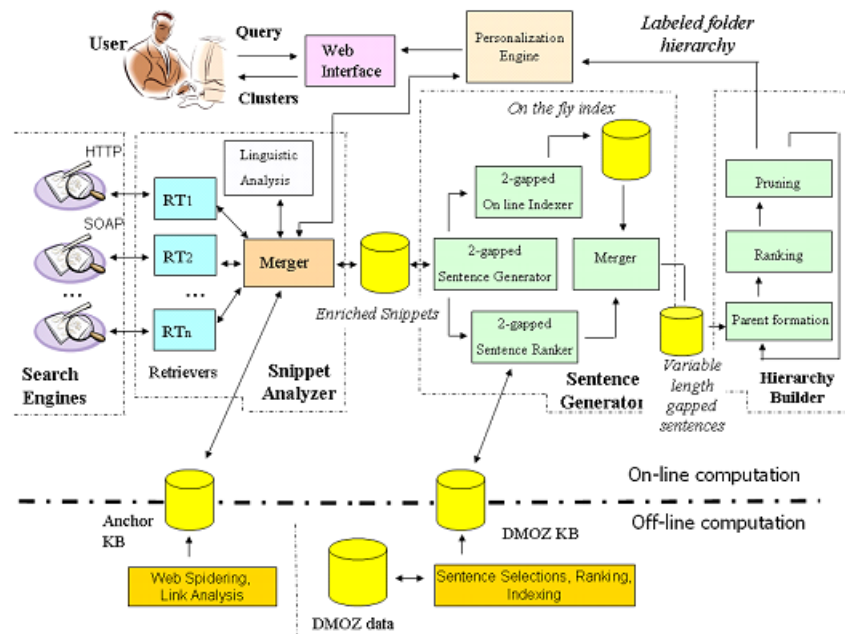


Figure 2: The Architecture of SNAKE T.

ing each hyperlink that points to p (about 20 terms). We refer to $A(p)$ as the *content* of p , and to $B(p)$ as the *context* of p . The virtual document for p is then given by $A(p) \cup B(p)$. We build the A_{link} index which collects the anchor texts drawn from more than 50 millions web pages, crawled using the *Nutch*'s open source spider and selected among those ones which were top-cited during the crawling process.

The Semantic Knowledge Base: One of the key ingredients to effectively extract meaningful folder labels is a method that *rank*s and *select*s a good set of candidate sentences drawn from the snippets of the query results. We indexed the directory DMOZ, freely available on the web, and calculated an ad-hoc TFxIDF measure for its controlled and high-quality lexicon.

We are the first, to the best of our knowledge, to use the whole DMOZ for web-snippet clustering.¹ Specifically, we indexed this valuable archive by developing a *ranking engine*, hereafter denoted by R_{dmz} , based on inverted lists. The lexicon of R_{dmz} consists of more than three million single terms.

Let $\#(t)$ be the total number of occurrences of the term t into DMOZ, $\#_c(t)$ be the number of DMOZ categories in which t appears, and let $\#_c$ be the total number of DMOZ categories. Moreover let $ns(C_i)$ be a boosting factor for the category that takes into account its depth in the DMOZ

hierarchy (i.e. we are postulating an increased importance for deeper categories), and $b(t, C_i)$ be a boosting factor for the term t if it is placed in a *relevant* part of C_i such as e.g. its description or title (i.e. we are postulating an increased importance of terms in crucial parts of the category's description). We define slight variants of the classical TF-IDF measure:

$$TF(t)=1+\log\#(t), IDF(t)=\log\frac{\#_c}{\#_c(t)}.$$

The rank of term t with respect to a category C_i is thus defined as: $R(t, C_i) = b(t, C_i) * TF(t) * IDF(t) * ns(C_i)$. The rank for a sentence $s_{h,k}$ formed by two terms (t_h, t_k) is then defined as: $R(S_{h,k}) = \max_{C_i} R(t_h, C_i) * R(t_k, C_i)$.

Finally, the rank of a longer sentence s is defined by identifying the set $P(s)$ of pairs of *contiguous* terms within s , and then taking the sum of their ranks:

$$R(s) = \sum_{S_{h,k} \in P(s)} R(S_{h,k}).$$

2.2 The First Module: Snippet Analysis

SNAKE T fetches the snippets from 16 search engines thus offering a large coverage of the web [6]. They are A9.COM, ABOUT, ALLTHEWEB, ALTAVISTA, ASK.COM, E-SPOTTING, FINDWHAT, GIGABLAST, GOOGLE, LOOKSMART, MSN, OVERTURE, YAHOO, GOOGLE NEWS, A9, and BLOGLINE. All of these search engines are queried by the so called *Retrievers*, that return a ranked list of results (with their snippets). The average retrieval time is of about 2 secs [5].

Snippets referring to the same page are merged, and then enriched with the anchor texts pointing to each result page

² Unlike [2], we are using DMOZ only for the *ranking* of the sentences which are extracted on-the-fly from the snippets.

p , retrieved from the A_{link} index. Subsequently, the enriched snippets are segmented into sentences by choosing as separators the punctuation symbols and HTML tags.

The sentences are finally processed by a *Linguistic Analyzer* which (i) filters a *stop list* of about 2000 words belonging to 12 different languages, (ii) stems the resulting sentences by using the Snowball stemmers [17], and (iii) extracts Part of Speeches (PoS) and Named Entities.

2.3 The Second Module: Sentences Selection and Ranking

We aim to compute folder labels that identify the snippets themes and thus are long and intelligible sentences, rather than single terms. It is well known [14] [9] that the extraction of sentences formed by *not contiguous terms*, hereafter called *gapped sentences*, may boost the precision of the folder labels and, in turn, the usefulness of the folder hierarchy to humans. For example, if a snippet contains the phrase "George W. Bush, the President of USA", its gapped sentences of length three are: "George Bush President", "George Bush USA", "George President USA", "Bush President USA". Our approach to sentences selection and ranking is syntactic, works in three phases and considers also the word-based permutations of the gapped sentences.

First phase: formation of 2-gapped sentences. From each sentence s of a snippet, we build the set $AS^2(s,d)$ formed by all pairs of terms that occur in s within a proximity d (like [9]). If a term pair (w_p, w_k) occurs in s , we insert it in $AS^2(s,d)$ together with its transpose (w_k, w_p) . Of course, some of the term pairs in $AS^2(s,d)$ could introduce artificial meanings. For instance, the sentence "Alexei Abrikosov, Vitaly Ginzburg and Anthony Legget received the nobel prize ..." derives the 2-gapped sentence "Vitaly Legget" which is probably an artificial name. To circumvent this problem, we *discard* from $AS^2(s,d)$ any pair whose terms appear contiguously neither in a snippet nor in DMOZ.

Second phase: selection and ranking of 2-gapped sentences. The 2-gapped sentences are ranked using *local* information and *background* information. Local information is a set of statistics collected from the snippets about a 2-gapped sentence, such as the number of occurrences of a 2-gapped sentence within the snippets, its HTML contexts, and the rank attributed to the snippets by the queried search engines. Background information is a judgement on the relevance of a 2-gapped sentence derived from R_{dmz} (see Section 2.1). The 2-gapped sentences in $AS^2(s,d)$ having rank below a given threshold are discarded.

Third phase: building longer gapped sentences. The fast construction of longer and intelligible gapped sentences is based on an inverted-list data structure built *on-the-fly* over the 2-gapped sentences of $\cup_s AS^2(s,d)$. We build the longer 4-gapped sentences of by picking two 2-gapped sentences of AS^2 , say (w_p, w_k) and (w_x, w_y) , and merging them if they have same `snipID` and occur in the same sentence

within a fixed proximity window. The merging operates via a linear scan of the two sorted inverted lists associated with (w_p, w_k) and (w_x, w_y) . All the 2-gapped sentences which do not participate into any merging are in some sense *maximal* and thus they are inserted in a candidate list, hereafter denoted by CL . Finally, we rank the elements of AS^2 via R_{dmz} . Those sentences which have a rank below a given threshold are discarded. The process is iterated until the set CL contains all maximal gapped sentences of length at most $2^3=8$ whose rank is above a given threshold. Notice that every gapped sentence has associated the snippets from which it has been extracted.

2.4 The Third Module: Labeled Hierarchy Formation

SnakeT uses an innovative bottom-up hierarchical clustering algorithm whose aim is to construct a folder hierarchy which is *compact* in terms of total number of folders, *balanced* in terms of subtree sizes, and *overlapping* in that a snippet may cover multiple themes and thus belong to many leaf folders. SnakeT also aims at assigning folder labels that are *accurate* with respect to snippets' themes, *distinct* to avoid an overwhelming repetition in their constituting terms, and *intelligible* by means of variable-length sentences. The overall process is called *hierarchy builder* in Figure 2.

Initially, snippets are grouped into folders according to the gapped sentences of CL they share. These folders provide the leaves of our hierarchy, and their shared sentences provide their labels (called *primary labels*). We denote by $\ell(C)$ the primary label of folder C , and this is used to annotate the folder. We are postulating that *snippets sharing the same gapped sentence deal with the same theme*, and thus must be clustered into the same (leaf) folder.

In order to agglomerate folders for hierarchy construction, SnakeT enriches each folder C with a *set of secondary labels*, defined as gapped sentences that occur in a fraction c of the snippets contained into C (currently $c=80\%$). The intuition is that primary labels provide a finer description of folders, whereas secondary labels provide a coarser description of folders.

To manage primary and secondary labels efficiently, we concatenate them into a unique string, called the *signature* of the folder C , and hereafter denoted by $\text{sig}(C)$. The inductive step of the bottom-up hierarchy construction consists of three main phases detailed below.

Parent Formation. A *parent* folder P is created for each group C_1, C_2, \dots, C_j of folders that share a substring among their signatures (hence, both primary and secondary labels are deployed). The shared substring provides the primary label of P , and thus its annotating sentence $\ell(P)$. The set of secondary labels of P is formed by taking the labels of the C_i 's that occur in at least the $c\%$ of P 's snippets. $\text{sig}(P)$ is finally obtained by concatenating the primary label of P with all of its secondary labels. Since $\ell(P)$ is computed among both primary and secondary labels which are gapped



Figure 3: SNAKET on the Query "java".

sentences, $\ell(P)$ is a gapped sentence too, being not necessarily a substring of the primary labels $\ell(C_1), \dots, \ell(C_j)$.

Ranking. The parent folders built at the previous step, are just possible *candidates* of the final folder hierarchy. SNAKET ranks these candidate folders by exploiting the ranks of the labels of their children folders computed via R_{dmoz} (see Section 2.1). The rank adopted is a sum of the ranks of all the labels (primary and secondary) contained in the signatures.

Pruning. SNAKET builds a *weighted bipartite graph* G in which the two sets of vertices are given by the (candidate) parent folders and their children folders, the edges of G denote the parent-child relationship between these folders, and the weight of a vertex is the rank of the corresponding folder. The graph G is exploited to clean up the (candidate) parent folders that do not match the goals stated at the beginning of this section. The number of the *redundant* parent folders is not negligible and their deletion contributes to make the overall hierarchy more intelligible. We formal-



Figure 4: Personalized SNAKET. The user selects the two labels "Tutorial" and "Training" and obtains her *personalized* ranked list (on the right).

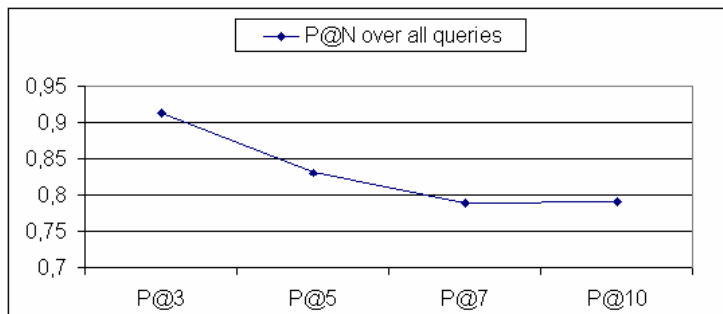


Figure 5: P@N on our Dataset.

ize this problem as a *covering* problem and solve it via a greedy approach which takes in account the ranking of the folders and their labels. Note that, at the end of the process, orphan children are put in a generic category "Other". SNAKET adopts two rules to detect and discard the redundant (candidate) parent folders:

- Context pruning rules.** It aims at discarding parent folders which are redundant with respect to a *graph-covering relation*: if two parent folders share (almost) the same children folders, then SNAKET keeps the parent folder having the largest rank. Moreover, a parent folder is discarded if it has *more than* a fixed number of children. These rules take into account the maximum overlap between folders, the maximum size of the folders, the balancing of the hierarchy.

- Content pruning rule.** It aims at discarding parent folders which are redundant with respect to a *syntactic-similarity relation* among their labels: if two parent folders are annotated with (almost) the same labeling terms, then SNAKET keeps the parent folder having the largest rank. This rule takes into account hierarchy compaction and folders overlapping. After the pruning phase, the remaining parent folders provide the next level upon which the bottom-up process is repeated. This process is stopped after that three levels have been built, because a deeper hierarchy would be not user friendly!

3 Personalizing Search Results

Link-based ranking algorithms tend to produce results which are biased towards the most popular meaning of an ambiguous query. At the time of this writing, "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 snippets *few key concepts* (possibly some of low rank, see Figure 4) that may be subsequently deployed by the user for many activities like query refinement, disambiguation, knowledge extraction, and even *personalization* of the ranked list of results produced by the underlying search engines.

Hierarchy browsing for knowledge extraction. Users can navigate through the hierarchy by expanding or collapsing on-the-fly its folders. The expansion is cheap since it occurs at the client side. This navigation can be seen as a form of *knowledge extraction* process that allows the user to acquire several points of view on the 200 or more query results fetched by SNAKET, without the effort of scanning all of them. This is useful because users frequently look at just the first top-ten results of the ranked list, and they issue poor queries mainly consisting of at most two terms.

Hierarchy browsing for results selection. Users can narrow the ranked list of results to those ones which generate a label *l*, by just clicking on *l*. This is pretty much simi-

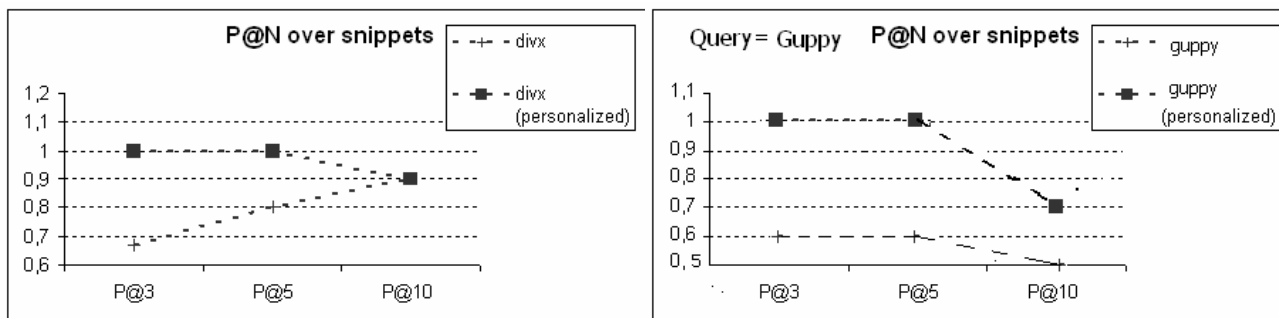


Figure 6: P@N for the Queries "divx" and "guppy".

lar to what VIVISIMO does, with the speciality that SNAKET does the narrowing at the client side.

Query Refinement. Once the user looks at the folder hierarchy, (s)he can decide to refine the query Q in two different ways. Either (s)he can take inspiration from the folder labels to compose a *new* query and then submit it to SNAKET. Or (s)he can click on some folder label l and then submit to SNAKET the *refined query* $Q \cup l$. This is a form of *query expansion/suggestion* used by many commercial search engines such as ASK.COM, here re-interpreted in the web-snippet hierarchical clustering framework.

Personalized Ranking. When a user issues a query, SNAKET generates the labeled folder hierarchy and the ranked list of results. The former is computed as we largely commented in the previous sections. The latter is produced by aggregating the query results provided by the 16 search engines according to a *static rule* which weights differently these 16 engines.

Users can select a set of labels $L = \{l_1, \dots, l_j\}$ from the hierarchy by clicking on the checkbox associated to each of them. Then, they may ask SNAKET to *filter out* from the ranked list of results the snippets which do not belong to the folders labeled with any of L 's labels. We think that this is an interesting feature offered by SNAKET's interface, because it allows to dynamically adapt the ranked list of (about 200 or more) results to the local choices made by the user. This feature turns out to be particularly effective when the users submit informative [1], polysemous, and poor queries. See Figure 4 for an example of personalized ranking in which 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*.

We note here that SNAKET's personalization is fully adaptive, scalable, privacy preserving and non intrusive for the underlying search engine. It is fully adaptive and scalable because it is not based on a user profile, and users can adapt the choice of their selected labels according to their subjective and time-varying interests. SNAKET also protects the user privacy because it does not require an explicit login, a pre-compilation of a user profile, or the tracking of the user's past searches. In short, SNAKET may be considered as a plug-in that turns any un-personalized (meta-)search engine into a personalized one, in an not intrusive way.

4 Experimental Results

SNAKET runs currently on a commodity PC with Linux, P4 CPU and RAM 1.5Gb. Its web interface is accessible at <http://snaket.di.unipi.it>. In what follows we report the most important results obtained in our experiments, and refer the reader interested in the whole figures to [16] [4]. These evaluations deploy a unique (in the literature) dataset of snippets, enriched with clustering results, that we have collected from our 16 search engines using 77 queries, selected

from the top searched ones on LYCOS and GOOGLE during 2004. This dataset is available on-line [16] and can be freely used by the scientific community either to reproduce our experiments or to test any new web-snippet clustering engine. We extensively tested SNAKET by following two main methodologies proposed in the literature: user surveys conducted on a set of queries, and some mathematical evaluations. There are many proposals for evaluating a flat clustering [7, 10], but it is still open the definition of a measure which operates on a labeled hierarchical clustering and is thus able to take into account the *expressiveness* of the folder labels [10]. Below, we propose an extension to the methodology of [15] with the goal of addressing the "label expressiveness" issue.

Users surveys: We selected 20 students at the University of Pisa and asked them to execute those queries on these two engines. 75% of them were satisfied of the quality of SNAKET's folder hierarchy and of its labels. Moreover, 82% of them have got a *good sense of alternatives from SnakeT*. Hence we can state that SNAKET achieves performance close to VIVISIMO.

Evaluation of SNAKET: We evaluated SNAKET by using our dataset and a mathematical evaluation measure that extends the one adopted in [15] by taking into account the *expressiveness* of the labeled folder hierarchy. For each one of the 77 queries, we evaluated the precision at the first N labels associated to the top-level folders generated by SNAKET. Precision at top N is defined as:

$$P@N = \frac{M@N}{N} \text{ where } M@N \text{ is the number of labels}$$

which have been *manually tagged* relevant among the N top-level labels computed by SNAKET. If a label has been manually tagged as "ambiguous", we judge it relevant if the majority of its children labels are relevant. We believe that $P@N$, specialized on the top-level folder labels, reflects the natural user behavior of considering these top-level labels as the most important for hierarchy navigation. We use $P@3$, $P@5$, $P@7$ and $P@10$ since lazy users do not like to browse a wider folder hierarchy. Notice that all queries produce at least three, and many of them produce up to ten, top-level labels. As far as the precision of SNAKET's top-level labels is concerned, we observe that on all queries of our dataset, SNAKET achieved an average precision of $P@3=91\%$, $P@5=83\%$, $P@7=78\%$ and $P@10=79\%$, see Figure 5.

Precision over the personalized results. We studied how the precision changes when SNAKET's personalization is applied, by using the following measure: $P@N_{snippets} = \frac{MS@N}{N}$, where $MS@N$ is the number of snippets which have been *manually tagged* as relevant among the N top-ranked snippets returned by SNAKET's personalization. We experienced an increase in $P@N_{snippets}$ of about 22% averaged over N and over all 77 queries of our dataset.

In Figure 6 we compute on $P@N_{snippets}$ personalized vs. unpersonalized results for the queries "divx" and "guppy".

5 Conclusion

SnakeT is a unifying hierarchical web-snippet clustering engine for many kinds of document collections: web, books, news and blogs. The use of SnakeT goes much beyond the Web, in that it may find useful applications also in enterprise search engines whose list of query results is much *long and undifferentiated* because their indexed documents lack of cross-linking patterns. In these contexts *pre-retrieval tagging* techniques are frequently used: they assign, *manually* or via a *software*, to the indexed documents a static set of (meta-)tags that concisely describe their content or other (meta-)information (like the author, the data, the language,...).

The labeled hierarchy obtained from these *pre-retrieval tagging* techniques is nonetheless limited because of its cost (human input occurs in at least one stage), poor quality (i.e. static taxonomy), limited flexibility (wrt multiple document sources).

Conversely the post-retrieval clustering techniques, like the ones implemented by Vivisimo and SnakeT, offer some advantages over these approaches which have been largely discussed in this paper. Given the experiments and the engineering of SnakeT's modules, we can safely state that its time performance and the quality of the labeled folder hierarchy are comparable to Vivisimo's ones. Additionally SnakeT provides an interesting form of personalized ranking that is fully adaptive to user needs, privacy preserving, scalable to the number of users, and non intrusive for the underlying (un-personalized) search engines.

References

- [1] A. Z. Broder. A taxonomy of Web search. In *SIGIR Forum* 36(2), pages 3–10, 2002.
- [2] H. Chen and S. T. Dumais. Bringing order to the Web: automatically categorizing search results. In *Proceedings of SIGCHI00*, pages 145–152, The Hague, The Netherlands, 2000.
- [3] P. A. Chirita, D. Olmedilla, and W. Nejdl. PROS: A personalized ranking platform for Web search. In *Proceedings of 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 34–43, Eindhoven, Netherlands, 2004.
- [4] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Journal of Software: Practice and Experience*, 2007, to appear.
- [5] A. Gulli and A. Signorini. Building an open source meta search engine. In *Proceedings of 14th International World Wide Web Conference*, pages 1004–1005, Chiba, Japan, 2005.
- [6] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Proceedings of 14th International World Wide Web Conference*, pages 902–903, Chiba, Japan, 2005.
- [7] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. In *Journal of Intelligent Information Systems*, pages 107–145, 2001.
- [8] G. Jeh and J. Widom. Scaling personalized Web search. In *Proceedings of the 12th International World Wide Web Conference*, pages 271–279, Budapest, HU, 2003.
- [9] 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.
- [10] M. Meila. Comparing clusterings. Technical Report 418, University of Washington, 2002.
- [11] J. Mostafa. Seeking better Web searches. *Scientific American*, February 2005.
- [12] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.
- [13] Tamas Sarlos, Andras A. Benczur, Karoly Csalogany, Daniel Fogaras, and Balazs Racz. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. In *Proceedings of 15th International World Wide Web Conference*, Edinburgh, Scotland, U.K., 2006.
- [14] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. In *Proceedings of 8th International World Wide Web Conference*, pages 1361–1374, Toronto, Canada, 1999.
- [15] H. Zeng, Q. He, Z. Chen, and W. Ma. Learning to cluster Web search results. In *Proceedings of SIGIR04*, pages 210–217, Sheffield, U.K., 2004.
- [16] <<http://roquefort.di.unipi.it/~gulli/listAllowed/testSnakeT/>>.
- [17] <<http://snowball.tartarus.org/>>.
- [18] <http://www.betanews.com/article/Microsoft_Tests_Search_Clustering/1106319504>.
- [19] <<http://www.searchengine slowdown.com/2004/10/Web-20-exclusive-demonstration-of.html>>.