Università di Pisa
Dipartimento di Informatica

Technical Report: TR-04-05

The Anatomy of a Clustering Engine for Web-page Snippets

Paolo Ferragina
ferragina@di.unipi.it

Antonio Gulli
gulli@di.unipi.it

January 29, 2004

Address: via F. Buonarroti 2, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

The Anatomy of a Clustering Engine for Web-page Snippets

Paolo Ferragina
ferragina@di.unipi.it

Antonio Gulli
gulli@di.unipi.it

January 29, 2004

Abstract

Recently there has been a surge of commercial interest about novel IR-tools, like Vivisimo or Groxis, that support the user of a search engine in his/her query formulation and query refinement. The basic idea is that the *snippets* returned by the search engine are grouped into *clusters* which are then organized in a *hierarchy* whose nodes are properly *labeled* via meaningful sentences. Each sentence must capture the "theme" of the snippets contained into the cluster it labels. This way the user is provided with a small, but intelligible, picture of the query answers at various levels of details.

Despite this commercial interest, we found just four scientific papers on this topic [22, 23, 9, 19]. None of them achieved results comparable to Vivisimo, that actually represents the state-of-the-art. In the present paper we address this problem in its full generality: labels of variable length for denoting the clusters, labels drawn from the Web snippets as non contiguous sequences of terms, clusters possibly overlapping and organized within a hierarchy. We achieve this results by means of an algorithmic approach that exploits some innovative ideas, at least from the academic side!

1 Introduction

The goal of a search engine is to retrieve from a given collection the documents which are "relevant" for a user query. A user query is nowadays modeled as a set of keywords extracted from a large dictionary of words; a document is typically a Web page, a `pdf` or postscript or `doc` file, or whatever file can be parsed into a set of tokens. The key difficulty in solving the retrieval problem above relies in the characterization of the adjective *"relevant"*. Actually this task is much difficult in that the *relevance* of a document is a subjective and time-varying concept. In fact, the same set of keywords may abstract different user needs that may also vary over the time according to the context in which the user is formulating his/her own query. As an example, for the query "search engine" a researcher may be interested in finding scientific papers, whereas a student may be interested into easy-to-read descriptions; nonetheless, the same researcher might be interested in "popular works" in the case that he has to prepare a presentation for a high-school.

Actually this is not the only difficulty in which the design of modern search engines incur. The explosion of the digital information available over the Web is making, from the one hand, impossible to index the whole set of existing Web pages and, from the other hand, to restrict the number of relevant documents to be returned to the user, since this number grows proportionally with the Web size. It turns out that the first difficulty can be partially alleviated by the use of *meta-search engines* [13] which exploit a pool of individual search engines to collect a larger set of relevant answers for a given user query [2]. However the use of *multiple sources*, each one exploiting its own ranking policy, makes the retrieval of the relevant documents even harder to deal with because of the need to *merge* multiple ranked lists into one unique ranked list.

This scenario is further complicated by the *poor patience* of the users that commonly browse through the first ten results (i.e. one screen) pretending to find there the "right" document for their query.

Personalization (e.g. Eurekster)[1], collaborative filtering (e.g. Amazon)), query-terms suggestion (e.g. AskJeeves), user behavior profiling (e.g. Epinions) are just some of the tools provided by modern search engines, now at the third generation, to ultimately match the "user needs" as best as possible. Among these tools a special contribution has been recently provided, in our opinion, with the *"Web snippets*[2] clustering and meaningful sentence extraction" tool. This novel IR-tool was introduced in a primitive form by Northernlight, and then it has been improved in its structure and efficacy by some commercial softwares like Vivisimo, Copernic and Groxis. The basic idea is that the *snippets* returned by the search engine, as the result of a user query, are grouped into *clusters* which are then organized in a *hierarchy* whose nodes are properly *labeled* via meaningful sentences. Each sentence must capture the "theme" of the snippets contained into the cluster it labels. This way the user is provided with a *small but intelligible picture, with various levels of details* (i.e. hierarchy), of the search-engine answers to his/her query. The user can therefore narrow the answer set by *navigating* through the hierarchy guided by the meaningful sentences that label its nodes. In some sense, this tool is a sort of *precision booster* in that it allows the search engine to be not limited to the first-10 results but it may exploit more results and defer to the user the "easier" task of navigating among them. From another point of view, we can look at this clustering engine as a tool that circumvents the synonimy/polisemy problem and alleviates the need for the user to be *extremely precise* in the query formulation. Actually this is a difficult task for the dummy users, which is also exposed to the missing of useful hits.

The commercial scenario. In the last few months there has been a surge

[1] The home pages of all the IR-tools, cited in the paper, are listed after the bibliography.

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

of commercial interest onto this novel IR-tool [5]. NorthernLight has patented proprietary technology that classifies every document in the database by subject, type, language and source. They developed a complete 17,000-node subject taxonomy (actually a human intensive work) until they stopped working on 2002. After NorthernLight many other clustering tools entered the Web context with various aims: some of them operate within an individual search engine to improve its precision or support the user in the query formulation (e.g. AllTheWeb, WiseNut, Teoma, Altavista, The Internet Archive); some others addressed novel application domains (GoogleNews and Microsoft NewsBot, GoogleSet and Overture Cluster Graph); yet others have been proposed as meta-search engines to enlarge the coverage of the Web search using some forms of clustering for query refinement (e.g. iBoogie, Kartoo, Groxis, Vivisimo, QueryServer, Mooter).

Among all of these proposals Vivisimo deserves a special attention because it is the meta-search engine achieving the best clustering and labeling results to date. For this reason it received the *best meta-search engine award* of the years 2002 and 2003 by SearchEngineWatch. The only information available about its underlying technology is that it employees statistical, linguistic and subject matter knowledge, as well stemming and stopwords. But, no details are provided on these techniques !

The scientific scenario. Despite of this great variety of commercial solutions for Web snippets clustering, we found just four scientific papers on this topic [22, 23, 19, 9]. Grouper [22] is the first academic meta-search engine which exploits clustering algorithms to group Web snippets (see also [19]). It uses phrases of variable lengths as features, instead of the classical single term representation, to label the clusters. These phrases are drawn as a *contiguous sequence of words* from the Web snippets, and they are identified by means of a Suffix Tree data structure. Clusters are organized in a hierarchy according to the Web snippets they contain. SHOC [23] extracts phrases to denote the clusters by using a lightweight approach based on Suffix Arrays. Clusters are organized in a hierarchy via an SVD approach applied on the single words composing the labeling phrases. FIHC [9] uses instead an analysis based on the Frequent Itemsets Problem in order to construct the clusters and to organize them in a hierarchy. In this case, items are single terms extracted from controlled document collections.

None of these papers, however, achieved results comparable to the ones provided by Vivisimo. Moreover, none has devised an open-source architecture with commodity components that allows researchers to start a productive investigation of such a promising research field. This scenario causes the cluster engine technology to remain largely a black art.

The algorithmic problem. It goes without saying that the clustering step might benefit from the huge scientific literature nowadays available for this topic (see e.g.[4, 10, 8, 6]), but here the hierarchical clustering formation is made complicated by three demanding requirements: (i) for obvious computational reasons, the engine must cluster the Web snippets returned by the queried search engine, and clearly, these snippets are much less informative than their corresponding entire documents because they consist of about 100 words; (ii) every cluster must have assigned a meaningful sentence, possibly formed by more than one word, in order to capture in an intelligible way the "category" of the documents contained in that cluster; (iii) clusters must be organized in a hierarchy and its nodes must be assigned with intelligible sentences in order to allow the post-navigation of the user for query refinement. All these issues make this problem not just *yet another clustering problem*, but a challenging variant both from an algorithmic and an IR point of view.

Our contribution. This scenario has motivated us to choose an intentionally pompous title for our paper that mimics the title of the paper by Brin and Page on Google's Anatomy [3]. Our choice was driven by the persuasion that the commercial vs. scientific scenario for the "clustering of Web snippets" mimics nowadays closely the search-engine scenario available in the 1998. In the present paper, we address the "Web snippets clustering and meaningful sentence extraction" problem in its full generality by proposing a novel algorithmic approach based on three key ideas:

- For the *sentence formation process*, we start from the "future directions of research" of the Grouper tool [22] and design an algorithm that constructs labels as *approximation of sentences appearing in the Web snippets*. Namely, a label does not need to be an exact sentence of a snippet, but it may miss some terms provided that they are not too many. This overcomes the "contiguity limitation" of Grouper's labels but it also introduces some demanding computational requirements that we address by devising proper data structures and algorithms.

- For the *sentences selection and cluster-assignment process*, we use some ranking functions which exploit a pool of *two knowledge bases* (KB) properly built by our engine at preprocessing time. One KB offers some *link and anchor-text information* derived by a small crawling of the Web done via the *Nutch open-source spider*. The other KB offers a narrow sampling of Web sites with their *high-quality humanly-made* descriptions drawn from the *Open Directory Project*.

- For the *hierarchical organization and labeling of the clusters* we exploit a novel algorithmic approach driven by the approximate sentences created above. We logically assign to each cluster *a set of (approximate) sentences* that describe its content at various levels of details. All of these sentences participate to the labeling of the ancestors of that cluster in the hierarchy. As a result, the labels lying over any hierarchy's path are *not necessarily one the substring of the other*, and thus they result much more descriptive.

The resulting software is built upon commodity components and should help the scientific community to investigate this challenging search-engine tool.

2 The software architecture

Our software provides the user with a Web interface, available for beta-tester at `http://roquefort.di.unipi.it/gulli/cluster/` The overall picture is provided in Figure 7.

2.1 The two knowledge bases

We remark that the following two KBs are built at preprocessing time and then used for the following ranking and sentence extraction processes, which occur at query time.

2.1.1 The Anchor Text and Link Database

Several search engines exploit the hyperlinks among Web pages as a source of information for ranking and retrieval. In this report, we use this information for the *clustering 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 surrounding each hyperlink that points to p. 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 $A(p) \cup B(p)$.

The motivation which underlies the use of $B(p)$ is that anchor texts often provide a precise, yet concise, description of the Web page p. In the search-engine literature, the use of anchor texts is well established [11]. In fact, the query "Madonna" submitted to Google successfully returns as the first hit the home page `http://www.madonna.com`, even if this page does not contain any textual information but just a shockwave presentation. [3] Unfortunately, deriving $B(p)$ from common search engines seems impossible because these terms neither occur into the returned snippets nor can be retrieved by formulating a proper query to the search engine itself. Nevertheless, we wished to exploit the meaningful information offered by $B(p)$, and thus we deployed the *Nutch's open source spider* to collect more than 50 millions Web pages selected among the top-cited ones.[4] After the gathering phase, contexts referring to each page p are grouped together and indexed via a B-tree called $\mathcal{A}_{\texttt{link}}$. We use the URL of p as the key for later retrieving. $\mathcal{A}_{\texttt{link}}$ is accessed trough a memory cache and takes about 4,5Gb of disk space. In Figure 1 we show some illustrative examples of the context data referring to well-known URLs.

2.1.2 The Semantic Knowledge Base

One of the key ingredients to effectively extract meaningful sentences for labeling our clusters is a method that *ranks* a set of candidate sentences.

[3] Google claims that "These terms only appear in links pointing to this page: Madonna".

[4] We are currently integrating a new Link Database built upon 200 millions of Web pages and about 400 millions of hyperlinks, given to us by the Nutch organization.

www.sigir2003.org	
SIGIR 2003 Home Page	4
SIGIR 2003	3
SIGIR	2
26th Annual International ACM SIGIR Conference	2
26th Annual International	1
Twenty-Sixth Annual International ACM SIGIR Conference Research and Development	1
SIGIR 2003 Conference Information	1
www.celera.com	
Celera	21
Celera Genomics	21
www.celera.com	14
Celera Genomics Applied Biosystems	1
CeleraScience	1
Celera Genomics group	1
Celera Corporation	1
Celera Genomics	1
Celera Genome Research	1

Figure 1: Two examples of queries on the Link Database. For each URL we show the text surrounding the hyperlinks that point to that URL, and the number of hyperlinks having that context.

There are many proposals in the IR literature to rank individual terms, either based on precomputed archives (e.g. ontologies, dictionaries and lexical databases, as the ones provided by WordNet and AppliedSemantics) or on frequency information [20]. In the former approach terms are weighted and related to each other by means of an intensive human effort. Hence the associated ranking is of high quality but does not scale and is mainly static. On the opposite, the frequency information is automatically computed on the top of an already collected index. This provides the so called `TFxIDF` ranking usually adopted in the modern (news [12]) search engines. Its main drawback however is that frequencies are computed over uncontrolled data which contains several non meaningful terms.

In order to not renounce to the positive features of both rankings, we designed our clustering engine based on a *mixed* strategy. We indexed the *Open Directory Project (DMOZ)*, a directory freely available on the Web, and calculated an ad-hoc `TFxIDF` measure for its controlled, and of high-quality, lexicon. Actually DMOZ classifies more than 3,500,000 sites which are organized in more than 460,000 categories. The DMOZ project claims to *"provide the means for the Internet to organize itself. As the Internet grows, so do the number of net-citizens. These citizens can each organize a small portion of the Web and present it back to the rest of the population, culling out the bad and useless and keeping only the best content"*. It is therefore not surprising that many Web search engines, like Google, use DMOZ both for ranking and retrieval.

We are the first, to the best of our knowledge, to use the whole DMOZ for Web snippets clustering. Specifically, we indexed this valuable archive by developing a *ranking engine*, hereafter denoted by $\mathcal{R}_{\texttt{dmoz}}$, based on inverted lists. The lexicon of $\mathcal{R}_{\texttt{dmoz}}$ consists of more than three millions single terms. Upon $\mathcal{R}_{\texttt{dmoz}}$ we implemented a `TFxIDF` measure, taking into account single terms, pairs of terms, and longer sentences. Unlike the standard `TFxIDF` measure, that is document centered, here we adopt a *category* centered formulation.

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 C_i that takes in account its depth in the DMOZ hierarchy (we are postulating an increased importance for deeper categories), and let $b(t, C_i)$ 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 (we are postulating an increased importance of terms in crucial parts of the category's description). We define $\text{TF}(t) = 1 + \log \#(t)$, $\text{IDF}(t) = \log \frac{\#_C}{\#_C(t)}$ and compute the rank of t, with respect to a category C_i, as:

$$\text{KB}(t, C_i) = b(t, C_i) * \text{TF}(t) * \text{IDF}(t) * ns(C_i)$$

The rank for a sentence $s_{h,k}$ formed by two terms (t_h, t_k) is then defined as:

$$\text{KB}(s_{h,k}) = \max_{C_i}\{ \prod_{r=h,k} b(t_r, C_i) * \text{TF}(t_r) * \text{IDF}(t_r) * ns(C_i)\}$$

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

$$\text{KB}(s) = \sum_{s_{h,k} \in P(s)} \text{KB}(s_{h,k}).$$

acm invitation to the 2003 conference
  acm invitation = 46
  invitation conference = 140
  = 186; [0.16, 0.23]

acm journal of experimental algorithmics
  acm journal = 412
  journal experimental = 268
  experimental algorithmics = 34
  = 714; [0.08, 0.23]

Figure 2: Two examples of sentences ranking by $\mathcal{R}_{\texttt{dmoz}}$. We also report the [average time, actual time] for ranking in seconds.

In order to speed up the on-line computation of our clustering engine, we used $\mathcal{R}_{\texttt{dmoz}}$ to compute off-line the `TFxIDF` ranking of ten millions of term pairs lying within a proximity of four adjacent terms in DMOZ (skipping the stopwords). Then we stored these rankings in $\mathcal{P}_{\texttt{dmoz}}$, requiring about 500Mb of disk space.

2.2 The first module: The Snippet Analyzer

The task of the first module is to forward the user query to a set of selected, remote search engines and to collect and merge the returned snippets. These results are eventually enriched with information retrieved from the local $\mathcal{A}_{\texttt{link}}$ index , and analyzed from a syntactic and linguistic point of view. The collected information form the so called *Enriched Virtual Document Collection* that constitutes the input for the next *Sentences Generator* module.

2.2.1 The Retrievers

The pool of queried search engines consists of Lycos, Altavista, Yahoo, Teoma and AllTheWeb, but other search engines could be easily added. In this pool we have also Google which is queried through its Web Service API. All of these search engines, queried by the *Retrievers*, return a list of Web pages (and their snippets) properly ranked according to their own ranking policy. For each listed Web page p, the Retrievers query also the local database $\mathcal{A}_{\texttt{link}}$ to fetch the *anchor texts* surrounding the external hyperlinks that point to p.

Currently the Retrievers request an average of 40 Web snippets to each search engine and we are retrieving a total 200 results for each user query. All of these parameters, and many others, are configurable by the user via our Web interface.

2.2.2 The Merger

All the retrieved snippets and anchor texts are analyzed by the *Merger* whose aim is to build a Virtual Document for each Web page p. The analysis consists of three phases. It first builds for p a *temporary document* consisting of all the snippets and anchor texts referring to p. Each temporary document is then augmented by other syntactic information by segmenting it in different sections that identify its title, description, text anchors, categories and category's titles. Then each section is further segmented into sentences by choosing as separators the punctuation symbols and HTML tags. The result is a Collection of *Virtual Documents* (*VDC*).

2.2.3 The Linguistic Analyzer

The VDC is then processed by several linguistic tools which filter a *stop list* of about 2000 words, belonging to 12 different languages; stem the result by using a variant of the Snowball stemmer[15]; and finally, extract Part of Speeches (PoS) and Named Entities [18, 14]. The output is the so called *Enriched Virtual Document Collection* (EVDC) that constitutes the input for the next *Sentence Generator* module.

U: http://www.acm.org/sigir/
T: information server . . . sigir information server
CT: SIGIR: Information Retrieval
C: Addresses issues ranging from theory to user demands in the application of computers to the acquisition, . . .
D: . . . sigir-announce the official electronic mailing list of sigir. sig-irlist an electronic newsletter on ir-related topics. acm sigir web site
L: http://www.acm.org/sigir/ . ACM SIGIR . ACM SIGIR Home Page . ACM Special Interest Group on Information Retrieval (SIGIR) . ACM SIGIR Forum (Special Interest Group on Information Retrieval) . ACM . ACM Interest Group on Information Retrieval . ACM:SIGIR . Special . ACM SIGIR . Special Interest Group on Information Retrieval . ACM SIGIR Information . Special Interest . ACM's SIGIR

Figure 3: An example of EVD for the snippets related to SIGIR2003. We use the notation: U = url, T = title, CT = Category Title, C = Category, D = description, L = Context's Link. A single "dot" denotes different contexts or segmented sentences; whereas multiple "dots" denote something omitted for conciseness.

2.3 The second module: The Sentences Generator

The task of this module is to create a set of properly labeled *clusters* starting from the EVDC. These clusters are yet "flat" without any hierarchical organization. It will be the last module of our engine that will structure them into a hierarchy.

The classical approach for clustering documents is to carefully extract single keywords from their content and to group together documents which show the maximal homogeneity within the groups and the maximal heterogeneity among different groups. It has been shown [17, 16] that in many situations the use of single terms as keywords (such as "white" or "house") provides enough power to index documents with acceptable precision, and that the use of multiple terms such as ("white house") could increase the precision but at the cost of a slight reduction in recall. Some recent results have introduced a different point of view in the context of the Clustering of Web Snippets [22, 23]. Here, the main goal is to provide a high precision to help the user in identifying relevant documents among the huge amount of answers returned by current search engines. Moreover, for an human being is much simpler to identify a cluster if it is labeled with an intelligible sentence instead of a single term. The Grouper engine [22] achieved these conclusions: *"The most distinctive aspect of the algorithm was its treatment of documents as strings of words, represented by a suffix tree, in contrast with the standard vector-based representation"*.

The decision to label the clusters by using long sentences is powerful but it introduces some difficulties. From a syntactical point of view, the sentences "John Fitzegerald Kennedy", "Kennedy John" and "John F. Kennedy" are different from each other, but it should be desirable to consider them as the same sentence. Grouper and the other academic tools [22, 23, 19, 9] did not deal with this problem because they used indexing data structures like Suffix

Trees and Suffix Arrays that allow the efficient treatment of sentences *formed by contiguous terms*. Nevertheless it was clear in the scientific literature, see [21], that the extraction of "approximate" sentences (i.e. involving not contiguous terms) might boost the precision of the cluster labels and, in turn, the usefulness of the cluster hierarchy to humans.

This has been our starting point for the design of the Sentence Generator module. To the best of our knowledge, we are the first to design a clustering engine that tries to cluster together Web snippets containing *almost the same sentences*. For efficiency reasons, our approach is syntactic and works in four phases detailed below.

2.3.1 2-Approximate Sentences Generator

Given the set of documents which belong to the EVDC, we take into consideration each sentence s identified by the Snippet Analyzer (see Section 2.2.2). For each s, we take all the pairs of terms that occur in s within a proximity d. Stop words are skipped. Some of these term pairs are dropped if they have been already generated within the same sentence or if their second term is a suffix/prefix of the first one. This set of 2-approximate sentences is hereafter denoted by $\texttt{A2S}(s,d)$.

We point out that, due to the approximation introduced by the proximity, some of the term pairs in $\texttt{A2S}(s,d)$ could introduce artificial meaning not present in the EVDC. For instance, consider the sentence "Alexei Abrikosov, Vitaly Ginzburg and Anthony Legget received the nobel price ...". Among the others, we derive the 2-approximate sentence "Vitaly Legget" which is an artificial name not present in the sentence above.

To circumvent this problem, we apply a simple rule to the set $\texttt{A2S}(s,d)$. Let $s_{h,k} = (t_h, t_k)$ be a 2-approximate sentence generated from some virtual document VD_i. We say that $s_{h,k}$ is *valid* if either the two terms t_h and t_k appear contiguous in some other VD_j or they appear contiguous in $\mathcal{P}_{\texttt{dmoz}}$. Sentences not marked as valid are discarded. This syntactic approach allows to overcome the "contiguity constraint" of the algorithms based on Suffix Trees and Suffix Arrays, and also allows to avoid the introduction of any fake semantic meaning.

2.3.2 2-Approximate Sentences Ranker

The task of this submodule is to rank each valid 2-approximate sentence and to suggest the correct order of its two terms. Our ranking has two peculiarities: it exploits "local" information and "background" information. Local information is a set of statistics collected from the Web snippets themselves about a 2-approximate sentence. Background information is a judgement on the importance and quality of a 2-approximate sentence assigned according to the database $\mathcal{P}_{\texttt{dmoz}}$, computed at preprocessing time. Let $s_{h,k} \in \texttt{A2S}(s,d)$, let $\#s_{h,k}$ be the number of times that $s_{h,k}$ occurs in the EVDC, and let $ws(s_{h,k})$ be a boosting factor that assigns more weight to those sentences occurring in the Web snippets returned in the top positions of a search engine. We compute the rank of $s_{h,k}$ as a function of $\#s_{h,k}$, $ws(s_{h,k})$ and $KB(s_{h,k})$. The 2-approximate sentences having a rank below a given threshold are discarded. For the remaining sentences, this submodule queries $\mathcal{P}_{\mathtt{dmoz}}$ to infer the most common order of their terms, and then reverse the term pair which has the lower rank.

2.3.3 Indexer for the 2-Approximate Sentences

This submodule builds *on-the-fly* an indexing data structure on the well-ranked 2-approximate sentences. We use neither suffix arrays nor suffix trees but just a combination of bitmaps and inverted lists. They are much efficient to store and allow the fast construction of longer and intelligible k-approximate sentences from the available 2-approximate sentences.

Bitmaps are used to know whether a 2-approximate sentence occurs in a virtual document of the EVDC. Inverted lists are used to know more detailed information on a 2-approximate sentence such as the `snipID`, the subsection (e.g. title, description, link, category, category title), the exact sentence number where $s_{h,k}$ occurs, and the exact positions where the terms t_h and t_k occur within $s_{h,k}$. These inverted lists are sorted to exploit compression and perform some operations faster.

We point out that by "inverting" the Inverted Lists we derive the 2-approximate sentences that are *candidate* labels for every snippet.

2.3.4 Longer Sentences and Clusters Creation

This submodule constructs longer k-approximate sentences from the 2-approximate ones, using the ranking information and the indexing data structures computed by the previous submodules.

The set $\mathtt{A2S}(s,d)$ is ordered by rank and the m topmost sentences are considered. They form the set $\mathtt{A2S}_m(s,d)$. To create the longer 4-approximate sentences we pick from $\mathtt{A2S}_m(s,d)$ a pair of sentences $s_{h,k}$ and $s_{x,y}$ and merge them if they have the same snipID, the same subsection, the same sentence, and their terms occur within a fixed proximity window. Actually, this merging requires just a linear scan of the two sorted inverted lists associated with $s_{h,k}$ and $s_{x,y}$.

The process can be generalized to obtain a k-approximate sentence occurring in s, for any given k. Here some *proximity constraints* that make the sentence valid are taken into account. Various parameters can be tuned to modify these proximity constraints and create longer sentences. The cost of this process is negligible with respect to the other computations, because it requires a linear scan of the available inverted lists. We also point out that our approach has something in common with the Frequent Itemset Problem [1] in that we consider a k-approximate sentence interesting (i.e. well-ranked), if all of its constituting 2-approximate sentences are interesting.

Given the k-approximate sentences, we postulate that *"virtual documents sharing the same k-approximate sentences belong to the same cluster"*. This is the key policy we adopt to cluster the snippets and therefore to form the leaves of our hierarchy. We point out that, according to this policy, a snippet may occur in many clusters. Thus clusters may be overlapping. This is consistent with the observation that a Web page can cover *multiple themes*. The result of this clustering step is a flat pool of clusters, each one having assigned a k-approximate sentence as meaningful denoting label.

2.4 The third module: The Hierarchical Organizer

From the algorithmic point of view, this is the tricky part of the software which we just sketch here, for the sake of brevity.

We know that each cluster, produced by the second module above, has been assigned with a k-approximate sentence, that we hereafter call the *primary label* of that cluster. This is indeed a sentence that occurs within all the (virtual) documents forming that cluster and it has the largest rank. Primary labels are of *high quality* but are often *too much specific* to generate the labels of the nodes in the higher levels of the hierarchy. Hence our first key choice has been to associate to each cluster also a *set of secondary labels* defined as follows: they are k-approximate sentences which have a *good rank* and occur in *at least* the $c\%$ of the documents contained in the cluster. Both primary and secondary labels give a description of the cluster at various levels of details. So they are useful for the formation of the sentences labeling the nodes at the various levels of the hierarchy.

In order to manage primary and secondary labels efficiently, we mapped them to a single string which is formed by concatenating all of these labels, separated by a special symbol \$. This string is called *the signature* of the cluster.

The third module forms the hierarchy by merging the clusters into superclusters, driven by their primary and secondary labels. It consists of three phases, executed repeatedly until a few number of superclusters remain: *abstraction*, *pruning* and *fusion*.

The abstraction phase aims at building another level of the hierarchy starting from a set of (super)clusters $\mathcal{SC}_1, \ldots \mathcal{SC}_z$, each having its own signature. The algorithm compares the signatures of all $\mathcal{SC}_i$s and extracts the common substrings which do not include the special symbol \$. These substrings provide *candidates* for the new superclusters and their labels.

The next merging phase actually forms and labels these superclusters. If $\mathcal{SC}_{i_1}, \ldots, \mathcal{SC}_{i_z}$ have signatures containing the same substring s, then we form a super-cluster $\mathcal{SC}'$ whose primary label is s and whose documents are given by the union of the documents of $\mathcal{SC}_{i_1}, \ldots, \mathcal{SC}_{i_z}$. We can look at $\mathcal{SC}'$ as the *father* of $\mathcal{SC}_{i_1}, \ldots, \mathcal{SC}_{i_z}$ in the hierarchy. The rank of s is computed as a function of the rank of the (primary or secondary) labels having s as a substring. We just note here that by exploiting the properties of the signatures we do not need to execute an all-against-all comparison among these signatures, but we may adopt a computationally lighter approach. Moreover we point out that, since the signatures are approximate sentences drawn from the virtual documents, the above substrings do not necessarily occur contiguously into the original Web page.

The final pruning phase aims at simplifying the hierarchy obtained so far, by removing the superclusters that appear to be *redundant*. The number of these superclusters is not negligible and their deletion contributes to make the overall hierarchy more intelligible. We formalize this problem as a *covering* problem and solve it via a greedy approach. More precisely, we try to select a set of superclusters that cover *most of the superclusters in the lower level*, are labeled by *long and well-ranked sentences*, and are quite heterogeneous. The unselected superclusters are canceled; the documents of the uncovered clusters (of the lower level) are put into an Other Topic supercluster. The remaining superclusters provide the next level of the hierarchy. For each remaining supercluster SC', we compute its secondary label by uniting the secondary labels of $SC_{i_1}, \ldots,$ SC_{i_z} and by discarding then those labels which occur in less than $c\%$ of the documents of SC'.

It may happen that the primary labels of the SC_{i_j}s share many terms. This is good from a clustering point of view, but it is redundant from a user point of view because (s)he sees many similar labels below the same father thus jeopardizing the reading of the hierarchy. We thus introduce some heuristics, based on entropy maximization of the distribution of the terms in the labels, that try to prevent this phenomenon and drive us to drop some clusters or merge some other similar ones.[5]

At this point we have re-established the inductive conditions on the superclusters at the current level, and thus we can repeat the above three phases.

3 Some experimental Results

We performed our experiments on a commodity PC running Linux with a CPU P4 and RAM 1.5Gb. The software has been written in Perl, and is compiled into a standard Apache Web server for efficiency. In Figure 5 we show the results for some queries whose topic ranges from Computer Science to Finance, from Politics to Travels. For each query we show the hierarchy and the labels produced by our clustering engine. The hierarchy is stopped to two levels, the (super)clusters are ordered by decreasing rank, dropping the ones whose rank is below some threshold. In the Figure we also draw arrows close to labels that we judge interesting.

We point out that our labels are approximate sentences occurring in the Web snippets and have variable length; this is crucial for their readability. Furthermore, the label assigned to a cluster is *not a substring* of the label assigned to its ancestors; nonetheless we note surprising "semantic" relations among them. The picture offered by the hierarchical labels recalls a sort of knowledge extraction process, which is useful to capture the "intention behind the user query".

We do not comment further because we worry about a "horoscope effect", so we leave to the reader the task of judging the quality of the results shown in Figure 5. In fact, although do exist several criteria for evaluating clustering

[5]As suggested by Raul Valdez-Perez, President of Vivisimo, important factors for cluster descriptions are label conciseness, accuracy and distinctiveness.

results (see e.g. [11]), to the best of our knowledge, none of them seems to have taken into account the expressiveness of the cluster labels. This appears to be an interesting research problem. It goes without saying that our software needs further improvements as shown in Table 4.

Software Modules	*secs*
Retrievers	20
Linguistic Analysis	2
Sentences Generator	
2-approximate sentences	2
k-approximate sentences	2
Hierarchical Organizer	13

Figure 4: The time is averaged over various queries, but it shows large variance whose causes needs to be further investigated.

We plan to optimize the Retrievers by using a multi-thread approach, and to engineer the Hierarchical Organizer (currently it takes quadratic time complexity) by means of proper indexing data structures for substring matching that should allow to make the computation in overall linear time [7].

In the light of the experimental results, the proposed software and algorithmic approach appear much promising so that many issues remain to be investigated:

- We are starting the engineering of the various modules that constitute our Clustering Engine, and aim at porting them in C++.

- Our experiments indicate that skipping the stopwords is necessary during the clustering and ranking process. Nevertheless, users like to have them in the labels associated to the clusters (for instance "searching using a suffix tree" is more meaningful than "searching suffix tree"). We plan to (re)introduce the stopped terms in the k-approximate sentences.

- We plan to introduce some kind of semantical analysis which should work together with our syntactical, statistical and linguistic approach for building up more intelligible sentences and for simplifying the hierarchy.

Finally, we aim at applying our clustering engine to contexts which are different from Web Snippets analysis. One application is the suggestion of "Related Search" terms, as used by many engines such as Altavista, Teoma, AllTheWeb.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

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

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[4] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *WWW6*, 1997.

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

[6] D. Fasulo. An analysis of recent work on clustering algorithms. Technical Report 01-03-02, Department of Computer Science and Engineering, University of Washington, 1999.

[7] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *IEEE Symposium on Foundations of Computer Science (FOCS '00)*, pages 390–398, 2000.

[8] W. B. Frakes and R. Baeza-Yates. *Clustering Algorithms*, chapter 16. Prentice Hall, Englewood Clifs, NJ, 1992.

[9] B. Fung, K. Wang, and M. Ester. Large hierarchical document clustering using frequent itemsets. *SIAM International Conference on Data Mining*, 2003.

[10] Johannes Grabmeier and Andreas Rudolph. Techniques of cluster algorithms in data mining. In *Data Mining and Knowledge Discovery*, volume 6(4), pages 303–360, 2002.

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

[12] Monika Henzinger, Bay-Wei Chang, Brian Milch, and Sergey Brin. Query-free news search. In *WWW12*, pages 1–10, 2003.

[13] Adele E. Howe and Daniel Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.

[14] Lingua Perl Modules. http://www.cpan.org.

[15] SnowBall. A language in which stemming algorithms can be easily represented. http://snowball.tartarus.org.

[16] Tomek Strzalkowski, Gees C. Stein, G. Bowden Wise, Jose Perez Carballo, Pasi Tapanainen, Timo Jarvinen, Atro Voutilainen, and Jussi Karlgren. Natural language information retrieval: TREC-7 report. In *Text REtrieval Conference*, pages 164–173, 1998.

[17] Xiang Tong, Chengxiang Zhai, Natasa Milic-Frayling, and David A. Evans:. Evaluation of syntactic phrase indexing. In *Text REtrieval Conference*, 1997.

[18] TreeTagger. A language independent part-of-speech tagger. http://www.ims.uni-stuttgart.de/projekte/corplex/.

[19] D. Weiss and J. Stefanowski. Web search results clustering in polish: Experimental evaluation of carrot. In *New Trends in Intelligent Information Processing and Web Mining Conference*, 2003.

[20] I.A. Witten, A.Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kauffam, 1999.

[21] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[22] Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks*, 31(11–16):1361–1374, 1999.

[23] Dell Zhang and Yisheng Dong. Semantic, hierarchical, online clustering of web search results. In *ACM 3rd Workshop on Web Information and Data Management(WIDM'01)*, 2001.

Web Sites cited in the Report

http://labs.Google.com/sets
http://www.AllTheWeb.com
http://www.Altavista.com
http://www.Amazon.com
http://www.AppliedSemantics.com
http://www.Archive.org
http://www.AskJeeves.com
http://www.cogsci.princeton.edu/∼wn/ (`WordNet`)
http://www.Copernic.com
http://www.Epinions.com
http://www.Eurekster.com
http://www.Google.com
http://www.Groxis.com
http://www.Kartoo.com
http://www.iBoogie.tv
http://www.Lycos.com
http://www.Mooter.com
http://www.NorthernLight.com
http://www.Nutch.org
http://www.Overture.com
http://www.QueryServer.com
http://www.SearchEngineWatch.com
http://www.Teoma.com
http://www.Vivisimo.com
http://www.Yahoo.com
http://www.WiseNut.com

Query: “sony”	
music sony	
	music entertainment
	artists ensembles music ⟸
	sony nashville
	music copyright
	launch music player
mobile multimedia	
	ericsson mobile communications ⟸
	sony mobile
	mobile multimedia consumer
	multimedia products people
	multimedia consumer products
computer entertainment	
	sony computer
	playstation north america
	computer entertainment america
	official corporate site ⟸
	official site game
	official playstation gaming ⟸
	computer entertainment japan
	computer entertainment europe
site game information	
	playstation north america
	official site game
	official playstation gaming
	official site game information
...	

Query: “last minute”	
car rental	
	minute travel
	minute discounts
	cheap airline tickets
	car rental companies
	car rental deals
	negotiated prices ⟸
	travel discounts hotwireċom
	hotwire hot-fares
travel reservation discounts	
	travel reservation discounts line
	holiday offers discount
	offers discount travel ⟸
latest news	
	minute wales
	wales latest news
	latest news holidays
discount travel	
	travel reservation discounts
	travel reservation discounts line
	discount travel deals
	travel holiday deals
	deals airline tickets ⟸
	minute offers
	vacations greece cyprus ⟸
	minute holiday
...	

Query: “bush”	
site white house	
	official white house
	states white house
	press releases president george
	executive office president
	43rd president united ⟸
	president united states ⟸
	links executive branch
development family literacy	
	foundation family literacy
	barbara foundation ⟸
	bush foundation
	barbara bush
	mission development family literacy
site presidential campaign	
	site president george
	official campaign site
	official re-election site ⟸
	bush cheney
	re-election site president ⟸
	official presidential campaign
defense council	
	bush record
	council environment watch
	resources defense council
	natural resources defense
...	

Query: “dvd”	
dvd news	
	dvd reviews
	digital bits
	easter eggs ⟸
	dvd movie
	anime dvd
	dvd reviewer
	dvd faq
	channel news
reviews dvd	
	movie database
	dvd heimkino
	magazin vielen
	area dvd ⟸
	latest update jim taylor
online search	
	price search
	comparison engine online
	online search deep discount
	dvd planet
	price comparison engine ⟸
	dvd price ⟸
	prices online
open source	
	open source development
	resources open source
movie dvd	
	movie database
	movie reviews
	dvd store
japanese animation ⟸	
	anime dvd
	dvd animania
	animation news source
	resource japanese animation
	focuses japanese animation
...	

Query: “clustering”	
software clustering ⟸	
	society north america
	classification society north
	online software ⟸
	list cluster analysis
	multivariate analysis ⟸
	software package
cluster analysis	
	society north america
	classification society north
	fuzzy cluster analysis ⟸
	cluster analysis matlab ⟸
	resources cluster analysis
	multivariate analysis
	matlab code
	zipped ṁ files
search results ⟸	
	automatic categorization ⟸
	offers search results
	search results document
	categorizes search results
engine	
	document clustering
	data engine
	clustering engine
	meta search engine ⟸
technical note	
	src technical note
	digital systems research
	systems research center
	technical note 1997-015
red hat applications	
	family red hat
	red hat server ⟸
	red hat content
	red hat suite technologies
...	

Query: “equity”	
loan equity	
	loans loan rates
	mortgage loan rates ⟸
	exchange mortgage rates
online loan rates	
	rate debt consolidation
	credit cards loans
	loans equity
	loans online loan
	loans credit cards
	loans mortgage loan
exchange mortgage rates	
	loans loan rates
	mortgage loan rates
	lenders exchange mortgage rates
loans bad credit ⟸	
	credit equity
	refinancing loans
	mortgage rate calculator
	loans equity
	loans perfect credit
retail investor	
	retail investor services
	equity services
	equity investor
	poor retail investor
mortgage equity	
	line credit finish application
venture capital ⟸	
	european venture capital
	venture capital association
	canadian venture capital
	facilitate european venture
	equity atlantic
	equity partners
	acf equity
...	

Figure 5: For each query we show the hierarchy and the labels produced by our clustering engine. The hierarchy is stopped to two levels. The (super)clusters are ordered by decreasing rank, dropping the ones whose rank is below some threshold. Arrows are drawn close to labels that we judge interesting!

Query: "iraq"
radio free europe
government views
offered radio free
database latest
civilian body count
count database latest
body count database
database latest updates
oil food ⟸
iraq programme
office iraq
office programme
oil food programme
office program
united nations office
oil food program
country iraq
library congress federal
library map collection
division library congress
perry-castaeda map collection
middle east historical
federal research division
division country studies
information iraq
perry-castaeda map collection
department commerce
target iraq
task force ⟸
peace iraq
iraq center
analysis middle east ⟸
peace center
fellowship national network
education peace
map collection
iraq maps
library map collection
perry-castaeda map collection
middle east historical
map collection university
middle east historical
library map collection
perry-castaeda map collection
middle east historical maps
news iraq
bbc news conflict
bbc news indepth
iraq action
links bbc news
action coalition
resource information site
conflict iraq
news history culture
institute university ⟸
learn lost treasures ⟸
treasures iraq
oriental institute university
chicago oriental institute ⟸
institute university chicago
...

Query: "data mining"
data mining
conference knowledge discovery ⟸
international knowledge discovery
knowledge discovery nuggets ⟸
gregory piatetsky-shapiro knowledge discovery
siam international conference
data mine
discovery databases
knowledge discovery resources ⟸
knowledge discovery nuggets
gregory piatetsky-shapiro knowledge discovery
data conference
international knowledge discovery
siam international conference ⟸
ieee international conference ⟸
publication business intelligence
premier publication business
information conference site
ibm almaden research
ibm research
search ibm research
data project
resources
resources knowledge discovery
connecticut state university
mining resources
...

Figure 6: Other examples of queries

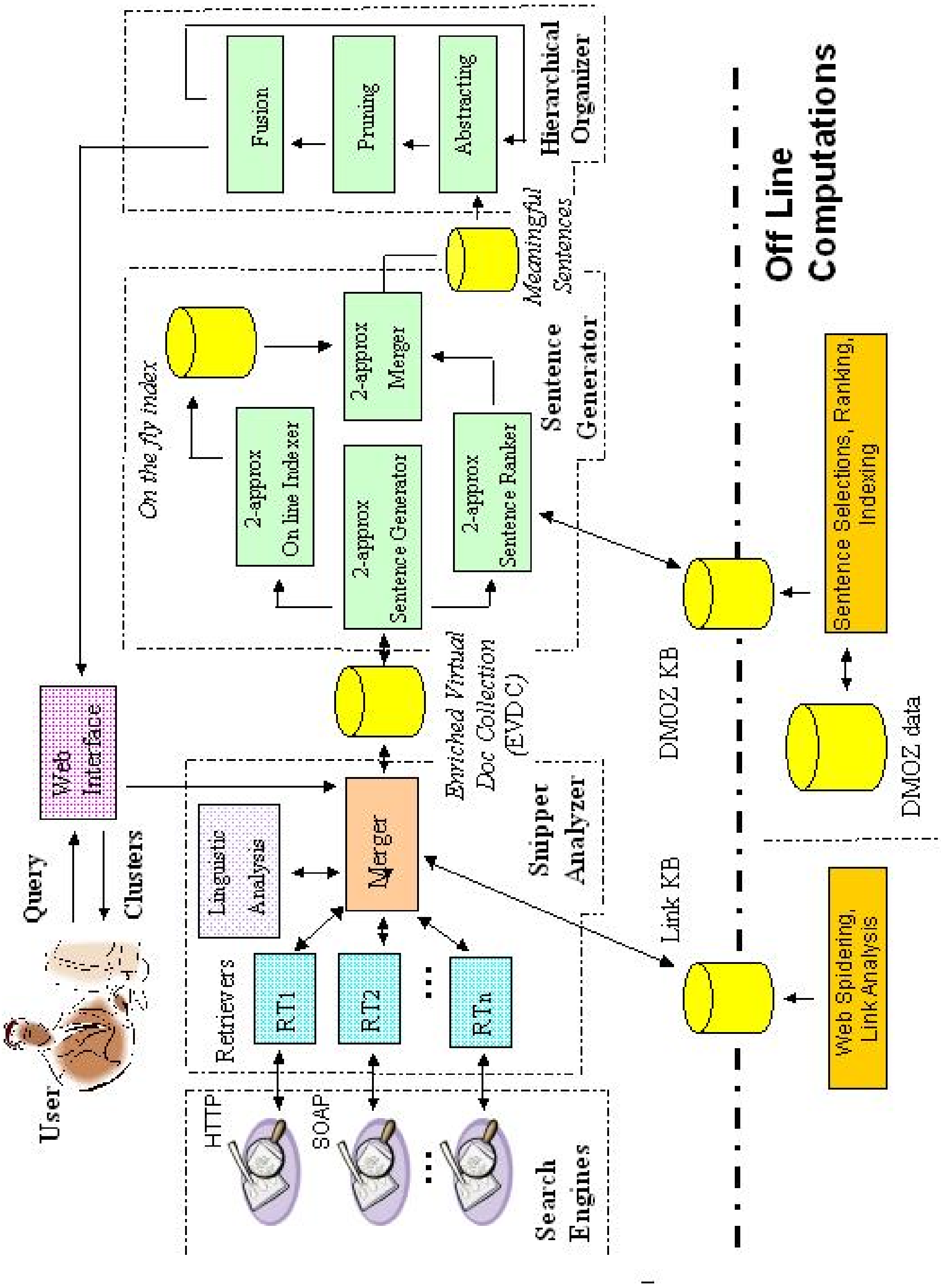

Figure 7: Architecture of the Clustering Engine. There are two types of computations: an off-line computation that builds two knowledge bases at preprocessing time; and an on-line computation that is executed at query time and groups the snippets into clusters, extracts meaningful sentences to label them and produces a hierarchy.