# Cluster Generation and Cluster Labelling for Web Snippets

Filippo Geraci[1,2], Marco Pellegrini[1], Marco Maggini[2], and Fabrizio Sebastiani[3]

[1] Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Via G Moruzzi 1, 56124 Pisa, Italy
{f.geraci, m.pellegrini}@iit.cnr.it
[2] Dipartimento di Ingegneria dell'Informazione, Università di Siena, Via Roma 56, 53100 Siena, Italy
maggini@ing.unisi.it
[3] Istituto di Scienza e Tecnologia dell'Informazione, Consiglio Nazionale delle Ricerche, Via G Moruzzi 1, 56124 Pisa, Italy
fabrizio.sebastiani@isti.cnr.it

**Abstract.** This paper describes Armil, a meta-search engine that groups into disjoint labelled clusters the Web snippets returned by auxiliary search engines. The cluster labels generated by Armil provide the user with a compact guide to assessing the relevance of each cluster to her information need. Striking the right balance between running time and cluster well-formedness was a key point in the design of our system. Both the clustering and the labelling tasks are performed on the fly by processing only the snippets provided by the auxiliary search engines, and use no external sources of knowledge. Clustering is performed by means of a fast version of the furthest-point-first algorithm for metric $k$-center clustering. Cluster labelling is achieved by combining intra-cluster and inter-cluster term extraction based on a variant of the information gain measure. We have tested the clustering effectiveness of Armil against Vivisimo, the *de facto* industrial standard in Web snippet clustering, using as benchmark a comprehensive set of snippets obtained from the Open Directory Project hierarchy. According to two widely accepted "external" metrics of clustering quality, Armil achieves better performance levels by 10%. We also report the results of a thorough user evaluation of both the clustering and the cluster labelling algorithms.

## 1 Introduction

An effective search interface is a fundamental component in a Web search engine. In particular, the quality of presentation of the search results often represents one of the main keys to the success of such systems. Most search engines present the results of a user query as a ranked list of Web snippets. *Meta-search engines* (MSEs) integrate the items obtained from multiple "auxiliary" search engines, with the purpose of increasing the coverage of the results. However, without an accurate design, MSEs might in principle even worsen the quality of the information access experience, since the user is typically confronted with an even larger set of results. Thus, key issues to be faced by MSEs concern the exploitation of effective algorithms for merging the ranked lists of results (while at the same time removing the duplicates), and the design of advanced user interfaces based on a structured organization of the results. This latter aspect is

F. Crestani, P. Ferragina, and M. Sanderson (Eds.): SPIRE 2006, LNCS 4209, pp. 25–36, 2006.

usually implemented by grouping the results into homogeneous groups by means of clustering or categorization algorithms.

This paper describes the Armil system[1], a meta-search engine that organizes the Web snippets retrieved from auxiliary search engines into disjoint clusters and automatically constructs a title label for each cluster by using only the text excerpts available in the snippets. Our design efforts were directed towards devising a fast clustering algorithm able to yield good-quality homogeneous groups, and a distillation technique for selecting appropriate and useful labels for the clusters. The speed of the two algorithms was a key issue in our design, since the system must organize the results on the fly, thus minimizing the latency between the issuing of the query and the presentation of the results. Second-level clustering is also performed at query time (i.e. not on demand) to minimize latency. In Armil, an equally important role is played by the clustering component and by the labelling component. Clustering is accomplished by means of an improved version of the furthest-point-first (FPF) algorithm for $k$-center clustering [1]. To the best of our knowledge this algorithm had never been used in the context of Web snippet clustering or text clustering. The generation of the cluster labels is instead accomplished by means of a combination of intra-cluster and inter-cluster term extraction, based on a modified version of the information gain measure. This approach tries to capture the most significant and discriminative words for each cluster.

One key design feature of Armil is that it relies only on the information returned by the auxiliary search engines, i.e. the snippets; this means that no external source of information, such as ontologies or lexical resources, is used. We thus demonstrate that such a lightweight approach, together with carefully crafted algorithms, is sufficient to provide a useful and successful clustering-plus-labelling service. Obviously, this assumption relies on the hypothesis that the quality of the results and of the snippets returned by the auxiliary search engines is satisfactory. We have tested the clustering effectiveness of Armil against Vivisimo, the *de facto* industrial standard in Web snippet clustering, using as benchmark a comprehensive set of snippets obtained from the Open Directory Project hierarchy. According to two metrics of clustering quality that are normalized variants of the Entropy and the Mutual Information [2], Armil achieves better performance levels by 10%. Note that, since the normalization reduces the ranges of these measures in the interval $[0, 1]$, an increase of 10% is noteworthy. We also report the results of a thorough user evaluation of both the clustering and the cluster labelling algorithms.

**Outline of the clustering algorithm.** Clustering and labelling are both essential operations for a Web snippet clustering system. However, each previously proposed such system strikes a different balance between the two aspects. Some systems (e.g. [3, 4]) view label extraction as the primary goal, and clustering is a by-product of the label extraction procedure. Other systems (e.g. [5, 6]) view instead the formation of clusters as the most important step, and the labelling phase is considered as strictly dependent on the clusters found. We have followed this latter approach. In order to cluster the snippets in the returned lists, we map them into a vector space endowed with a distance function, which we treat as a metric; then a modified furthest-point-first algorithm (M-FPF) is applied to generate the clusters. The M-FPF algorithm generates the same clusters of the "standard" FPF algorithm, but uses filters based on the triangular inequality to speed up the

[1] The Armil system can be freely accessed at `http://armil.iit.cnr.it/`.

computation. As such, M-FPF inherits a very important property of the FPF algorithm, i.e. it is within a factor 2 of the optimal solution for the $k$-center problem [7]. The second interesting property of M-FPF is that it does not compute centroids of clusters. Centroids tend to be dense vectors and, as such, their computation and/or update in high-dimensional space is a computational burden. M-FPF relies instead only on pairwise distance calculations between snippets, and as such better exploits the sparsity of the snippet vector representations.

**Outline of the cluster labelling algorithm.** The cluster labelling phase aims at extracting from the set of snippets assigned to each cluster a sequence of words highly descriptive of the corresponding group of items. The quality of the label depends on its well-formedness (i.e. whether the text is syntactically and semantically plausible), on its descriptive power (i.e. how well it describes what is contained in the cluster), and on its discriminative power (i.e. how well it differentiates what is contained in the cluster with respect to what is contained in other clusters). The possibility to extract good labels directly from the available snippets is strongly dependent on their quality and, obviously, on the homogeneity of the produced clusters. In order to pursue a good tradeoff between descriptive and discriminative power, we select candidate words for each cluster by means of $IG_m$, a modified version of the *Information Gain* measure [2]. For each cluster, $IG_m$ allows the selection of those words that are most representative of its contents and are least representative of the contents of the other clusters. Finally, in order to construct plausible labels, rather than simply using the list of the top-scoring words (i.e. the ones that maximize $IG_m$), the system looks within the titles of the returned Web pages for the substring that best matches the selected top-scoring words.

Once each cluster has been assigned a set of descriptive and discriminative words (we call such set the cluster *signatures*), all the clusters that share the same signature are merged. This reduces the arbitrariness inherent in the choice of their number $k$, that is fixed *a priori* independently of the query.

**Outline of the paper.** The paper is organized as follows. In Section 2 we review related work on techniques for the automatic re-organization of search results. Section 3 introduces the data representation adopted within Armil and sketches the properties of the M-FPF clustering algorithm and of the cluster labelling algorithm. The results of the system evaluation are reported in Sections 5 and 4. Finally, in Section 6 conclusions and prospective future research are discussed. A full version of this paper with more details is in [8].

## 2 Previous Work

Tools for clustering Web snippets have recently become a focus of attention in the research community. In the past, this approach has had both critics [9, 10] and supporters [11], but the proliferation of commercial Web services such as Copernic, Dogpile, Groxis, iBoogie, Kartoo, Mooter, and Vivisimo seems to confirm the validity of the approach. Academic research prototypes are also available, such as Grouper [12, 6], EigenCluster [13], Shoc [14], and SnakeT [3]. Generally, details of the algorithms underlying the commercial Web services are not in the public domain.

Maarek et al. [15] give a precise characterization of the challenges inherent in Web snippet clustering, and propose an algorithm based on complete-link

hierarchical agglomerative clustering that is quadratic in the number $n$ of snippets. They introduce a technique called "lexical affinity" whereby the co-occurrence of words influences the similarity metric.

Zeng et al. [16] tackle the problem of detecting good cluster names as preliminary to the formation of the clusters, using a supervised learning approach. Note that the methods considered in our paper are instead all unsupervised, thus requiring no labelled data.

The EigenCluster [13], Lingo [17], and Shoc [14] systems all tackle Web snippet clustering by performing a singular value decomposition of the term-document incidence matrix[2]; the problem with this approach is that SVD is extremely time-consuming, hence problematic when applied to a large number of snippets. Zamir and Etzioni [12, 6] propose a Web snippet clustering mechanism (Suffix Tree Clustering – STC) based on suffix arrays, and experimentally compare STC with algorithms such as $k$-means, single-pass $k$-means [18], Backshot and Fractionation [19], and Group Average Hierarchical Agglomerative Clustering. They test the systems on a benchmark obtained by issuing 10 queries to the Metacrawler meta-search engine, retaining the top-ranked 200 snippets for each query, and manually tagging the snippets by relevance to the queries. They then compute the quality of the clustering obtained by the tested systems by ordering the generated clusters according to precision, and by equating the effectiveness of the system with the average precision of the highest-precision clusters that collectively contain 10% of the input documents. Interestingly, the authors show that very similar results are attained when full documents are used instead of their snippets, thus validating the snippet-based clustering approach.

Lawrie and Croft [4] view the clustering/labelling problem as that of generating multilevel summaries of the set of documents (in this case the Web snippets returned by a search engine). The technique is based on first building off-line a statistical model of the background language (e.g. the statistical distribution of words in a large corpus of the English language), and on subsequently extracting "topical terms" from the documents, where "topicality" is measured by the contribution of a term to the Kullback-Leibler divergence score of the document collection relative to the background language. Intuitively, this formula measures how important this term is in measuring the distance of the collection of documents from the distribution of the background language. The proposed method is shown to be superior (by using the KL-divergence) to a naive summarizer that just selects the terms with highest $tf * idf$ score in the document set.

Kammamuru et al. [5] propose a classification of Web snippet clustering algorithms into *monothetic* (in which the assignment of a snippet to a cluster is based on a single dominant feature) and *polythetic* (in which several features concur in determining the assignment of a snippet to a cluster). The rationale for proposing a monothetic algorithm is that the single discriminating feature is a natural label candidate. The authors propose such an algorithm in which the snippets are seen as sets of words and the next term is chosen so as to maximize the number of newly covered sets while minimizing the hits with already covered sets. The paper reports empirical evaluations and user studies over two classes of queries, "ambiguous" and "popular". The users were asked to compare 3 clustering algorithms over the set of queries and, for each query, were asked to answer 6 questions of a rather general nature on the generated hierarchy.

[2] The Eigencluster system is available on-line at `http://www-math.mit.edu/cluster/`

Ferragina and Gulli [3] propose a method for hierarchically clustering Web snippets, and produce a hierarchical labelling based on constructing a sequence of labelled and weighted bipartite graphs representing the individual snippets on one side and a set of labels (and corresponding clusters) on the other side. Data from the Open Directory Project (ODP)[3] is used in an off-line and query-independent way to generate predefined weights that are associated on-line to the words of the snippets returned by the queries. Data is collected from 16 search engines as a result of 77 queries chosen for their popularity among Lycos and Google users in 2004. The snippets are then clustered and the labels are manually tagged as relevant or not relevant to the cluster to which they have been associated. The clusters are ordered in terms of their weight, and quality is measured in terms of the number of relevant labels among the first $n$ labels, for $n \in \{3, 5, 7, 10\}$. Note that in this work the emphasis is on the quality of the labels rather than on that of the clusters, and that the ground truth is defined "a posteriori", after the queries are processed.

## 3 The Clustering Algorithm and the Labelling Algorithm

**The clustering algorithm.** We approach the problem of clustering Web snippets as that of finding a solution to the classic $k$-center problem: *Given a set $S$ of points in a metric space $M$ endowed with a metric distance function $D$, and given a desired number $k$ of resulting clusters, partition $S$ into non-overlapping clusters $C_1, \ldots, C_k$ and determine their "centers" $\mu_1, \ldots, \mu_k \in M$ so that the radius $\max_j \max_{x \in C_j} D(x, \mu_j)$ of the widest cluster is minimized.* The $k$-center problem can be solved approximately using the furthest-point-first (FPF) algorithm [7, 20], which we now describe. Given a set $S$ of $n$ points, FPF builds a sequence $T_1 \subset \ldots \subset T_k = T$ of $k$ sets of "centers" (with $T_i = \{\mu_1, \ldots, \mu_i\} \subset S$) in the following way.

1. At the end of iteration $i-1$ FPF holds the mapping $\mu$ defined for every point $p_j \in S \setminus T_{i-1}$ as: $\mu(p_j) = \arg\min_{\mu_s} D(p_j, \mu_s)$ i.e. the center in $T_{i-1}$ closest to $p_j$; $\mu(p_j)$ is called the *leader* of $p_j$. Note that this mapping is established in the first iteration in time $O(n)$.
2. At iteration $i$, among all points $p_j$, FPF picks $\mu_i = \arg\max_{p_j} D(p_j, \mu(p_j))$ i.e. the point for which the distance to its leader is maximum, and makes it a new center, i.e. adds it to $T_{i-1}$, thus obtaining $T_i$. This selection costs $O(n)$.
3. Compute the distance of $\mu_i$ to any point in $S \setminus T_i$ and update the mapping $\mu$ if needed. Thus $\mu$ is now correct for the beginning of iteration $i+1$. This update phase costs $O(n)$.

The final set of centers $T = \{\mu_1, \ldots, \mu_k\}$ defines the resulting $k$-clustering, since each center $\mu_i$ implicitly identifies a cluster $C_i$ as the set of data points whose leader is $\mu_i$. Note that $T_1$ is initialized to contain a single point chosen at random from $S$; this random choice is due to the fact that, in practice, both the effectiveness and the efficiency of the algorithm can be seen experimentally to be insensitive to this choice.

Most of the computation is actually devoted to computing distances and updating the auxiliary mapping $\mu$: this takes $O(n)$ time per iteration, so the total

[3] `http://www.dmoz.org/`

computational cost of the algorithm is $O(nk)$. In [1] we have thus defined an improved version of this algorithm that exploits the triangular inequality in order to filter out useless distance computations. This modified algorithm (M-FPF), which we now describe, works in any metric space, hence in any vector space[4].

Consider, in the FPF algorithm, any center $\mu_x \in T_i$ and its associated set of closest points $N(\mu_x) = \{p_j \in S \setminus T_i \mid \mu(p_j) = \mu_x\}$. We store $N(\mu_x)$ as a ranked list, in order of decreasing distance from $\mu_x$. When a new center $\mu_y$ is added to $T_i$, in order to identify its associated set of closest points $N(\mu_y)$ we scan every $N(\mu_x)$ in decreasing order of distance, and stop scanning when, for a point $p_j \in N(\mu_x)$, it is the case that $D(p_j, \mu_x) \leq \frac{1}{2} D(\mu_y, \mu_x)$. By the triangular inequality, any point $p_j$ that satisfies this condition cannot be closer to $\mu_y$ than to $\mu_x$. This rule filters out from the scan points whose leader cannot possibly be $\mu_y$, thus significantly speeding up the identification of leaders. Note that all distances between centers in $T_i$ must be available; this implies an added $O(k^2)$ cost for computing and maintaining these distances, which is anyhow dominated by the term $O(nk)$.

**Using medoids.** The M-FPF is applied to a random sample of size $\sqrt{nk}$ of the input points (this sample size is suggested in [21]). Afterwards the remaining points are associated to the closest (according to the Generalized Jaccard Distance) center. We obtain improvements in quality by making an iterative update of the "center" when a new point is associated to a cluster. Within a cluster $C_i$ we find the point $a_i$ furthest from $\mu_i$ and the point $b_i$ furthest from $a_i$ (intuitively this is a good approximation to a diametral pair). The medoid $m_i$ is the point in $C_i$ that has the minim value of the function $|D(a_i, x) - D(b_i, x)| + |D(a_i, x) + D(b_i, x)|$, over all $x \in C_i$.[5] When we add a new point to $C_i$, we check if the new point should belong to the approximate diametral pair $(a_i, b_i)$, and if so we update $m_i$ accordingly. The association of the remaining points is done with respect to the medoids, rather than the centers. The application of M-FPF plus the iterative recomputation of medoids gave us a clustering of better quality than simply using M-PFP on the whole input set.

**The distance function.** Each snippet is turned into a "bag of words" after removing stop words and performing stemming. In [1] we report experiments using, as a distance function, (i) the cosine distance measure (i.e. the complement to 1 of the cosine similarity function) applied to vectors of terms weighted by $tf*idf$, and (ii) a slight modification of the standard Jaccard Distance, which we call *Weighted Jaccard Distance* (WJD); in those experiments, (ii) has performed at the same level of accuracy as (i), but has proven much faster to compute. In this paper we improve on the results of [1] by using the Generalized Jaccard Distance described in [22]. Given two "bag-of-words" snippet vectors $s_1 = (s_1^1, ...s_1^h)$ and $s_2 = (s_2^1, ...s_2^h)$, the *Generalized Jaccard Distance* is: $D(s_1, s_2) = 1 - \frac{\sum_i \min(s_1^i, s_2^i)}{\sum_i \max(s_1^i, s_2^i)}$. The term weights $s_a^i$ consist of "weighted term frequencies", obtained as weighted sums of the numbers of occurrences of the term in the snippet, where weight 3 is assigned to a term occurring in the page title, weight 1 to a term occurring in the text fragment, and weight 0 is assigned to a term occurring in the URL (since,

[4] We recall that any vector space is also a metric space, but not vice-versa.

[5] This formula mimics in a discrete setting the task of finding the cluster point closest to the median point to the segment $(a_i, b_i)$.

in previous experiments we had run, the text of the URL had proven to give no contribution in terms of cluster quality). Note that, when using unit weights only, the Generalized Jaccard Distance coincides with the standard Jaccard Distance.

**The candidate words selection algorithm.** We select candidate terms for labelling the generated clusters through a modified version of the information gain function [2]. For term $t$ and category $c$, information gain is defined as $IG(t,c) = \sum_{x\in\{t,\bar{t}\}}\sum_{y\in\{c,\bar{c}\}} P(x,y)\log\frac{P(x,y)}{P(x)P(y)}$ Intuitively, $IG$ measures the amount of information that each argument contains about the other; when $t$ and $c$ are independent, $IG(t,c)=0$. This function is often used for feature selection in text classification, where, if $IG(t,c)$ is high, the presence or absence of a term $t$ is deemed to be highly indicative of the membership or non-membership in a category $c$ of the document containing it. In the text classification context, the rationale of including in the sum, aside from the factor that represents the "positive correlation" between the arguments (i.e. the factor $P(t,c)\log\frac{P(t,c)}{P(t)P(c)} + P(\bar{t},\bar{c})\log\frac{P(\bar{t},\bar{c})}{P(\bar{t})P(\bar{c})}$), also the factor that represents their "negative correlation" (i.e. the factor $P(\bar{t},c)\log\frac{P(\bar{t},c)}{P(\bar{t})P(c)} + P(t,\bar{c})\log\frac{P(t,\bar{c})}{P(t)P(\bar{c})}$), is that, if this latter factor has a high value, this means that the absence (resp. presence) of $t$ is highly indicative of the membership (resp. non-membership) of the document in $c$. That is, the term is useful anyway, although in a "negative" sense.

However, in our context we are interested in terms that *positively describe* the contents of a cluster, and are thus only interested in positive correlation. Therefore, we drop the factor denoting negative correlation from the $IG$ formula, yielding the modified version $IG_m(t,c) = P(t,c)\log\frac{P(t,c)}{P(t)P(c)} + P(\bar{t},\bar{c})\log\frac{P(\bar{t},\bar{c})}{P(\bar{t})P(\bar{c})}$ that coincides with the positive correlation factor of $IG$. We use $IG_m$ to select, for each cluster, words that are *representative* of the cluster and, at the same time, allow to discriminate among clusters.

## 4 Experimental Evaluation of the Clustering Algorithm

**The baseline.** As baseline against which to compare the clustering capabilities of Armil, we have chosen Vivisimo[6]. Vivisimo is considered an industrial standard in terms of clustering quality and user satisfaction, and in 2001 and 2002 it has won the "best meta-search-award" assigned annually by the on-line magazine SearchEngineWatch.com. Vivisimo thus represents a particularly difficult baseline, and it is not known if its clustering quality only depends on an extremely good clustering algorithm, or rather on the use of external knowledge or custom-developed resources. To the best of our knowledge, this is the first published experiment comparing the clustering quality of an academic prototype and Vivisimo. Vivisimo's advanced searching feature allows a restriction of the considered auxiliary search engines to a subset of a range of possible auxiliary search engines. For the purpose of our experiment we restrict our source of snippets to the ODP directory.

**Measuring clustering quality.** Following a consolidated practice, in this paper we measure the effectiveness of a clustering system by the degree to which it is able to "correctly" re-classify a set of pre-classified snippets into exactly the same

[6] `http://vivisimo.com/`

categories without knowing the original category assignment. In other words, given a set $C = \{c_1, \ldots, c_k\}$ of categories, and a set $\Theta$ of $n$ snippets pre-classified under $C$, the "ideal" term clustering algorithm is the one that, when asked to cluster $\Theta$ into $k$ groups, produces a grouping $C' = \{c'_1, \ldots, c'_k\}$ such that, for each snippet $s_j \in \Theta$, $s_j \in c_i$ if and only if $s_j \in c'_i$. The original labelling is thus viewed as the latent, hidden structure that the clustering system must discover.

The measure we use is *normalized mutual information* (see e.g. [23, page 110]), i.e. $NMI(C, C') = \frac{2}{\log |C||C'|} \sum_{c \in C} \sum_{c' \in C'} P(c, c') \cdot \log \frac{P(c, c')}{P(c) \cdot P(c')}$ where $P(c)$ represents the probability that a randomly selected snippet $s_j$ belongs to $c$, and $P(c, c')$ represents the probability that a randomly selected snippet $s_j$ belongs to both $c$ and $c'$. The normalization, achieved by the $\frac{2}{\log |C||C'|}$ factor, is necessary in order to account for the fact that the cardinalities of $C$ and $C'$ are in general different [2]. Higher values of $NMI$ mean better clustering quality. The clustering produced by Vivisimo has partially overlapping clusters (in our experiments Vivisimo assigned roughly 27% of the snippets to more than one cluster), but $NMI$ is designed for non-overlapping clustering. Therefore, in measuring $NMI$ we eliminate from the ground truth, from the clustering produced by Vivisimo, and from that produced by Armil, the snippets that are present in multiple copies.

However, in order to also consider the ability of the two systems to "correctly" duplicate snippets across overlapping clusters, we have also computed the *normalized complementary entropy* [23, page 108], in which we have changed the normalization factor so as to take overlapping clusters into account. The entropy of a cluster $c'_l \in C'$ is $E(c'_l, C) = \sum_{k=1}^{|C|} -\frac{|c'_l \cap c_k|}{|c_k|} \log \frac{|c'_l \cap c_k|}{|c_k|}$. The normalized complementary entropy of $c'_l$ is $NCE(c'_l, C) = 1 - \frac{E(c'_l, C)}{\log |C|}$. $NCE$ ranges in the interval $[0, 1]$, and a greater value implies better quality of $c'_l$. The complementary normalized entropy of $C'$ is the weighted average of the contributions of the single clusters in $C'$. Let $n' = \sum_{l \in 1}^{|C'|} |c'_l|$ be the sum of the cardinalities of the clusters of $C'$. Note that when clusters may overlap it holds that $n' \geq n$. Thus $NCE(C', C) = \sum_{l \in 1}^{|C'|} \frac{|c'_l|}{n'} NCE(c'_l, C)$. $NCE$ values reported below are thus obtained on the full set of snippets returned by Vivisimo.

**Establishing the ground truth.** Following [24], we have made a series of experiments using as input the snippets resulting from queries issued to the Open Directory Project (ODP – see Footnote 3). The ODP is a searchable Web-based directory consisting of a collection of a few million Web pages (as of today, ODP claims to index 5.1M Web pages) pre-classified into more than 590K categories by a group of volunteer human experts. The classification induced by the ODP labelling scheme gives us an objective "ground truth" against which we can compare the clustering quality of Vivisimo and Armil. In ODP, documents are organized according to a hierarchical ontology. For any snippet we obtain a label for its class by considering only the first two levels of the path on the ODP category tree. This coarsification is needed in order to balance the number of classes and the number of snippets returned by a query.

Queries are submitted to Vivisimo, asking it to retrieve pages only from ODP. This is done to ensure that Vivisimo and Armil operate on the same set of snippets, hence to ensure full comparability of the results. The resulting set of snippets

**Table 1.** Results of the comparative evaluation

| | Vivisimo | Armil(40) | Armil(30) |
|---|---|---|---|
| $NCE$ | 0.667 | 0.735 (+10.1%) | 0.683 (+2.3%) |
| $NMI$ | 0.400 | 0.442 (+10.5%) | 0.406 (+1.5%) |

is parsed and given as input to Armil. Since Vivisimo does not report the ODP category to which a snippet belongs, for each snippet we perform a query to ODP in order to establish its ODP-category.

**Outcome of the comparative experiment.** The queries used in this experiment are the last 30 of those reported in Appendix A (the first 5 have been excluded since too few related snippets are present in ODP). On average, ODP returned 41.2 categories for each query. In Table 1 we report the $NMI$ and $NCE$ values obtained by Vivisimo and Armil on these data. Vivisimo produced by default about 40 clusters; therefore we have run Armil with a target of 40 clusters (thus with a choice close to that of Vivisimo, and to the actual average number of ODP categories per query) and with 30 (this number is the default used in the user evaluation).

The experiments indicate an substantial improvement of about 10% in terms of cluster quality of Armil(40) with respect to Vivisimo.[7] This improvement is an important result since, as noted in 2005 in [3], "[T]he scientific literature offers several solutions to the web-snippet clustering problem, but unfortunately the attainable performance is far from the one achieved by Vivisimo." It should be noted moreover that Vivisimo uses a proprietary algorithm, not in the public domain, which might make extensive use of external knowledge. In contrast our algorithm is open and disclosed to the research community.

## 5 User Evaluation of the Cluster Labelling Algorithm

Assessing "objectively" the quality of a cluster labelling method is a difficult problem, for which no established methodology has gained a wide acceptance. For this reason a user study is the standard testing methodology. We have set up a user evaluation of the cluster labelling component of Armil in order to have an independent and measurable assessment of its performance. We performed the study on 22 volunteer master students, doctoral students and post-docs in computer science at our departments. The volunteers have all a working knowledge of the English language.

The user interface of Armil has been modified so as to show clusters one-by-one and proceed only when the currently shown cluster has been evaluated. The queries are supplied to the evaluators in a round robin fashion from a list of 35 predefined queries. For each query the user must first say whether the query is meaningful to her; an evaluator is allowed to evaluate only queries meaningful to her. For each cluster we propose three questions: (a) Is the label syntactically well-formed?; (b) Can you guess the content of the cluster from the label?; (c) After inspecting the cluster, do you retrospectively consider the cluster as well described

[7] For the sake of replicating the experiments all the search results have been cached and are available at `http://psp1.iit.cnr.it/~mcsoft/armil` .

**Table 2.** Correlation tables of questions row-(a) and column-(b) (left), row-(b) and column-(c) (middle), row-(a) and column-(c) (right). Entries in the top part give the percentage over all answers, and entries in the bottom part give percentage over rows.

| | Yes | Sort-of | No | Yes | Sort-of | No | Yes | Sort-of | No |
|---|---|---|---|---|---|---|---|---|---|
| Yes | 42.67% | 12.81% | 5.11% | 33.52% | 12.81% | 3.72% | 35.98% | 18.93% | 5.68% |
| Sort-of | 5.74% | 15.27% | 4.41% | 11.36% | 16.85% | 3.66% | 8.64% | 12.81% | 3.97% |
| No | 1.64% | 3.78% | 8.52% | 2.14% | 8.90% | 7.00% | 2.39% | 6.81% | 4.73% |
| Yes | 70.41% | 21.14% | 8.43% | 66.96% | 25.59% | 7.44% | 59.37% | 31.25% | 9.37% |
| Sort-of | 22.58% | 60.04% | 17.36% | 35.64% | 52.87% | 11.48% | 33.99% | 50.37% | 15.63% |
| No | 11.76% | 27.14% | 61.08% | 11.88% | 49.30% | 38.81% | 17.19% | 48.86% | 33.93% |

by the label? The evaluator must choose one of three possible answers (Yes; Sort-of; No), and her answer is automatically recorded in a database. Question (a) is aimed at assessing the gracefulness of the label produced. Question (b) is aimed at assessing the quality of the label as an instrument predictive of the cluster content. Question (c) is aimed at assessing the correspondence of the label with the content of the cluster. Note that the user cannot inspect the content of the cluster before answering (a) and (b).

**Selection of the queries**. Similarly to [3, 5], we have randomly selected 35 of the most popular queries submitted to Google in 2004 and 2005[8]; from the selection we have removed queries (such as e.g. "Spongebob", "Hilary Duff") that, referring to someone or something of regional interest only, were unlikely to be meaningful to our evaluators. The queries are listed in Appendix A.

**Discussion of the results**. Each of the 35 queries has been evaluated by two different evaluators, for a total of 70 query evaluations and 1584 cluster evaluations. The results are displayed in the following table:

| | Yes | Sort-of | No |
|---|---|---|---|
| (a) | 60.5% | 25.5% | 14.0% |
| (b) | 50.0% | 32.0% | 18.0% |
| (c) | 47.0% | 38.5% | 14.5% |

By checking the percentages of No answers, we can notice that sometimes labels considered non-predictive are nonetheless considered well descriptive of the cluster; we interpret this fact as due to the discovery of meanings of the query string previously unknown to the evaluator. The correlation matrices in Table 2 show more precisely the correlation between syntax, predictivity and representativeness of the labels. The data in Table 2 (left) show that there is a strong correlation between syntactic form and predictivity of the labels, as shown by the fact that in a high percentage of cases the same answer was returned to questions (a) and (b). The middle and right part of Table 2 confirms that while for the positive or mildly positive answers (Yes, Sort-of) there is a strong correlation between the answers returned to the different questions, it is often the case that a label considered not predictive of the content of the cluster can still be found, after inspection of the cluster, to be representative of the content of the cluster.

**Running times.** Our system runs on an AMD Athlon (1Ghz Clock) processor with 750Mb RAM and operating system FreeBSD 4.11 - STABLE. The code

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

was developed in Python V. 2.4.1. Excluding the time needed to download the snippets from the auxiliary search engines, the 35 queries have been clustered and labelled in 0.72 seconds on average; the slowest query took 0.92 seconds.

## 6 Conclusions and Future Work

Why is Armil not "yet another clustering search engine"? The debate on how to improve the performance of search engines is at the core of the current research in the area of Web studies, and we believe that so far only the surface of the vein has been uncovered. The main philosophy of the system/experiments we have proposed follows these lines: (i) principled algorithmic choices are made whenever possible; (ii) clustering is clearly decoupled from labelling; (iii) attention is paid to the trade-off between response time and quality while limiting the response time within limits acceptable by the user; (iv) a comparative study of Armil and Vivisimo has been performed in order to assess the quality of Armil's clustering phase by means of effectiveness measures commonly used in clustering studies; (v) a user study has been set up in order to obtain an indication of user satisfaction with the produced cluster labelling; (vi) no use of external sources of knowledge is made.

Further research is needed in two main areas. First, we plan to assess to what extent a modicum of external knowledge can improve the system's performance without speed penalties. Second, it is possible to introduce in the current pipeline (input snippets are clustered, candidates are extracted, labels are generated) of the architecture a feedback loop by considering the extracted candidates/labels as predefined categories, thus examining which snippets in different clusters are closer to the generated labels. Snippets close to the label of cluster $C_x$ but in a different cluster $C_y$ could be shown on the screen as related also to $C_x$. This would give the benefits of soft clustering without much computational overload.

## References

1. Geraci, F., Pellegrini, M., Pisati, P., Sebastiani, F.: A scalable algorithm for high-quality clustering of Web snippets. In: Proceedings of SAC-06, 21st ACM Symposium on Applied Computing, Dijon, FR (2006) 1058–1062
2. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons, New York, US (1991)
3. Ferragina, P., Gulli, A.: A personalized search engine based on Web-snippet hierarchical clustering. In: Special Interest Tracks and Poster Proceedings of WWW-05, 14th International Conference on the World Wide Web, Chiba, JP (2005) 801–810
4. Lawrie, D.J., Croft, W.B.: Generating hierarchical summaries for Web searches. In: Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval. (2003) 457–458
5. Kummamuru, K., Lotlikar, R., Roy, S., Singal, K., Krishnapuram, R.: A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In: Proceedings of WWW-04, 13th International Conference on the World Wide Web, New York, NY (2004) 658–665
6. Zamir, O., Etzioni, O., Madani, O., Karp, R.M.: Fast and intuitive clustering of Web documents. In: Proceedings of KDD-97, 3rd International Conference on Knowledge Discovery and Data Mining, Newport Beach, US (1997) 287–290
7. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38**(2/3) (1985) 293–306

8. Geraci, F., Pellegrini, M., Sebastiani, F., Maggini, M.: Cluster generation and cluster labelling for web snippets: A fast and accurate hierarchical solution. Technical Report IIT TR-1/2006, Institute for Informatics and Telematics of CNR (2006)
9. Kural, Y., Robertson, S., Jones, S.: Clustering information retrieval search outputs. In: Proceedings of the 21st BCS IRSG Colloquium on Information Retrieval, Glasgow, UK (1999)
10. Kural, Y., Robertson, S., Jones, S.: Deciphering cluster representations. Information Processing and Management **37** (1993) 593–601
11. Tombros, A., Villa, R., van Rijsbergen, C.J.: The effectiveness of query-specific hierarchic clustering in information retrieval. Information Processing and Management **38**(4) (2002) 559–582
12. Zamir, O., Etzioni, O.: Web document clustering: A feasibility demonstration. In: Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, AU (1998) 46–54
13. Cheng, D., Kannan, R., Vempala, S., Wang, G.: On a recursive spectral algorithm for clustering from pairwise similarities. Technical Report MIT-LCS-TR-906, Massachusetts Institute of Technology, Cambridge, US (2003)
14. Zhang, D., Dong, Y.: Semantic, hierarchical, online clustering of Web search results. In: Proceedings of APWEB-04, 6th Asia-Pacific Web Conference, Hangzhou, CN (2004) 69–78
15. Maarek, Y., Fagin, R., Ben-Shaul, I., Pelleg, D.: Ephemeral document clustering for Web applications. Technical Report RJ 10186, IBM, San Jose, US (2000)
16. Zeng, H.J., He, Q.C., Chen, Z., Ma, W.Y., Ma, J.: Learning to cluster Web search results. In: Proceedings of SIGIR-04, 27th ACM International Conference on Research and Development in Information Retrieval, Sheffield, UK (2004) 210–217
17. Osinski, S., Weiss, D.: Conceptual clustering using Lingo algorithm: Evaluation on Open Directory Project data. In: Proceedings of IIPWM-04, 5th Conference on Intelligent Information Processing and Web Mining, Zakopane, PL (2004) 369–377
18. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. Volume 1. (1967) 281–297
19. Cutting, D.R., Pedersen, J.O., Karger, D., Tukey, J.W.: Scatter/Gather: A cluster-based approach to browsing large document collections. In: Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval, Kobenhavn, DK (1992) 318–329
20. Hochbaum, D.S., Shmoys, D.B.: A best possible approximation algorithm for the $k$-center problem. Mathematics of Operations Research **10**(2) (1985) 180–184
21. Indyk, P.: Sublinear time algorithms for metric space problems. In: Proceedings of STOC-99, ACM Symposium on Theory of Computing. (1999) 428–434
22. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proceedings of STOC-02, 34th Annual ACM Symposium on the Theory of Computing, Montreal, CA (2002) 380–388
23. Strehl, A.: Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining. PhD thesis, University of Texas, Austin, US (2002)
24. Haveliwala, T.H., Gionis, A., Klein, D., Indyk, P.: Evaluating strategies for similarity search on the Web. In: Proceedings of WWW-02, 11th International Conference on the World Wide Web, Honolulu, US (2002) 432–442

## A Queries Used in the User Evaluation

skype, winmx, nintendo revolution, pamela anderson, twin towers, wallpaper, firefox, ipod, tsunami, tour de france, weather, matrix, mp3, new orleans, notre dame, games, britney spears, chat, CNN, iraq, james bond, harry potter, simpsons, south park, baseball, ebay, madonna, star wars, tiger, airbus, oscars, london, pink floyd, armstrong, spiderman.