Consiglio Nazionale delle Ricerche

Cluster Generation and Cluster Labelling for Web Snippets: A Fast and Accurate Hierarchical Solution

F. Geraci, M. Pellegrini, M. Maggini, F. Sebastiani

IIT TR-01/2006

Technical report

Febbraio 2006

Istituto di Informatica e Telematica

Cluster Generation and Cluster Labelling for Web Snippets: A Fast and Accurate Hierarchical Solution

Filippo Geraci[1,2], Marco Pellegrini[1], Marco Maggini[2], Fabrizio Sebastiani[3]
(1) Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche
Via G Moruzzi, 1 – 56124 Pisa, Italy
(2) Dipartimento di Ingegneria dell'Informazione, Università di Siena
Via Roma, 56 – 53100 Siena, Italy
(3) Dipartimento di Matematica Pura e Applicata, Università di Padova
Via GB Belzoni, 7 – 35131 Padova, Italy
{f.geraci,m.pellegrini}@iit.cnr.it maggini@ing.unisi.it fabrizio.sebastiani@unipd.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. On a standard 1GHz machine, Armil performs clustering and labelling altogether in less than one second.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; H.3.4 [**Information Storage and Retrieval**]: Systems and software—*Web (WWW)*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

General Terms

Algorithm, Design, Experimentation, Measurements

Keywords

Web snippets, clustering, meta-search engines, metric spaces, information gain

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. Ranking algorithms play a crucial role in this approach, since users usually browse at most the 10 top-ranked items. Snippet quality is also an important issue, since good-quality snippets allow the user to determine whether the referred pages match or not her information need. In order to provide a useful hint about the real content of the page, a Web snippet includes both the page title and a short text fragment, that often displays the query terms in context.*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 can even worsen the quality of the information access experience, since the user is in principle 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 retrieved by the different auxiliary search engines (while at the same time removing the duplicates), and the design of advanced user interfaces based on a structured organization of the results, so as to help the user to focus on the most relevant subset of results. This latter aspect is 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

[1]An armillary sphere (also known as a spherical astrolabe, armilla, or *armil*) is a navigation tool in the form of a model of the celestial sphere, invented by Eratosthenes in 255 BC. Renaissance scientists and public figures were often portrayed with one hand on an armil, since it represented the height of wisdom and knowledge (see `http://en.wikipedia.org/wiki/Armillary_sphere`).
The Armil system can be freely accessed at `http://ubi8.imc.pi.cnr.it/`.

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 [6]. 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.

1.1 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 view label extraction as the primary goal, and clustering is a by-product of the label extraction procedure. Other systems view 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 computation. As such, M-FPF inherits a very important property of the FPF algorithm, i.e. it is 2-competitive for the k-center problem. In other words, for any fixed number k, M-FPF produces a k-clustering (i.e. a partition of the items into k non-overlapping clusters) such that the maximum cluster diameter is at most twice as large as that of the "optimal" k-clustering (i.e. the one that minimizes such maximum diameter). The competitive property of M-FPF is even stronger: the approximation factor of 2 cannot be improved with any polynomial approximation algorithm, unless $P = NP$. The strength of this formal property has been our main motivation for selecting M-FPF as the algorithmic backbone for Armil. 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[2]. M-FPF relies instead only on pairwise distance calculations between snippets, and as such better exploits the sparsity of the snippet vector representations.

1.2 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 candidates words for each cluster by means of IG_m, a modified version of the *Information Gain* measure [3]. 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.

1.3 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. In Section 4 the architecture of Armil is described in detail. The results of the system evaluation are reported in Sections 7 and 6. Finally, in Section 8 conclusions and prospective future research are discussed.

2. PREVIOUS WORK

Tools for clustering Web snippets have recently become a focus of attention in the research community, also as a result of the success of commercial Web services such as Copernic, Dogpile, Groxis, iBoogie, Kartoo, Mooter, and Vivisimo. Academic research prototypes are also available, such as

[2]The clustering literature also discusses the notion of cluster "medoid"; similarly to a centroid, a cluster medoid plays the role of cluster representative, but typically has a sparse representation. Unfortunately, the methods proposed in the literature for finding high-quality medoids are not compatible with the real-time nature of the envisioned online Web service.

Grouper [17, 18], EigenCluster [2], Shoc [20], and SnakeT [5]. Generally, details of the algorithms underlying the commercial Web services are not in the public domain.

Maarek et al. [13] 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. [19] 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 [2], Lingo [15], and Shoc [20] systems all tackle Web snippet clustering by performing a singular value decomposition of the term-document incidence matrix[3]; the problem with this approach is that SVD is extremely time-consuming, hence problematic when applied to a large number of snippets. By testing a number of queries on Eigencluster we have observed that, when operating on many snippets (roughly 400), a reasonable response time (under 1 second) is attained by limiting the number of generated clusters to a number between 5 and 10, and avoiding a clustering decision for over 50% of the data. Zamir and Etzioni [17, 18] 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 [14], Backshot and Fractionation [4], and Group Average Hierarchical Agglomerative Clustering (GAHAC). 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. This methodology had been advocated in [9], and is based on the assumption that the users will anyway be able to spot the clusters most relevant to their query. Average precision as computed with this method ranges from 0.2 to 0.4 for all the algorithms tested (STC coming out on top in terms of both effectiveness and efficiency). 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 [12] 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. Additionally, the "predictiveness" of each term is measured. Intuitively, predictiveness measures how close a term appears (within a given window size) to other terms. In the summaries, terms of high topicality and high predictiveness are preferred.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. [11] 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.

Ferragina and Gulli [5] 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)[4] 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 (although the two concepts are certainly related), and that the ground truth is defined "a posteriori", after the queries are processed.

3. THE CLUSTERING ALGORITHM AND THE LABELLING ALGORITHM

We now describe in detail the methods used by Armil for Web snippet clustering and cluster labelling.

3.1 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*

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

[4]http://www.dmoz.org/

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, 10], 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. At iteration i

1. for every point $p_j \in S \setminus T_{i-1}$, FPF determines

$$\mu(p_j) = \arg\min_{\mu_s} D(p_j, \mu_s)$$

   i.e. the center in T_{i-1} closest to p_j; $\mu(p_j)$ is called the *neighbour* of p_j;

2. of all points p_j, FPF picks

$$\mu_i = \arg\max_{p_j} D(p_j, \mu(p_j))$$

   i.e. the point for which such minimal distance is maximum, and makes it a center, i.e. adds it to T_{i-1}, thus obtaining T_i.

The final set of centers $T = \{\mu_1, \ldots, \mu_k\}$ defines the resulting k-clustering, since each center μ_i implicitly identifies a cluster C_i as the set of data points whose neighbour is μ_i.

Most of the computation is actually devoted to computing distances: if this is done in a straightforward manner, i.e. computing the distance of each point from each center in T_{i-1}, thist takes $O(n)$ time per iteration, so the total computational cost of the algorithm is $O(nk)$. In [6] 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[5].

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 μ_x. When a new center μ_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 μ_y than to μ_x. This rule filters out from the scan points whose neighbour cannot possibly be μ_y, thus significantly speeding up the identification of neighbours. 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.

3.1.1 Using medoids

The M-FPF is applied the a random sample of size $\sqrt{nk}$ of the input points. Afterwards the remaining points are associated to the closest 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 μ_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

$$|D(a_i, x) - D(b_i, x)| + |D(a_i, x) + D(b_i, x) - D(a_i, b_i)|,$$

over all $x \in C_i$.[6] 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.

3.1.2 The distance function

Each snippet is turned into a "bag of words" after removing stop words and performing stemming. In [6] we report experiments using, as a similarity function, (i) the cosine similarity measure applied to vectors of terms weighted by $tf * idf$, and (ii) a slight modification of the standard Jaccard Coefficient, which we call *Weighted Jaccard Coefficient* (WJC); in those experiments, (ii) has performed at the same level of accuracy as (i), but has proven much faster to compute. The WJC takes advantage of the intrinsic structure of the snippets, by weighting different parts of the snippet (page title, text fragment, URL) differently; more precisely, weight 3 is assigned to the page title, weight 1 to the text fragment, while the URL is ignored (since, in previous experiments we had run, the text of the URL had proven to give no contribution in terms of cluster quality). So, our WJC distance is defined as

$$d(s_1, s_2) = \begin{cases} 1 & \text{if } |s_1 \cap s_2| = 0 \\ 0 & \text{if } 2p(s_1, s_2) \geq |s_1| + |s_2| \\ 1 - \frac{p(s_1,s_2)}{|s_1|+|s_2|-p(s_1,s_2)} & \text{otherwise} \end{cases}$$

where

$$p(s_1, s_2) = \sum_{i \in s_1 \cap s_2} \frac{w(s_1, i) + w(s_2, i)}{2}$$

and $w(s, i)$ is the weighted number of occurrences of word i in snippet s. In this paper we use a generalized Jaccard distance described in [1]. Given two "bag-of-words" snippet vectors $s_1 = (s_1^1, \ldots s_1^h)$ and $s_2 = (s_2^1, \ldots s_2^h)$, the generalized Jaccard distance is:

$$D(s_1, s_2) = 1 - \frac{\sum_i \min(s_1^i, s_2^i)}{\sum_i \max(s_1^i, s_2^i)}.$$

We take advantage of the structure of the snippets, by weighting different parts of the snippet (page title, text fragment, URL) differently; more precisely, weight 3 is assigned to the page title, weight 1 to the text fragment, while the URL is ignored (since, in previous experiments we had run, the text of the URL had proven to give no contribution in terms of cluster quality). Note that using uniform weights the above formula coincides with the standard Jaccard distance.

3.2 The labelling algorithm

We select terms candidate for labelling the generated clusters through a modified version of the information gain function [3]. For term t and category c, information gain is defined as

$$IG(t, c) = \sum_{x \in \{t, \bar{t}\}} \sum_{y \in \{c, \bar{c}\}} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

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

[6] 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).

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. OVERVIEW OF THE SYSTEM

We discuss here in more detail the architecture of Armil. Overall the computation flow is a pipeline consisting in (i) data collection and cleaning, (ii) first-level clustering, (iii) candidate word extraction for labelling, (iv) second-level clustering, and (v) cluster labelling. Let us review these steps in order.

(1) Querying one or more search engines: The user of Armil issues a query string that is re-directed by Armil to the selected search engines (at the moment the user can select Google and/or Yahoo!). As a result Armil obtains a list (or several lists) of snippets describing Web pages that the search engines deem relevant to the query. An important system design issue is deciding the type and number of snippet sources to be used as auxiliary search engines. It is well-known that the probability of relevance of a snippet to the user information need quickly decays with the rank of the snippet in the list returned by the search engine. Therefore the need of avoiding low-quality snippets suggests the use of many sources each supplying a low number of high-quality snippets. On the other hand, increasing the number of snippet sources raises the pragmatic issue of handling several concurrent threads, the need to detect and handle more duplicates, and the need for a more complex handling of the composite ranking by merging several snippet lists (both globally and within each cluster separately). Since we consider Armil a "proof-of-concept" prototype rather than a full-blown service, we have chosen only two (high-quality) sources, Google and Yahoo!. We limit the total number of snippets to be collected to 200 (of which 80 from Yahoo! and 120 from Google; these numbers optimize the total waiting time). We produce the initial composite ranking of the merged snippet list by a very simple method, i.e. by alternatively picking snippets from each source list.

(2) Cleaning and filtering: The input is filtered by removing non-alphabetic symbols, digits, HTML tags, stop words, and the query terms. These latter are removed since they are likely to be present in every snippet, and thus are going to be useless for the purpose of discriminating different contexts. We then identify the prevalent language of each snippet, which allows us to choose the appropriate stop word list and stemming algorithm. Currently we use the ccTLD of the url to decide on the prevalent language of a snippet. For the purpose of the experiments we only distinguish between English and Italian. For snippets of English Web pages we use Porter's stemming algorithm, while for [other language] ones we use a simple rule-based stemmer we developed in-house. Currently, no other languages are supported.

(3) First-level clustering: We build a flat k-clustering representing the first level of the cluster hierarchy, using the k-center algorithm and the Generalized Jaccard Coefficient as described in Section 3. An important issue is deciding the number k of clusters to create. Currently, this number is fixed to 30, but it is clear that the number of clusters should depend on the query and on the number of snippets found. A general rule seems difficult to find; therefore, besides providing a default value,, we allow the user to increase or decrease the value of k to her liking. Clusters that contain one snippet only are probably outliers of some sort, and we thus merge them under a single cluster labelled "Other topics".

(4) Cluster re-ranking: A cluster small enough that the list of its snippets fits in the screen does not require a sophisticated order of presentation. However, in general users are greatly facilitated if the snippets in a cluster are listed in order of their likely importance to the user. We identify the "core" of the cluster by applying the M-FPF algorithm within the cluster, using as termination criterion the emergence of a cluster with roughly half of the cluster points. The point in the "core" are shown in the listing before those not in the core. Within core and non-core points we use a relative ranking obtained by a linear combination of the native ranking generated by the auxiliary search engines.

(5) Candidate words identification: For each cluster we need to determine a set of words candidate for appearing in its label; these will hereafter be referred as *keywords*. For this purpose, for each word that occurs in the cluster we sum the weights of all its occurrences in the cluster and pre-select the 10 words with the highest global score. We refer to this as *local keyword selection*, since it is done independently for each cluster. For each of the 10 selected terms we compute IG_m, as explained in Section 3.2. The three terms with the highest score are chosen as keywords. We refer to this as *global keyword selection*, because the computation of IG_m for a term in a cluster is dependent also on the contents of the other clusters. Global selection has the purpose of obtaining different labels for different clusters. At the end of this procedure, if two clusters have the same signature we merge them, since this is an indication that the target number of clusters k may have been too high for this particular query[7].

(6) Second-level clustering: Although the clustering algorithm could in principle be iterated recursively in each cluster up to an arbitrary number of levels, we limit our algorithm to only two levels, since this is likely to be the maximum depth a user is willing to explore in searching for an answer to her information needs. Second-level clustering is applied to first-level clusters larger than a predetermined threshold (at the moment this is fixed to 10 snippets, excluding duplicates). For second-level clustering we adopt a different approach, since metric-based clustering applied at the second level tends to detect a single large "core" cluster and several small "outlier" clusters. The second-level part of the hierarchy is generated based on the keywords found during first-level clustering. For the identified set K of three keywords we consider all its subsets as candidate signatures. A snippet x is assigned to a signature s if and only if all the signature elements are in x and no keyword in $K \setminus s$ is in x. If a signature is assigned too few snippets (e.g. 1) it is considered as an outlier and it is not shown to the user. Also, if most of the snippets at the first level end up associated to a single signature, then the second-level clusters are not shown to the user since the second-level subdivision would not be any more useful than that the first-level subdivision.

(7) Labelling: Lists of keywords are not an intuitive way of conveying meaning. Therefore we choose to use the keywords just as a basis for generating well-formed phrases that will be shown to the user as real cluster labels. Given the title of the Web page contained in the snippet, considered as a sequence of words (this time including stop words) we consider all its contiguous subsequences and we give each subsequence a score as follows: keywords are given a high positive score, query words a low positive score, all other words a high negative score. For labelling a cluster, among all its snippets we select the shortest substring of a snippet title among those having the highest score.

(8) Duplicate removal: Since Armil collects data from several search engines it is possible that the same URL (maybe with a different snippet fragment of text) is present in duplicate. We consider this fact as an indication of the importance of the URL. Therefore, duplicates are accounted for in determining weights and distances. Since clustering is based on title and text, it is possible that the same URL ends up in different clusters, for which it is equally relevant. However, if duplicate snippets appear in the same cluster, they are listed only once. Thus duplicate removal is done just before the presentation to the user.

(9) User Interface: The user interface is important for the success of a Web-based service. We have adopted a scheme common to many search engines and meta-search engines (e.g. Vivisimo), in which the data are shown in ranked list format in the main frame while the list of cluster labels are presented on the left frame as a navigation tree. The interface also allows the user to select the number of clusters, by increasing or decreasing the default value of 30.

[7]More precisely, we consider the two original clusters with the same signature as second-level clusters, and we produce for each a different second-level label.

5. ANECDOTAL EVIDENCE

In this section we report an analysis of the output of the system on selected queries. In particular we highlight some very high quality clusters obtained.

5.1 Query: armstrong

The query "armstrong" returns 174 snippets organized in 28 clusters, with a clustering time of 0.94 seconds (see Figure 1). Cluster #2, labelled "Armstrong Ceilings", contains 12 snippets, among which 5 are relative to a company manufacturing ceilings and floors, 4 to companies manufacturing other hardware equipment (e.g. molds, pumps, tools), and one to companies manufacturing medical equipment. Cluster #3, labelled "Lance Armstrong Foundation", contains 14 snippets, of which 12 are related to the sport champion. Cluster number #4, labelled "Luis Jazz", contains 20 snippets all related to the well-known jazz musician and mostly in English, while Cluster #6, with the same label, contains 12 snippets of which 9 are relative to the same musician but mostly in Italian, and 3 are relative to other musicians named Armstrong. Cluster #12, labelled "Neil Armstrong", has 7 snippets of which 6 are related to the well-known astronaut. Cluster #19, labelled "George", contains 4 snippets, two of which are related to the life of General George Armstrong Custer. Cluster #18, labelled "Edwin Howard Armstrong", contains 6 snippets, three of which are devoted to Edwin Howard Armstrong, the inventor of the frequency modulation radio broadcasting technique.

5.2 Query: madonna

The query "madonna" returns 175 snippets organized in 19 clusters (after manual adjustment) in 0.88 seconds (see Figure 2). Cluster #4, labelled "santuario della madonna del divino", contains 9 snippets, of which 6 are relative to holy shrines. Cluster #5, labelled "su louise veronica madonna ciccone", contains 11 snippets, all of them high-quality sites devoted to the well-known popstar. Cluster #7, labelled "Madonna di Campiglio", contains 13 snippets, of which 11 are related to the Italian ski resort. Cluster #10, labelled "music", contains 51 snippets, most of which related to the popstar.

5.3 Query: allergy

The query "allergy" returns 100 snippets organized in 27 clusters in 0.94 seconds (see Figure 3).

Cluster #3 labelled "Asthma" contains 8 snippets, of which 6 related to asthma and allergy. Cluster #5 labelled "Allergy Journal" contains 4 snippets, all of which relate to periodic medical publications. Cluster #6 labelled "guide to allergies and allergy" contains 7 snippets, of which at least 5 offer general advise on allergy treatments. Cluster #11 labelled "food allergy" contains 5 snippets, all of which related to food allergies.

6. EXPERIMENTAL EVALUATION OF THE CLUSTERING ALGORITHM

We have performed an experiment aimed at assessing the performance of the clustering algorithm that Armil uses.

6.1 The baseline

Figure 1: Labels generated by Armil for the query "armstrong".

As baseline against which to compare the clustering capabilities of Armil, we have chosen Vivisimo[8]. 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 on an objective basis (see below) 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.

6.2 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 categories without knowing the original category assignment. In other words, given a set $C = \{c_1, \ldots, c_k\}$ of categories, and a set Θ of n snippets pre-classified under C, the "ideal" term clustering algorithm is the one that, when asked to cluster Θ 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.

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

Figure 2: Labels generated by Armil for the query "madonna".

Following [16, page 110], the measure we use is *normalized mutual information*, 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'. 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* [16, 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

$$E(c'_l, C) = \sum_{k=1}^{|C|} -\frac{|c'_l \cap c_k|}{|c_k|} \log \frac{|c'_l \cap c_k|}{|c_k|}$$

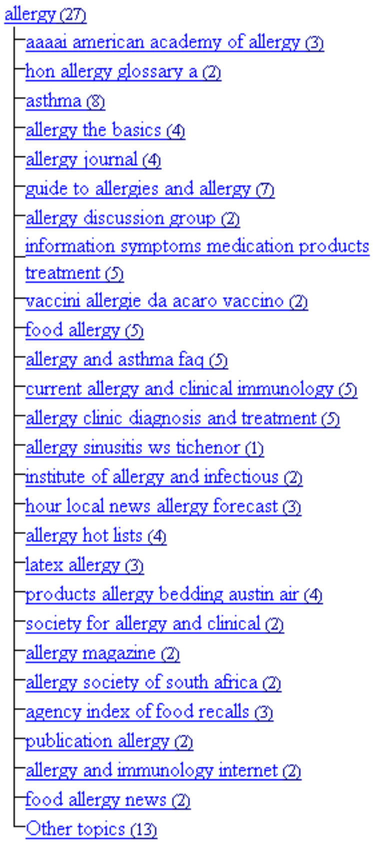

Figure 3: Labels generated by Armil for the query "allergy".

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.

6.3 Establishing the ground truth

The ephemeral nature of the Web is amplified by the fact that search engines have at best a partial view of the available pages relevant to a given query. Moreover search engines must produce a ranking of the retrieved relevant pages and display only the pages of highest relevance. Thus establishing a "ground truth" in a context of the full Web is problematic.Following [8], we have made a series of experiments using as input the snippets resulting from queries issued to the Open Directory Project (ODP – see Footnote 4). 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. For example, if a document belongs to class **Games/Puzzles/Anagrams** and another document belongs to class **Games/Puzzles/Crosswords**, we consider both of them to belong to class **Games/Puzzles**. 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 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.

6.4 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). 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 with 30 (this number is the default used in the user 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%)

Table 1: Results of the comparative evaluation.

The experiments indicate an substantial improvement of about 10% in terms of cluster quality of Armil(40) with respect to Vivisimo.[9]. This improvement is an important result since, as noted in 2005 in [5], "[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.

[9]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/armil.html

7. 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 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. We have selected 35 of the most popular queries submitted to Google in 2004 and 2005 (see `http://www.google.com/press/zeitgeist.html`); in the selection we have avoided queries whose meaning was too specific to a particular sub-culture in the Web space. The selected 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%

Summing the very positive and the mildly positive answers we can conclude that, in this experiment, 86.0% of the labels are syntactically well formed, 82.0% of the labels are predictive and 85.5% of the clusters are sufficiently well described by their label. 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. Entries in the top part give the percentage over all answers, and entries in the bottom part give percentage over rows.

	b-Yes	b-Sort-of	b-No
a-Yes	42.67%	12.81%	5.11%
a-Sort-of	5.74%	15.27%	4.41%
a-No	1.64%	3.78%	8.52%
a-Yes	70.41%	21.14%	8.43%
a-Sort-of	22.58%	60.04%	17.36%
a-No	11.76%	27.14%	61.08%

	c-Yes	c-Sort-of	c-No
b-Yes	33.52%	12.81%	3.72%
b-Sort-of	11.36%	16.85%	3.66%
b-No	2.14%	8.90%	7.00%
b-Yes	66.96%	25.59%	7.44%
b-Sort-of	35.64%	52.87%	11.48%
b-No	11.88%	49.30%	38.81%

	c-Yes	c-Sort-of	c-No
a-Yes	35.98%	18.93%	5.68%
a-Sort-of	8.64%	12.81%	3.97%
a-No	2.39%	6.81%	4.73%
a-Yes	59.37%	31.25%	9.37%
a-Sort-of	33.99%	50.37%	15.63%
a-No	17.19%	48.86%	33.93%

Table 2: Correlation tables of questions (a) and (b) (top), (a) and (c) (middle), (b) and (c) (bottom).

The data in Table 2 (top) 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 bottom 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.

7.1 Running times

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

8. 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 proposes 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, keywords are extracted, labels are generated) of the architecture a feedback loop by considering the extracted keywords/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. Finally, methods for detecting automatically the desired number of clusters will be studied.

9. REFERENCES

[1] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[2] D. Cheng, R. Kannan, S. Vempala, and G. Wang. On a recursive spectral algorithm for clustering from pairwise similarities. Technical Report MIT-LCS-TR-906, Massachusetts Institute of Technology, Cambridge, US, 2003.

[3] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, New York, US, 1991.

[4] D. R. Cutting, J. O. Pedersen, D. Karger, and J. W. Tukey. 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*, pages 318–329, Kobenhavn, DK, 1992.

[5] P. Ferragina and A. Gulli. 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*, pages 801–810, Chiba, JP, 2005.

[6] F. Geraci, M. Pellegrini, P. Pisati, and F. Sebastiani. A scalable algorithm for high-quality clustering of Web snippets. In *Proceedings of SAC-06, 21st ACM Symposium on Applied Computing*, Dijon, FR, 2006. Forthcoming.

[7] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2/3):293–306, 1985.

[8] T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating strategies for similarity search on the Web. In *Proceedings of WWW-02, 11th International Conference on the World Wide Web*, pages 432–442, Honolulu, US, 2002.

[9] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 76–84, Zürich, CH, 1996.

[10] D. S. Hochbaum and D. B. Shmoys. A best possible approximation algorithm for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[11] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. 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*, pages 658–665, New York, NY, 2004.

[12] D. J. Lawrie and W. B. Croft. Generating hierarchical summaries for Web searches. In *Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval*, pages 457–458, 2003.

[13] Y. Maarek, R. Fagin, I. Ben-Shaul, and D. Pelleg. Ephemeral document clustering for Web applications. Technical Report RJ 10186, IBM, San Jose, US, 2000.

[14] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[15] S. Osinski and D. Weiss. 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*, pages 369–377, Zakopane, PL, 2004.

[16] A. Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, University of Texas, Austin, US, 2002.

[17] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 46–54, Melbourne, AU, 1998.

[18] O. Zamir, O. Etzioni, O. Madani, and R. M. Karp. Fast and intuitive clustering of Web documents. In *Proceedings of KDD-97, 3rd International Conference on Knowledge Discovery and Data Mining*, pages 287–290, Newport Beach, US, 1997.

[19] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster Web search results. In *Proceedings of SIGIR-04, 27th ACM International Conference on Research and Development in Information Retrieval*, pages 210–217, Sheffield, UK, 2004.

[20] D. Zhang and Y. Dong. Semantic, hierarchical, online clustering of Web search results. In *Proceedings of APWEB-04, 6th Asia-Pacific Web Conference*, pages 69–78, Hangzhou, CN, 2004.

APPENDIX

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.