

A Scalable Algorithm for High-Quality Clustering of Web Snippets

Filippo Geraci^{1,2}, Marco Pellegrini¹, Paolo Pisati¹, Fabrizio Sebastiani³

(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,p.pisati}@iit.cnr.it fabrizio.sebastiani@unipd.it

ABSTRACT

We consider the problem of partitioning, in a highly accurate and highly efficient way, a set of n documents lying in a metric space into k non-overlapping clusters. We augment the well-known *furthest-point-first* algorithm for k -center clustering in metric spaces with a filtering scheme based on the triangular inequality. We apply this algorithm to Web snippet clustering, comparing it against strong baselines consisting of recent, fast variants of the classical k -means iterative algorithm. Our main conclusion is that our method attains solutions of better or comparable accuracy, and does this within a fraction of the time required by the baselines. Our algorithm is thus valuable when, as in Web snippet clustering, either the real-time nature of the task or the large amount of data make the poorly scalable, traditional clustering methods unsuitable.

Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Clustering

Keywords

Meta Search Engines, Web Snippets, Clustering, Metric Spaces

1. INTRODUCTION

Clustering is an important operation in the exploratory analysis of large data sets. Given a data set from some domain, it is sometimes desirable to group data items that are similar to each other under the same cluster, and data items that are dissimilar from each other under different clusters. As a consequence, the subdivision of the data set into clusters makes the latent structure of the data space explicit, and facilitates further human or automatic analysis.

Tools for clustering the results returned by Web search engines have recently become a focus of attention in the IR research community, also due to the success of commercial

services such as Vivisimo¹. Real-time clustering technology is a key ingredient of such systems, since the partition of the search results into clusters must be generated on-the-fly.

In this paper we propose an algorithm for real-time clustering, and discuss its accuracy and efficiency for Web snippet clustering [4, 15]. In the terminology of Web search engines, a *snippet* is the concise description of a Web page p that is returned to the user as a result of a query, and usually consists of at least (i) the title of p , (ii) its URL, and (iii) a piece of text extracted from p . The function of the snippet is to provide a highly succinct indication of why p was selected, and to allow the user to decide whether p is likely to be relevant to his/her information need *before* actually clicking on the associated URL. In the context of Web search engines, “Web snippet clustering” is a synonym of “clustering the results of a Web search”, since the snippet is all that is returned about p by the search engine, hence all that is known about p to the user or to a clustering service external to the search engine.

More abstractly, in this paper we consider the following type of applicative scenario: (a) n documents must be partitioned into a set of k non-overlapping clusters, for a predefined k ; (b) n and k are both large; (c) the documents admit a distance function that is a metric²; (d) the partition should be highly accurate and should be found quickly.

Our algorithm is a variation of the *furthest-point-first* (FPF) algorithm for k -center clustering [5] which attempts to minimize the radius of the widest cluster over all possible groupings into k clusters (k -clusterings) of a set of n documents in a metric space. We modify the FPF algorithm by using the triangular inequality to filter out redundant distance computations, thus significantly speeding up the computation with respect to the standard implementation.

In the context of on-line applications to IR, the algorithm of the Scatter/Gather system [6] is often used as a baseline. Scatter/Gather employs a variant of Lloyd's k -means algorithm with additional split/join operations on clusters intended to improve the accuracy of lower-accuracy clusters.

¹<http://vivisimo.com/>

²A distance function d defined on a domain \mathcal{D} is a *metric* when, for any data points $x, y, z \in \mathcal{D}$, it is true that (i) $d(x, y) \geq 0$ (*non-negativity*); (ii) $d(x, y) = d(y, x)$ (*symmetry*); (iii) $d(x, y) = 0 \Leftrightarrow x = y$ (*identity*); (iv) $d(x, y) \leq d(x, z) + d(z, y)$ (*triangular inequality*). Metrics include, as special cases, Euclidean distance and other distance functions commonly used in IR applications. See [16] for a detailed treatment.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'06 April 23-27, 2006, Dijon, France

Copyright 2006 ACM 1-59593-108-2/06/0004 ...\$5.00.

Scatter/Gather inherits the high computational costs of the standard k -means algorithm. For this reason, Phillips’ recently proposed fast variants of k -means [11], which also exploit filters based on the triangular inequality, are stronger baselines, and are the ones that we will check our algorithm against. In this paper we show that our method attains solutions of better or comparable accuracy, and does this within a fraction of the time required by the fast variants of k -means.

The rest of the paper is organized as follows. Previous results on Web snippet clustering are summarized in Section 2. Our algorithm is described in Section 3. We describe the experimental setting in Section 4, while in Section 5 we report on our experiments at Web snippet clustering. Section 6 concludes.

2. PREVIOUS WORK

Scatter/Gather [6] is historically one of the first systems based on a variant of k -means for clustering search results. However, it was not targeted to *Web* search, hence it did not use snippets.

Zamir and Etzioni [15] propose a Web snippet clustering method (Suffix Tree Clustering – STC) based on suffix arrays, and show that it outperforms algorithms such as k -means, single-pass k -means, backshot and fractionation, and group average agglomerative hierarchical clustering. Interestingly, the authors show that very similar results are attained when full documents are used instead of their snippets, thus confirming the fact that Web search results can be effectively clustered by looking at their snippets only.

Maarek et al. [7] characterize the challenges inherent in Web snippet clustering, and propose an algorithm based on complete-link hierarchical agglomerative clustering.

The Lingo [9], Shoc [17] and Eigencluster [1] systems all tackle Web snippet clustering by performing a singular value decomposition of the term-document incidence matrix; the problem is that SVD is extremely time-consuming, hence problematic when applied to a large number of snippets.

Ferragina and Gulli [4] propose a method for hierarchical clustering of Web snippets which produces a hierarchy of labelled clusters by constructing a sequence of labelled and weighted bipartite graphs representing the individual snippets on one side and a set of labelled clusters on the other side. In this work the emphasis is on the accuracy of the labels rather than on that of the clusters.

3. IMPROVING THE FPF ALGORITHM FOR K-CENTER

We now move to discussing our proposed algorithm.

Many clustering algorithms are based on an explicit attempt to minimize a given function of the distribution of the documents within clusters, a function whose minimization is believed to coincide with a higher accuracy of the resulting k -clustering³. Likewise, the *furthest-point-first* (FPF)

³These functions are sometimes used within experimental evaluations as *internal* measures of accuracy, i.e. as measures of the “well-formedness” of the resulting k -clustering. We will instead use an approach to evaluation based on *external* measures of accuracy, i.e. an approach in which the resulting k -clustering is checked against a “ground truth”, or “gold standard”, consisting of a group of n documents preclassified under k predefined classes. The accuracy of the clustering engine will be defined in terms of its ability at replicating this ground truth.

algorithm [3] is an attempt to solve the so-called *k -center clustering problem*, defined as follows:

The k -centers 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, \dots, C_k and determine their “centers” $\mu_1, \dots, \mu_k \in M$ so that $\max_j \max_{x \in C_j} D(x, \mu_j)$ (i.e. the radius of the widest cluster) is minimized.*

Given a set S of n points, the FPF algorithm builds a sequence $T_1 \subset \dots \subset T_k = T$ of k sets of “centers” (with $T_i = \{\mu_1, \dots, \mu_i\} \subset S$) in the following way. At iteration i :

1. for every point $p_j \in S \setminus T_{i-1}$, it 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 it picks $\mu_i = \arg \max_{p_j} D(p_j, \mu(p_j))$, i.e. the point which maximizes such minimal distance, and makes it a center, i.e. adds it to T_{i-1} , yielding T_i .

The final set of centers $T = \{\mu_1, \dots, \mu_k\}$ defines the resulting k -clustering, since each center μ_i 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} , it takes $O(n)$ time per iteration, so the total computational cost of the algorithm is $O(nk)$. We have thus improved this algorithm by exploiting the triangular inequality to filter out redundant distance computations.

The algorithm works as follows. Consider, in the FPF algorithm, any center $\mu_x \in T_i$ and its associated set of closest points $N(\mu_x) = \{p_j \in S \setminus T_i \mid \mu(p_j) = \mu_x\}$. Store $N(\mu_x)$ as a ranked list, in order of decreasing distance to μ_x . When a new center μ_y is selected, scan $N(\mu_x)$ 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. Note that this modified algorithm works in any metric space, hence in any vector space⁴.

This algorithm is advantageous in applications, such as Web snippet clustering, in which all data are loaded into main memory and clustering must be performed on-the-fly. In such scenarios scalability is an issue not because of the need to access slow secondary memories, but because of the high number of distance computations. In this context, algorithms (such as most variants of hierarchical agglomerative clustering) that compute all $O(n^2)$ pairwise distances between the data points are too expensive. Also methods that perform $O(nk)$ distance computations are not feasible here, except when k is a very small number (e.g. 2 or 3).

4. EXPERIMENTAL SETUP

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

4.1 The k -means algorithm and its variants

Our experiments have been run with two variants of our k -center algorithm and three recently proposed, fast variants of k -means.

The k -means algorithm can be seen as an iterative cluster quality booster. It takes as input a rough k -clustering (or, more precisely, k candidate centroids) and produces as output another k -clustering, hopefully of better quality. It has been shown [12] that by using the sum of squared Euclidean distances as objective function⁵, the procedure converges to a local minimum for the objective function within a finite number of iterations.

The main building blocks of k -means are (i) the generation of the initial k candidate centroids, (ii) the main iteration loop, and (iii) the termination condition. In the main iteration loop, given a set of k centroids, each input point is associated to its closest centroid, and the collection of points associated to a centroid is considered as a cluster. For each cluster, a new centroid that is a (weighted) linear combination of the points belonging to the cluster is recomputed, and a new iteration starts⁶. Several termination conditions are possible; e.g. the loop can be terminated after a predetermined number of iterations, or when the variation that the centroids have undergone in the last iteration is below a predetermined threshold.

Given the importance of this algorithm, there is a vast literature that discusses its shortcomings and possible improvements to the basic framework. In particular, one well-known such shortcoming is that some clusters may become empty during the computation. To overcome this problem, following [11] we adopt the “ISODATA” technique that, when a cluster becomes empty, splits one of the “largest” clusters so as to keep the number of clusters unchanged.

It is well-known, and our experiments confirm this, that the quality of the initialization (i.e. the choice of the initial k centroids) has a deep impact on the resulting accuracy. Several methods for initializing k -means are compared in [10]. As our baselines we have chosen the three such methods that are most amply cited in the literature while being at the same time relatively simple; we have ruled out more advanced and complex initializations since the possible boost in quality would be paid immediately in terms of computational cost, thus bringing about too slow an algorithm for our intended application.

4.2 Algorithms, baselines, and variants

The five algorithms we compare in our experiments are (i) two variants of our algorithm (KC and RS), and (ii) three recently proposed, fast variants of k -means (RC, RP, MQ) [11]. More in detail:

KC This is our k -Center algorithm with triangular inequality filtering, as described in Section 3.

⁵More precisely, this corresponds to partitioning S , at every iteration, into non-overlapping clusters C_1, \dots, C_k and determining their centroids $\mu_1, \dots, \mu_k \in V$ so that the sum of the squares of the inter-cluster point-to-center distances

$$\sum_j \sum_{x \in C_j} (D(x, \mu_j))^2$$

is minimized.

⁶Note that k -means is defined on vector spaces but not in general on metric spaces, since in metric spaces linear combinations of points are not points themselves.

RS This is the same as our KC algorithm, but applied to a Random Sample of the input points of size $n' = \sqrt{nk}$ (given that $k \leq n$, it is always true that $n' \leq n$). The remaining $(n - n')$ input points are subsequently associated to the closest center.

RP This is k -means (as described in Section 4.1) with an initialization based on Random Perturbation: for each dimension d_j of the space, the distribution of the projections on d_j of the data points is computed, along with its mean μ_j and its standard deviation σ_j ; the k initial centroids are obtained through k perturbations, driven by the μ_j 's and σ_j 's, of the centroid of all data points [10].

MQ This is MacQueen's [8] variant of k -means: the initial centroids are randomly chosen among the input points, and the remaining points are assigned one at a time to the nearest centroid, and each such assignment causes the immediate recomputation of the centroid involved.

RC This is again k -means where the initial centroids are Randomly Chosen among the input points and the remaining points are assigned to the closest centroid.

4.3 Quality measures

We denote with $\mathcal{L} = \{L_1, \dots, L_k\}$ the ground truth partition formed by a collection of *classes*, and with $\mathcal{C} = \{C_1, \dots, C_k\}$ the outcome of the clustering algorithm. As measures of accuracy we use F_1 , *entropy* and *purity*, three well-known measures that have been widely used in text clustering (see e.g. [1, 13] and the references therein). As evident from Table 1, the three measures tend to rank our algorithms in a fairly consistent way.

F_1 is the harmonic mean of *precision* (π) and *recall* (ρ). Given a cluster C_j and a class L_i , precision is defined as $\pi^{ij} = |L_i \cap C_j|/|C_j|$, i.e. the probability that an element of cluster C_j is actually an element of class L_i , while recall is defined as $\rho^{ij} = |L_i \cap C_j|/|L_i|$, i.e. the probability that an element of class L_i falls in cluster C_j ; F_1^{ij} is thus defined as $F_1^{ij} = 2(\pi^{ij} \cdot \rho^{ij})/(\pi^{ij} + \rho^{ij})$. On an entire k -clustering F_1 is defined as $F_1 = \frac{1}{n} \sum_i |L_i| \max_j F_1^{ij}$, where n is the total number of documents. The value of F_1 is in the range $[0, 1]$ and a higher value indicates better quality.

Entropy is a widely used measure in information theory. In our context, it measures our amount of uncertainty about the ground truth provided the available information is the computed k -clustering. Given a cluster C_j and a class L_i , we can define $E_j = \sum_i \pi^{ij} \log \pi^{ij}$ (where π^{ij} is defined as above), and $E = \frac{1}{n} \sum_j |C_j| \cdot E_j$. The value of E is in the range $[0, \log n]$ and a lower value indicates better quality.

While the entropy of a k -clustering is an average of the entropy of single clusters, the notion of *purity* is obtained using simply the maximum operator, i.e. $P_j = \max_i \pi_{ij}$ and $P = \frac{1}{n} \sum_j |C_j| \cdot P_j$. Purity is in the range $[0, 1]$, and a higher value indicates better quality.

4.4 Hw and sw platform

We have run our tests on a processor AMD Athlon (1Ghz Clock) with 750Mb RAM (however, the system could allocate only 512Mb RAM) and operating system FreeBSD 4.11-STABLE. The code was developed in Python V. 2.4.1.

5. EXPERIMENTS

We have made a series of experiments using as input the snippets resulting from queries to the Open Directory Project (ODP)⁷. The ODP is a searchable Web-based directory consisting of a collection of a few millions 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 our clustering efforts. In ODP, documents are organized according to a hierarchical ontology. For any snippet we have obtained 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/** 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. In ODP the textual quality of the snippets is quite variable; therefore, in order to filter out noise, for each query we first collect the top-ranked 200 snippets and then we discard those that are shorter than 40 characters and contain fewer than 3 words. Each retained snippet is turned into vectorial form by removing stop words, performing stemming, and weighting the terms by cosine-normalized $tf * idf$. The sine of the angle between two vectors is used by the clustering algorithms as the measure of their distance (note that the often used cosine is a measure of closeness, not of distance).

Since Web snippet clustering is used as an on-line support to Web browsing, real-time response is a critical parameter. A clustering phase that introduces a delay comparable to the time needed for just downloading the snippets, thus in fact doubling the user waiting time, is not acceptable for most users. For this reason, instead of using a fixed number of iterations as the termination condition of the k -means algorithm, we opt for a fixed time deadline (5 seconds, in our experiments) and we halt the clustering algorithms at the end of the first iteration that has run over the deadline.

In Table 1 we report clustering time and output quality of several variants of k -center and k -means on a sample of 12 queries subdivided, according to the method of [2], in three broad families: *ambiguous queries* (armstrong, jaguar, mandrake, java), *generic queries* (health, language, machine, music, clusters), and *specific queries* (mickey mouse, olympic games, steven spielberg). In Table 1, for each query group we indicate averages of the number n of snippets found, the number k of ODP classes to which they are associated, and the time τ (in seconds) used to build the vectorial representations. For each query group and each algorithm we indicate the time used for clustering and the quality of the outcome; we highlight in bold the two best values in each column⁸. The main conclusion of these experiments is that our two versions of FPF (KC and RS) are 5 to 10 times faster than the three fast versions of k -means (RP, MQ, and RC), and obtain a level of accuracy comparable or superior to these latter.

Note that in these experiments we ask our system to partition the snippets into exactly k clusters, where k is the number of labels under which the snippets are partitioned in the ground truth. These experiments thus do not address

⁷<http://www.dmoz.org/>

⁸For the sake of replicating the experiments all the search results have been cached and are available at <http://pspl.iit.cnr.it/~mcsoft/fpf/fpf.html>

Algorithm	Clustering Time	F_1	E	P
Query group = “ambiguous”				
$avg(n) = 165$		$avg(k) = 38.75$		$avg(\tau) = 0.695$
KC	0.923	0.383	0.967	0.590
RS	1.053	0.393	0.901	0.595
RP	45.310	0.270	1.135	0.456
MQ	6.775	0.332	0.981	0.555
RC	6.991	0.366	0.880	0.589
Query group = “generic”				
$avg(n) = 181.4$		$avg(k) = 42$		$avg(\tau) = 0.889$
KC	1.208	0.379	1.005	0.541
RS	1.339	0.369	0.994	0.536
RP	56.912	0.279	1.184	0.431
MQ	7.164	0.364	0.991	0.532
RC	6.597	0.345	1.035	0.514
Query group = “specific”				
$avg(n) = 107.3$		$avg(k) = 26$		$avg(\tau) = 0.509$
KC	0.536	0.510	0.834	0.642
RS	0.606	0.530	0.758	0.647
RP	17.239	0.372	1.004	0.508
MQ	5.796	0.413	0.837	0.591
RC	5.092	0.427	0.845	0.589
Query group = “ambiguous \cup generic \cup specific”				
$avg(n) = 157.4$		$avg(k) = 36.9$		$avg(\tau) = 0.729$
KC	0.945	0.413	0.949	0.582
RS	1.060	0.417	0.904	0.583
RP	43.126	0.299	1.122	0.458
MQ	6.692	0.365	0.949	0.554
RC	6.352	0.372	0.935	0.557

Table 1: Results of Web snippet clustering for a sample of ODP snippets.

the problem of how the value of k should be chosen in an operational environment. Choosing this value can only be done empirically, since the “optimal” value of k is dependent on the user’s preferences as to the level of granularity at which information should be organized, which is highly subjective.

5.1 Redefining the distance function

In subsequent experiments we have tested the effect of using distance functions other than sine. We have obtained interesting results by using a slight modification of the standard Jaccard Coefficient, which we call *Weighted Jaccard Coefficient* (WJC) [14]. The WJC takes advantage of the intrinsic structure of the ODP snippets, by weighting different parts of the snippet (title, body, URL) differently; more precisely, we assign weight 3 to the title, weight 1 to the body, and ignore the URL (in experiments we have run, the text of the URL seems proves almost useless for increasing cluster quality). So, our WJC metric is defined as follow:

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

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 the snippet s .

It is easy to note that, by using the WJC, the preprocessing phase is reduced to stop word removal and stemming,

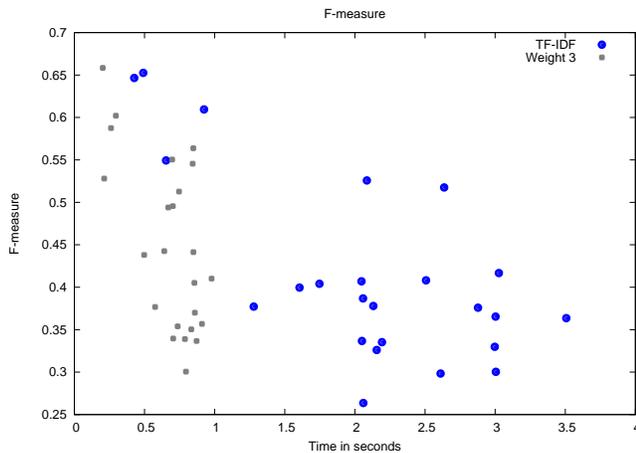


Figure 1: F_1 values obtained, for the same queries, using sine (large black dots) or WJC (small grey dots) as distance functions, running the two variants of our algorithm (KC and RS) on ODP data; time (in seconds) includes the construction of the vector representation and clustering.

and does away with the need to compute cosine-normalized $tf * idf$ weights; as a consequence, it is much faster. We do not report detailed results of these experiments for reasons of space, and we only summarize them qualitatively in Figure 1, which clearly shows how WJC has an accuracy comparable to sine while being far more efficient.

6. CONCLUSIONS

The k -means algorithm is a renowned clustering method used in a wide range of applications. In contrast, the furthest-point-first method for k -center clustering had been considered so far only of theoretical interest. In this paper we propose a much faster variant of FPF based on a filtering step that exploits the triangular inequality, and show its suitability for Web snippet clustering, a task at which our algorithm performs with accuracy comparable or often better than recently proposed, fast versions of k -means, while being far more efficient to run. We show that even higher efficiency can be obtained by using metrics that exploit the internal structure of the snippets.

Our algorithm dispenses with the need to compute centroids, and could thus easily deal with situations where the notion of a centroid is not a natural or well-defined one, or is expensive to compute and update.

7. REFERENCES

- [1] 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.
- [2] E. Di Giacomo, W. Didimo, L. Grilli, and G. Liotta. A topology-driven approach to the design of Web meta-search clustering engines. In *Proceedings of SOFSEM-05, 31st Annual Conference on Current Trends in Theory and Practice of Informatics*, Liptovský Ján, SK, 2005.
- [3] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proceedings of STOC-88, 20th ACM Symposium on Theory of Computing*, pages 434–444, Chicago, US, 1988.

- [4] 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, International Conference on the World Wide Web*, pages 801–810, 2005.
- [5] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2/3):293–306, 1985.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] J. M. Peña, J. A. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the k -means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.
- [11] S. J. Phillips. Acceleration of k -means and related clustering algorithms. In *Proceedings of ALENEX-02, 4th International Workshop on Algorithm Engineering and Experiments*, pages 166–177, San Francisco, US, 2002.
- [12] S. Selim and M. Ismail. K -means type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):81–87, 1984.
- [13] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *Proceedings of the ACM KDD-00 Workshop on Text Mining*, Boston, US, 2000.
- [14] A. Strehl, J. Ghosh, and R. J. Mooney. Impact of similarity measures on Web-page clustering. In *Proceedings of the AAAI Workshop on AI for Web Search*, pages 58–64, Austin, US, 2000.
- [15] 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.
- [16] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity search: The metric space approach*. Springer-Verlag, Heidelberg, DE, 2006. Forthcoming.
- [17] 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.