Document clustering using synthetic cluster prototypes

Argyris Kalogeratos[a], Aristidis Likas[a,*]

[a]*Department of Computer Science, University of Ioannina, 45110, Ioannina, Greece*

Abstract

The use of centroids as prototypes for clustering text documents with the k-means family of methods is not always the best choice for representing text clusters due to the high dimensionality, sparsity, and low quality of text data. Especially for the cases where we seek clusters with small number of objects, the use of centroids may lead to poor solutions near the bad initial conditions. To overcome this problem, we propose the idea of *synthetic cluster prototype* that is computed by first selecting a subset of cluster objects (instances), then computing the representative of these objects and finally selecting important features. In this spirit, we introduce the *MedoidKNN* synthetic prototype that favors the representation of the *dominant class* in a cluster. These synthetic cluster prototypes are incorporated into the generic spherical k-means procedure leading to a robust clustering method called *k-synthetic prototypes* (*k-sp*). Comparative experimental evaluation demonstrates the robustness of the approach especially for small datasets and clusters overlapping in many dimensions and its superior performance against traditional and subspace clustering methods.

Keywords: clustering methods, document clustering, text mining, term selection, subspace clustering

1. Introduction

Document clustering is an unsupervised learning approach for automatically segregating similar documents of a corpus into the same group, called *cluster*, and dissimilar documents to different groups. Formally, a corpus of N unlabeled documents is given and a solution $C = \{c_j \colon j=1, \ldots, k\}$ is searched that partitions the document into k disjoint clusters.

Even small text datasets carry large vocabularies and certain undesirable effects arise due to the *curse of dimensionality* [4]. The *high dimensional and sparse* (*HDS*) feature space in combination with language phenomena such as *polysemy*, *homosemy* and *metaphors*, constitute an adverse setting for clustering methods. When a labeled training dataset is provided, several statistical options are available for feature selection [5, 6], even in case of multilabeled data objects [53]. On the other hand, it is more complicated to select features in an unsupervised setting and it is usually achieved using heuristics [49, 50, 51, 52]. Methods such as *Latent Semantic Indexing* (*LSI*) [47], or *Latent Dirichlet Allocation* [48] (*LDA*), may discover the term correlations but they map the data into a feature space of much lower dimensionality.

*Corresponding author. Tel.: +30 26510 08810; Fax: +30 26510 08882.
Email addresses: akaloger@cs.uoi.gr (Argyris Kalogeratos), arly@cs.uoi.gr (Aristidis Likas)

Clustering algorithms are separated in two major categories *hierarchical* and *partitional* (for a survey see [7]). The former produce a hierarchy of solutions, either by merging, or by dividing clusters. Partitional approaches seek to discover a set of unique cluster representations that describe properly the underlying data classes of a dataset. An *objective function* $\Phi(C)$ evaluates the quality of a data partition by quantifying how good the derived representations are for the corresponding clusters. These methods start from a set of k cluster representations which are improved iteratively in a way that $\Phi(C)$ is optimized. *Probabilistic* methods use probabilistic *cluster models* (or *topic models*) [9, 8], while *non-probabilistic* methods utilize representatives in the feature space, called *prototypes*, that are used to represent the objects of a cluster. Typical prototypes are the *arithmetic mean* called *centroid*, and the *medoid* that is a real object which is representative for the cluster it belongs.

A popular partitional method is *k-means* [10] that represents each cluster with its centroid. Many heuristic variations of k-means have been proposed and applied for text collections [11, 12, 13, 14]. *Spherical k-means* (*spk-means*) [22] is a modified version that utilizes the *cosine similarity* measure to cluster the data by partitioning the unit hypersphere into k hypercones, one for each cluster. This method is fast and gives better clusters than traditional k-means [12].

Special algorithms have also been developed to deal with HDS feature spaces. The clustering methodology aiming at finding clusters in subspaces of data instead of the entire feature space is referred to as *subspace clustering* and its key characteristic is the simultaneous determination of the object membership to clusters and the subspace of each cluster. Surveys on subspace clustering in high dimensional spaces can be found in [29, 46]. Recently, much attention has been received by methods that aim to identify the cluster structure in on-line high-dimensional data streams [54, 55].

This work puts forth the idea that, although the centroids are the optimal cluster prototypes with respect to certain objective functions (e.g. based on cosine similarity), their optimality could also become a drawback in HDS feature spaces and in cases of low data quality (e.g. outliers, noise). Especially, as the number of data objects becomes smaller compared to the complexity of a clustering problem (i.e. number of clusters, dimensionality), the centroids become less appropriate cluster representatives. Text documents constitute a typical example of data where such an adverse setting is met.

In this paper we present the *synthetic prototype*, a novel type of cluster representative that, given the object assignment to clusters, is computed in two steps: a) a *reference prototype* is constructed for the cluster and then b) *feature selection* is applied on it. We propose the so-called *MedoidKNN* reference prototype which is based on a subset of K objects of a cluster that are close to its medoid. This synthetic prototype favors the representation of the objects of the *dominant class* in a cluster, i.e. the class to which the majority of the cluster objects belong. Finally, we modify the generic spk-means iterative procedure by incorporating synthetic prototypes. This leads to a novel, effective and quite simple clustering method called *k-synthetic prototypes* (*k-sp*). We conducted an extensive evaluation of the k-sp method examining several options for the synthetic prototypes and comparing it to several traditional clustering methods such as spherical k-means, agglomerative, spectral clustering and two soft subspace clustering methods.

The rest of this paper is organized as follows: in Section 2, a background discussion for the document clustering problem is provided. In Section 3, we present our novel synthetic prototype cluster representation and the k-synthetic prototypes clustering method. In Section 4, comparative experimental results are reported and discussed, and finally in Section 5, we present concluding remarks and future research directions.

2. Background

2.1. Document Representation

A preprocessing step on the corpus decides which terms are meaningful to be included in the *corpus vocabulary* V, a set of $|V|$ unique features. Despite the fact that it is reasonable to seek for complex representations for text data, such as graphs [1, 2, 3], the typical approach is to represent each input document as a *bag-of-words* [18] feature vector $d_i \in \mathbb{R}^{|V|}$, i=1, . . . , N, whose elements are weight values denoting the significance of each vocabulary term for the document. Typically, the weights are set using the $tf{\times}idf$ scheme and document vectors are normalized to unit length with respect to Euclidean L_2-norm. Hence, the i-th document is modeled as:

$$d_i = \frac{\left(tf_{i1} \log \frac{N}{N^{(1)}}, \dots, tf_{i|V|} \log \frac{N}{N^{(|V|)}}\right)}{\left[\sum_{j=1}^{|V|} tf_{ij}^2 \log^2 \frac{N}{N^{(j)}}\right]^{-1/2}}, \tag{1}$$

where tf_{ij} is the frequency of j-th term in the i-th document and $N^{(j)}$ the number of documents that contain j-th term. The proximity between two documents is computed using *cosine similarity*, considered to be an effective measure for text clustering [19, 20], that computes the cosine of the angle between the two document vectors:

$$sim^{(cos)}(d_i, d_j) = \cos(\theta(d_i, d_j)) = \frac{d_i^{\top}}{\|d_i\|_2} \cdot \frac{d_j}{\|d_j\|_2}. \tag{2}$$

2.2. Properties of the Representation Space of Documents

The properties of the vector space in which text documents are represented are closely related to the nature of human language. Even small text datasets carry very large vocabularies and, apart from the known negative effects of the curse of dimensionality, the learning algorithms have to deal with the existence of high sparsity. It has been observed that a document may have less than 1% of the global corpus vocabulary [21] (non-zero vector dimensions) since there are terms in the corpus vocabulary that do not appear in a given document although they are relevant to its content. This is due to the fact that each document usually is a specific *semantically narrow instance* of a much more general document class.

Moreover, two authors may express exactly the same ideas using different terms, for instance using homosemous terms, metaphors or complex expressions, which introduces additional sparsity and dimensionality. Some noise is also present in text datasets, such as confusing polysemous terms or even irrelevant features. Under this situation, documents of the same class present average pairwise similarity comparable in magnitude to the similarity between documents from different classes [27, 28]. For instance, let d_x, d_y, and d_z three documents of the same class; it is possible for d_x to share a set of terms with d_y and a different set of terms with d_z whereas, at the same time, d_y and d_z may exhibit no vocabulary intersection although this would be expected to hold mostly for pairs belonging to different classes. In this context, certain qualitative issues arise regarding the direct determination of a large number of nearest neighbors to an object [27, 28]. For example, if an object has non-zero similarity with K objects in the dataset (or a cluster), then the direct determination of its nearest K'>K objects would unavoidably make guesses.

Document clustering differentiates from the high dimensional data clustering problems that seek for a single *global subspace of features* in which there are observable clusters. Different document clusters are formed in generally different subspaces. Small text datasets should be treated as document clustering cases of special interest. According to *Heap's power law* [40], the increase of the corpus vocabulary is sublinear to the number of included documents. In other words, the *relative dimensionality* of the feature space, empirically defined as $\log(|V|/N)$, is expected to be much larger for small datasets than for larger ones. This large vocabulary diversity even between documents of the same class, justifies for the difficulty of clustering small document datasets.

2.3. Clustering using k-means Family of Methods

The k-means procedure is a generic clustering approach that assumes a prototype to represent each cluster and an objective function $\Phi(C)$ that evaluates the quality of a partition C. In order to solve a problem with k clusters the k prototypes are initialized usually by randomly selecting k objects as cluster centroids (*Forgy's approach*) and then the algorithm iterates to optimize the objective function:

1. *Reassignment step*: each object is assigned to the cluster whose prototype is nearest to the object.
2. *Prototype batch update step*: given the assignment of objects to clusters, each cluster prototype is updated in a way that optimizes the objective function.

k-means minimizes the sum of squared Euclidean distances between the objects of the clusters and the centroid prototypes Eq. 3, where the centroids are computed as the arithmetic mean $\mu_j = (1/n_j)\sum_{d_i \in c_j} d_i$ of the n_j objects of that cluster:

$$\Phi_{SSE}(C) = \sum_{j=1}^{k} \sum_{d_i \in c_j} \left\|\mu_j - d_i\right\|_2^2. \tag{3}$$

It converges to a local minimum of $\Phi_{SSE}(C)$ and the quality of the solution depends strongly on the initial conditions. Its time complexity is $O(tN\,|V|)$, where t is the number of iterations until convergence. The employed cluster prototypes constitute a choice that also affects the solution quality. *k-medoids* is a robust method that represents a cluster with the *medoid* object defined as the object that has the maximum average similarity to the objects of its cluster:

$$m_j = \arg\max_{d_i \in c_j} \left\{ \frac{1}{n_j} \sum_{d_q \in c_j} sim(d_i,\, d_q) \right\}. \tag{4}$$

In this case, Eq. 3 is computed with respect to the medoid prototypes and in Euclidean space there is the disadvantage of complexity $O(n_j^2)$ to determine a cluster medoid.

Spherical k-means (*spk-means*) is a variant of k-means that utilizes the cosine similarity for the data vectors normalized with respect to L_2-norm. The maximized objective function is the *clustering cohesion*. The optimal prototype for a cluster is its *normalized centroid* $u_j{=}s_j/\left\|s_j\right\|_2$, where $s_j{=}\sum_{d_i \in c_j} d_i$, and the overall clustering cohesion of a partition C is given by:

$$\Phi_{coh}(C) = \sum_{j=1}^{k} \sum_{d_i \in c_j} u_j^{\top} \cdot d_i = \sum_{j=1}^{k} u_j^{\top} s_j = \sum_{j=1}^{k} \frac{s_j^{\top} \cdot s_j}{\left\|s_j\right\|_2} = \sum_{j=1}^{k} \left\|s_j\right\|_2 . \tag{5}$$

A lot of research effort has been focused in the careful initialization of this family of algorithms, due to its importance for the final clustering quality [35, 43, 44, 45]. Among the typical object-based seeding techniques is the deterministic *Kaufman heuristic* (or *k-farthest heuristic*) [36] that tries to spread the initial centroids away from each other. It selects the most centrally located object as the first centroid and each additional centroid is determined to be the object farthest from the objects-centroids already selected. *k-means*++ [44], on the other hand, introduces stochasticity: it starts with the uniform random selection of one object as the first centroid, then each next centroid is determined using a weighted probability distribution. Specifically, the probability for a candidate object to be selected as a new centroid is proportional to the squared distance between the object and its nearest centroid previously selected. In [44] it is shown that this initialization guarantees an $O(\log k)$ approximation to the optimal k-partition. However, all the above initialization methods select objects as seeds and this may not be efficient in the text feature space, since a document usually contains a very small percentage of the vocabulary terms. This is further analyzed and experimentally illustrated in this work.

Clustering refinement is the *post-processing* procedure aiming to improve the clusters produced by a clustering method (note that in literature the term ‘refinement’ is also used to describe the iterative optimization of an objective function). The refinement algorithm may be a specialized algorithm that proceeds to small changes in the clusters, such as single object reassignment [41] and swapping the cluster memberships for pairs of objects [42]. It is also a practical choice to refine the produced clusters using a clustering method of different characteristics to the initial one (e.g. k-means starting with the clusters produced by a hierarchical clustering, agglomerative or bisecting k-means). An alternative approach is the *hybridized centroid-medoid heuristic* [14] that applies a small number of k-means iterations and tries to replace a centroid with a medoid belonging in a set of candidate medoids precomputed off-line.

2.4. Text Document Subspace Clustering

The different topics are usually described by generally different subsets of terms which, in combination with the high sparsity of the feature space, lead to the hypothesis that the underlying cluster structure may be better to be sought in subspaces of the original feature space. The feature selection that is applied in the preprocessing phase actually computes a single *global subspace* where data clustering is performed. A more *fuzzy feature selection* would assign a global weight to each dimension. *Subspace clustering* can be thought as to be an extension to feature selection in the sense that it determines a subspace explicitly for each cluster during clustering.

In brief and according to [29], the main categorization of subspace clustering methods is based on the relation between the axes of the subspaces they seek and the axes of the original feature space. One approach, called *generalized subspace clustering*, is to seek for arbitrarily oriented subspaces. Their major difficulty is to deal with the infinite search space of the candidate subspaces. A second and more widely used approach is constrained to seek for subspaces with axes parallel to the original. The *projected subspace clustering* lets no intersection between the dimensions that span the different subspaces and hence, 2^d-1 possible subspaces must be examined. The subcategory that lets different axis-parallel subspaces to have dimensions in common is called *soft projected clustering* and usually different feature weights in [0,1] are assigned for each cluster. The latter subcategory can be further split based on the searching approach adopted regarding the feature set a method starts to work with. *Top-down* approaches start with the full set of features and iteratively try to determine narrow subspaces for each cluster. On the other hand, *bottom-up* approaches start from single dimension subspaces and use a strategy similar to mining *frequent itemset* to increase their dimensionality.

Apparently, there are important methodological differences in the literature of subspace clustering, but a thorough analysis is beyond the scope of this work. In the rest of this section we will discuss the recent research on top-down soft projected subspace clustering methods that develop *feature weighting mechanisms* and incorporate them to k-means, and have also been tested on the document clustering problem.

An abstract framework is presented in [30] that, using multiple feature vectors to represent each data object, is able to integrate the heterogeneous feature spaces in the k-means algorithm. A *convex-k-means* algorithm is proposed that is based on a convex objective function constructed as a weighted combination of the distortions of each individual feature subspace. The algorithm simultaneously minimizes the average within-cluster dispersion and maximizes the average between-cluster dispersion along all of the feature spaces. A method that received much attention is *Clustering on Subsets of Attributes* (*COSA*) [37]. It is an iterative algorithm that considers a feature weight vector to each data point, initially containing equal weights for all features. Larger weights are assigned to features that present small dispersion in a neighborhood around the reference object, which means that are more important. The next step is to use these weights to compute some other weights corresponding to each pair of objects that, in turn, update the distances for the computation of the nearest neighbors. The algorithm stops iterating when weight vectors corresponding to objects become stable. COSA outputs a pairwise distance matrix based on a weighted inverse exponential distance and any distance-based clustering method can produce the final clusters. The algorithm requires the user specification of the size of neighborhood to consider, a second parameter that controls the fade of the exponential feature weighting, while the major issue is that all the $N \times |V|$ parameters should be estimated during the process.

Some other algorithms were then developed that consider one feature weighting vector for each cluster. *Feature Weighting K-means* (*fwk-means*) [39] aims to minimize the following objective function:

$$\Phi_{fwkm}(C) = \sum_{j=1}^{k} \sum_{d_i \in c_j} \sum_{l=1}^{|V|} w_{jl}^{h} \left[(\mu_{jl} - d_{il})^2 + \sigma \right], \tag{6}$$

subject to

$$\sum_{l=1}^{|V|} w_{jl} = 1, \;\; 0 \leq w_{jl} \leq 1, \;\; j = 1, \ldots, k, \tag{7}$$

where μ_j is the L_1-normalized centroid of the j-th cluster and $h>1$ a parameter that must be set in advance. The term $w_{jl}^{h}(\mu_{jl} - d_{il})^2$ computes the distance between the centroid μ_{ji} and a document d_i on the specific l-th feature dimension. Initially, the weights are set to $1/|V|$ and the k centroids are set in a random fashion. The optimization is then performed by iterating the following steps until convergence:

1. Object assignment to their nearest cluster using the computed centroids and feature weights.
2. Computation of the cluster centroids using the computed feature weights.
3. Computation of the feature weights for each cluster using the computed cluster centroids.

Given the cluster centroids and the k feature weighting vectors of the previous iteration, the optimal weight of the l-th feature for cluster c_j is computed by:

$$w_{jl} = \frac{1}{\sum_{t=1}^{|V|} \left[\frac{\sum_{d_i \in c_j} w_{jl} \left[(\mu_{jl} - d_{il})^2 + \sigma \right]}{\sum_{d_i \in c_j} w_{jl} \left[(\mu_{jt} - d_{it})^2 + \sigma \right]} \right]^{1/(h-1)}}, \tag{8}$$

where σ is the average dispersion of the vocabulary measured off-line in a sample of N_{sample} data objects. fwk-means adds this value because a feature weight is not computable if its dispersion in a cluster is zero. If we let mfv_l to be the mean feature value of the l-th feature in the data sample then σ is given by:

$$\sigma = \frac{\sum_{d_i \in c_{sample}} \sum_{l=1}^{|V|} (d_{il} - mfv_l)^2}{N_{sample} |V|}. \tag{9}$$

Locally Adaptive Clustering (*LAC*) algorithm presented in [33] is quite similar to the *Entropy Weighting k-means* (*ewk-means*) [31]. Both share some ideas with COSA, whereas the feature weighting vectors are assigned to clusters instead of objects. Moreover, their search strategy is more alike to fwk-means. A modified objective function is utilized, which is to add the *weight entropy* $e_j = \sum_{l=1}^{|V|} w_{jl} \log w_{jl}$ corresponding to each cluster in order penalize the identification of clusters in subspaces spanned by very few features. The objective function of ewk-means is:

$$\Phi_{ewkm}(C) = \sum_{j=1}^{k} \left[\sum_{d_i \in c_j} \sum_{l=1}^{|V|} w_{jl} (\mu_{jl} - d_{il})^2 + \gamma \, e_j \right], \ \gamma \geq 0 \tag{10}$$

subject to Eq. 7 and the value of γ controls the focus of the objective function on the feature weight entropy. The iterative optimization is identical to that of fwk-means and differ only on the weight computation:

$$w_{jl} = \frac{\exp(-disp_{jl}/\gamma)}{\sum_{t=1}^{|V|} \exp(-disp_{jt}/\gamma)}, \tag{11}$$

where

$$disp_{jl} = \sum_{d_i \in c_j} (\mu_{jl} - d_{il})^2. \tag{12}$$

COSA and fwk-means require the tuning of the value of the parameter controlling the size of the subspaces that are sought (the value of γ in ewk-means). LAC introduces an ensemble approach that combines multiple clustering solutions discovered by LAC using different γ values, which produces a superior result than that of the participating solutions. The feature weights of these methods enable the modeling of more complex cluster shapes than the spherical of traditional k-means. However, the parameters that need to be estimated are doubled compared to k-means: $2k \times |V|$ for the feature weights and the cluster centroids. This parameter increase unavoidably causes a large increase to the number of local minima of the search space. Recently, an adaptive weight-adjusting principle was adopted in [34], which at each step adds a Δw_{jl} to each w_{jl} weight computed based on the extend of contribution of the weight to the clustering quality. Finally, in [38] an algorithm similar to LAC and fwk-means is presented, also allowing the incorporation of constraints derived from a labeled data subset.

3. The k-synthetic prototypes clustering method

3.1. Clustering Using Centroids and Medoids

From an optimization point of view, the normalized centroid is the prototype that maximizes cluster's cohesion Eq. 5. However, this optimality may also become a drawback in such a feature space, especially at early clustering iterations where clusters have low homogeneousity due to random initialization. More specifically, there exist two undesirable phenomena concerning the use of centroids. At a data object level, the *self-similarity* phenomenon implies that the similarity of a document with itself becomes the dominant factor for deciding about its nearest cluster [12, 22]. This is explained by observing the similarity between a normalized centroid u_j and a document d in the respective cluster c_j:

$$u_j^\top \cdot d = \frac{1}{\left\|\sum_{d_i \in c_j} d_i\right\|_2} \Big(d^\top \cdot d + \sum_{\substack{d_i \in c_j \\ d_i \neq d}} d^\top \cdot d_i \Big). \tag{13}$$

Due to sparsity, the term $d^\top \cdot d_i{=}1$ can be large in magnitude compared to the sum of similarities between d and the documents of c_j, or the documents of other clusters. In an extreme case, a document $d \in c_j$ which has non-zero similarity only with documents from clusters other than c_j, may still determine c_j as its nearest cluster, since due to the self-similarity term it may hold that:

$$\frac{d^\top \cdot d}{\left\|\sum_{d_i \in c_j} d_i\right\|_2} > \frac{\sum_{d_i \in c_l} d^\top \cdot d_i}{\left\|\sum_{d_i \in c_l} d_i\right\|_2}, \quad l = 1, \ldots, k, \ l \neq j. \tag{14}$$

Hence, d may remain in an inappropriate cluster. This phenomenon appears more intense in cases where there is a small number of objects per cluster in combination with high sparsity.

The second phenomenon is the *feature over-aggregation* that occurs when computing a centroid for an impure cluster. Supposing that there is a feature subset f_j^+ strongly related to each document class j, and a usually much larger subset f_j^- containing the remaining $|V|-|f_j^+|$ terms, then the learning process aims to find a cluster prototype, i.e. a weight vector in $\mathbb{R}^{|V|}$, being discriminative for that class. This means that for each cluster the clustering algorithm should try to determine the $|f_j^+|$ representative features for its dominant class and to estimate their relative weight distribution in the possible presence of $|f_j^-|$ irrelevant features that should be assigned with very low weights. The effectiveness of such an algorithm may be greatly affected by the level of the relative significance of the features of f_j^+ to that of f_j^- in a cluster at a particular iteration, which can be formally expressed by the following ratio:

$$\delta_j = \frac{\sum_{i \in f_j^+} u_{ji}}{\sum_{i=1}^{|V|} u_{ji}}. \tag{15}$$

Feature over-aggregation appears at the initial iterations where very low δ-ratio values are observed in the clusters of poor quality. This prevents the prototypes from becoming more class discriminative, since the non-informative features also affect the object assignment to clusters and hence the problem is retained.

Both self-similarity and feature over-aggregation constrain the local search flexibility of the k-means procedure and lead to poor solutions strongly dependent on initial conditions, where often documents from two or more classes are assigned to the same cluster.

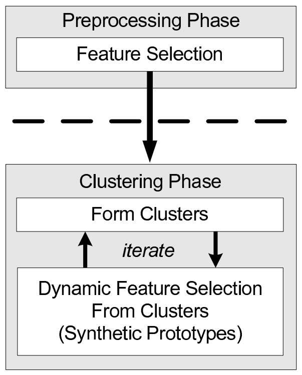


Figure 1: The k-sp framework using synthetic prototypes.

In what concerns the use of medoid as cluster prototype, it does not present the self-similarity and feature over-aggregation effect. However, as mentioned in Section 2.2, since each document is a specific semantically narrow instance of the more general topics of its class, it contains a very small fraction of vocabulary terms. Thus it is unlike for a *single document* to be a good cluster representative.

3.2. Synthetic Cluster Prototypes

Traditionally, feature selection (in our case term selection) takes place in the preprocessing phase. However, we adopt a dynamic selection scheme implemented in the form of *synthetic cluster prototypes*, which are computed by first selecting objects and then features from each cluster (Fig. 1). As clustering proceeds we exploit the information progressively produced in the formed clusters to retain the important features for each cluster. To compute a synthetic prototype we must define:

i. a *reference prototype*, an initial representative of the cluster constructed by a *subset* of its objects, and

ii. *feature selection on prototypes* in order to select features from the reference cluster prototype.

The L_2-normalized cluster representative derived by filtering the features of a reference prototype is a *synthetic prototype*. These prototypes are generic, in the sense that they can be constructed by considering any reference prototype or feature selection scheme. Omitting the feature selection step is also a viable option, thus a reference prototype is also a synthetic prototype. In this case feature selection is achieved implicitly since the reference prototype is computed using a subset of the cluster objects and it may not contain all the vocabulary terms.

The proposed clustering algorithm is called *k-synthetic prototypes* (*k-sp*) and incorporates the synthetic prototypes into the spk-means procedure. Note that spk-means is a *special case* of k-sp where the cluster centroids are used as reference prototypes and no feature selection is applied. By using synthetic prototypes the k-sp procedure aims to discover dynamically a different feature subspace in which each document class can be better separated but, at the same time as we explain, to mitigate the negative effects of the self-similarity and the feature over-aggregation phenomena. The explicit feature selection scheme we have considered is the simple thresholding on the feature weights of a reference prototype to keep the P most significant features of a cluster (see Section 3.3). Contrary to the typical preprocessing feature selection techniques, k-sp does not affect the original data objects and hence, does not constrain future iterations with previous cluster representations. In a later phase, one could consider much more detail (i.e. more objects and features) from the clusters to fine-tune the solution.

A straightforward option for reference prototype is the Centroid$^{(r)}$[1] of a cluster. The assumption behind this choice is that many of the representative features for the dominant class in a cluster would have high weights in the respective centroid. Thus, the feature selection on it would keep the highly descriptive features for this class. Obviously, this is not true for a cluster containing documents of more than one class none of which is clearly dominant (Fig. 2b).

We propose *MedoidKNN*$^{(r)}$, an approach to construct the reference prototypes by computing the centroid of a *subset* Y of documents assigned to a cluster that are descriptive of its dominant class. The set Y can be formed by selecting the K documents of the cluster being the *nearest neighbors to the medoid* of that cluster, including the medoid itself. As explained in Section 2.2, it would not be very efficient to directly determine a large number of nearest neighbors of a medoid using its pairwise similarities, since the medoid document may contain only a part of the features present in the cluster. This issue is further discussed on real world examples in Section 4.4.1. Therefore, we propose an incremental procedure to form the set Y that avoids computing a large number of nearest neighbors directly from the medoid object. Let λ be the number of desired steps and β_i, i=$1, \dots, \lambda$ a sequence of values such that $0<\beta_i<\beta_{i+1}<...<\beta_\lambda=1$. Starting with the medoid $Y_0=\{m\}$, each subset Y_i (for $i\geq1$), is formed by the $\lceil\beta_i K\rceil$ documents nearest to the centroid of subset Y_{i-1}. For a two-step example with $\beta_1=0.2$, and $\beta_2=1$, we first determine the medoid for a cluster c_j and then: i) we determine the $\lceil 0.2K\rceil$ objects in c_j nearest to the medoid and compute their centroid rp_1, ii) we locate the K objects in c_j nearest to rp_1 and compute rp_2 which is the final MedoidKNN$^{(r)}$. Notice that for K=n_j, the rp coincides with the centroid of cluster c_j, while for K=1 it is the cluster medoid. Typically, up to three steps (λ=3) are sufficient to determine a proper final set Y_λ.

One could argue that the set Y should contain the nearest documents to the cluster centroid and not to the medoid. As a matter of fact, the medoid is close to centroid in a homogeneous cluster and the nearest objects to medoid may also be the nearest objects to the centroid. However, if there are objects of more than one class in a cluster, the medoid-based construction of Y is more probable to lead to a sharp preference for one of the overlapping classes (see Fig. 2). This argument is strengthened by a usually holding property called *intracluster rNN-consistency*: any data object in a cluster and its r nearest objects in the same cluster will belong to the same class with high probability. We should remark that intracluster rNN-consistency is expected to be higher than the rNN-consistency of the whole dataset that can be similarly defined [13].

A proper synthetic prototype should cope with the two undesirable phenomena discussed in Section 3.1. When selecting features from Centroid$^{(r)}$: i) the role of self-similarity is degraded, but only for objects containing features that are represented with low weight values in the centroid. Since those features are eliminated, the data objects could be reassigned to another cluster. ii) Most of the eliminated terms belong to the f_j^- set of the noisy features for the cluster which increases its δ-ratio and help the cluster to become more class-discriminative. When considering the MedoidKNN$^{(r)}$: i) the self-similarity may affect only the objects included in the Y set for each

[1]In cases where we need to be more specific we denote explicitly with the superscripts (r) and (s) the reference and the synthetic prototypes, respectively.

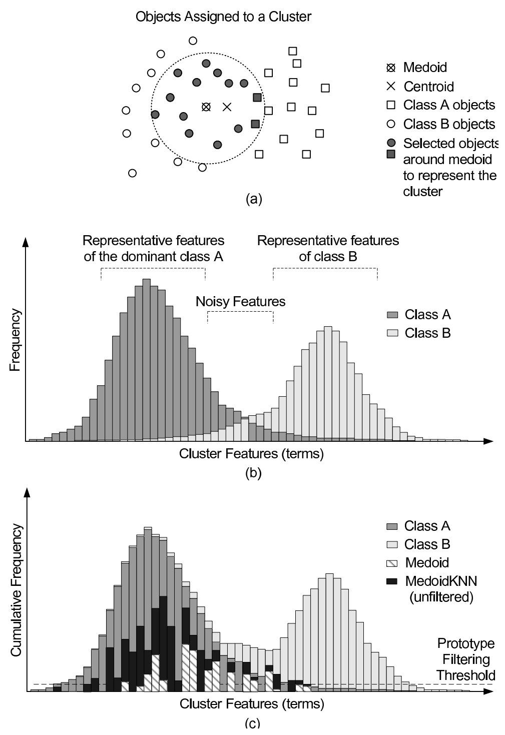


Figure 2: A cluster example that combines two data classes. It illustrates the rationale of using objects around the cluster medoid to favor the representation of the dominant class A and to enable the reassignment of the objects of the other class(es) to other clusters. (a) Object-level view of a cluster where the medoid's nearest neighbors belong mostly to the dominant class. (b) Feature-level view of a multidimensional cluster that illustrates the imaginary histogram of the feature frequency for each of the classes. On the horizontal axis, we suppose an ordering where features that exist in both class (probably noisy) lay between the two peaks of representative class features. (c) The histogram of the cumulative feature frequency over both classes. The respective distributions are also presented for the medoid and the MedoidKNN$^{(r)}$ cluster prototypes.

cluster and again the feature selection on the MedoidKNN$^{(r)}$ also helps some of these objects to move to another cluster, ii) the feature over-aggregation effect is reduced since we use only the vocabulary terms contained in a subset of core objects of a cluster.

Another advantage of k-sp method is that by ignoring some documents that are far from the synthetic prototypes, it provides robustness and ensures that possible outlier and noisy objects will not affect any cluster representation (similarly for noisy features). These objects are not discarded from the dataset. Besides, one object may be ignored as a noisy-outlier at an iteration when computing a cluster representative, while it could be later considered as one core object in case it is reassigned to another cluster, or its current cluster changes dramatically, and the object is now located near the new cluster medoid.

The k-sp exhibits some similarity with the soft subspace clustering methods. The object selection of the reference prototype defines implicitly a feature subspace for a cluster while the feature selection on it explicitly prunes this subspace. Instead of using a separate feature weighting mechanism per cluster, which also doubles the parameters need to be estimated, k-sp uses a heuristic way to directly determine better vector representations for the clusters. Using

Algorithm 1 k-Synthetic Prototypes Clustering Method

```
    function $kSP(k,\ p_{docs},\ p_{terms},\ ref\_flag)$
    input:  the number of clusters $k$, two parameters $p_{docs}, p_{terms}$ (see Algorithm 2), a refinement flag $ref\_flag$
    output: the $k$ clusters and the set of final prototypes
    let:  $C, S, H$, a partition, the synthetic cluster prototypes and the respective clustering cohesion
          $ConstructSP(C,\ p_{docs},\ p_{terms})$ Algorithm 2 for constructing prototypes for each cluster of the partition $C$
          $RefineSolution(C)$ k-sp using Centroid$^{(s)}$ prototypes (regular spk-means) initialized by the partition $C$
    end let
 1: $\{C, S\} \leftarrow InitializeClusters()$
 2: $H \leftarrow Cohesion(C, S)$
 3: repeat
 4:     $\{C^{(prev)}, S^{(prev)}, H^{(prev)}\} \leftarrow \{C, S, H\}$
 5:     $C \leftarrow AssignDocsToClusters()$
 6:     $S \leftarrow ConstructSP(C,\ p_{docs},\ p_{terms})$
 7:     $H \leftarrow Cohesion(C, S)$
 8: until $C \equiv C^{(prev)}$ or $H \leq H^{(prev)}$
 9: if $H < H^{(prev)}$ then
10:     $\{C, S, H\} \leftarrow \{C^{(prev)}, S^{(prev), H^{(prev)}}\}$
11: end if
12: if $ref\_flag$ == TRUE then
13:     $C \leftarrow RefineSolution(C)$
14: end if
15: return $\{C, S\}$
```

object selection it actually tries to favor the representation of the dominant class in a cluster which implicitly results in subspace cluster representation. Another worth mentioning difference is that we claim that after having concluded to a set of synthetic representatives defined in certain feature subspaces, then we may take into account the complete feature space to refine the clustering.

Algorithm 1 provides the pseudocode for the k-sp method that incorporates the synthetic prototypes, constructed using Algorithm 2, into the spk-means algorithm. The clustering cohesion is computed with respect to the synthetic prototypes. It must be noted that k-sp cannot guarantee the monotonicity of convergence. In the case of Centroid$^{(s)}$, we compute the cluster centroid as reference prototype that maximizes the cluster cohesion $\Phi_{coh}(c_j)$, but this optimality is lost after filtering its features. Similarly, for MedoidKNN$^{(s)}$ prototypes, it is not possible to guarantee that cluster cohesion will increase at all iterations and it is essential for k-sp to monitor the objective function and to terminate the procedure if a deterioration of the overall cohesion is observed (the condition $H<H^{(prev)}$ in Algorithm 1). In this case, the clusters of the previous iteration are considered as the solution to the problem produced by the main k-sp procedure.

3.3. Definition of Parameters

The k-sp parameters for computing the MedoidKNN$^{(s)}$ prototype can be defined with respect to the *volume of cluster information*, namely the number of cluster members n_j and the distribution of feature weights aggregated in the reference prototype of a cluster. Two parameters, both in [0, 1], must be specified by the user: p_{docs}, p_{terms}. The number of medoid neighbors K_j is computed as:

$$K_j = \left\lceil p_{docs}\, n_j \right\rceil . \tag{16}$$

Note that different number of neighbors are considered for each cluster c_j. In what concerns the feature selection, an option is to find the $P_j = \lceil p_{terms} |V_j^{(r)}| \rceil$ terms of highest frequency in the reference prototype of rp_j that would cost $O(|V_j^{(r)}|)$. Our implementation uses a more efficient

Algorithm 2 Construct MedoidKNN Synthetic Prototype

```
    function $ConstructSP\,(c,\ p_{docs},\ p_{terms}, \lambda, \beta)$
    input: a cluster $c$, a threshold $p_{docs} \in [0,\ 1]$ that determines the number of documents used for reference prototype
           construction, $p_{terms} \in [0,\ 1]$ for feature selection on it, the number of steps $\lambda$, and a vector $\beta$ of length $\lambda$ that
           control the incremental construction (see Section 3.2)
    output: the synthetic prototype MedoidKNN$^{(s)}$ for cluster $c$
    let: $n_c$ the number of documents in cluster $c$
         $NNDocs\,(c,\ rp,\ r)$ determines the $r$ nearest documents to $rp$ vector in cluster $c$
         $Centroid\,(Y_c)$ computes the centroid of a set $Y_c$
         $FSonRP\,(rp,\ p_{terms})$ applies feature selection on the reference prototype $rp$ based on the parameter $p_{terms}$
             and normalizes the final prototype to unit length ($L_2$-norm)
    end let
 1: $Y_c \leftarrow \{Medoid(c)\}$
 2: $rp \leftarrow Medoid(c)$
 3: $K_c \leftarrow \lceil p_{docs}\ n_c \rceil$
 4: if $K_c > 1$ then
 5:     do for $i$=1,...,$\lambda$
 6:         $Y_c \leftarrow NNDocs\,(c,\ rp,\ \lceil \beta_i K_c \rceil)$
 7:         $rp \leftarrow Centroid\,(Y_c)$
 8:     end for
 9: end if
10: $sp \leftarrow FSonRP\,(rp,\ p_{terms})$
11: return $\{sp\}$
```

approach which is to select the highest weighted features (including the idf component) that contain a fraction p_{terms} of the total feature weight sum $\sum_{i=1}^{V} rp_{ji}$ (*total information*) of the reference prototype vector rp_j. Let $y(i)$, i=1, …, P_j, a function that indexes the selected features which represent the specified p_{terms} information fraction, then P_j is described by:

$$P_j \leq |V_j^{(r)}| : \frac{\sum_{i=1}^{P_j} rp_{jy(i)}}{\sum_{i=1}^{|V_j^{(r)}|} rp_{ji}} \simeq p_{terms}. \tag{17}$$

The more uniform the weight distribution of rp_j, the more features are selected to represent the c_j cluster. Typically, the cost of this operation is $O(|V_j^{(r)}| \log(|V_j^{(r)}|))$, due to the need of weight ordering. However, this can be reduced to $O(|V_j^{(r)}| + z \log z)$ by splitting the range of feature weight values of a cluster into several intervals (bins), where only a small number of features z contained in one bin may be needed to be ordered and then to select the most informative subset out of them.

3.4. Refining the Solution of k-Synthetic Prototypes

The robustness of the proposed k-sp method is the result of its ability to overcome adverse situations in initial clustering iterations and hence to avoid poor locally optimal solutions. After the termination of the *basic* procedure of k-sp method, the result may be further *refined* by considering the centroids of the obtained clusters as the initial prototypes for a final run of k-sp that now coincides with the regular spk-means (this option is enabled by the flag ref_flag in Algorithm 1). This refinement strategy i) aims to improve the result of k-sp method by using more detailed information for homogeneous clusters already produced by the basic k-sp phase, ii) assists in reducing the sensitivity of the k-sp to parameter definition K and P (see Section 4), and iii) constitutes a straightforward approach to choose the best clustering solution among those obtained for different k-sp parameter settings by *comparing the values of the objective function after the refinement step*. This procedure is described in the next section.

The experimentally observed improvement achieved by refinement supports our basic assumption that centroids do not provide sufficient flexibility when clusters are not homogeneous and object reassignments should be encouraged. To tackle this problem one could try to improve the initialization of an iterative method with specialized object-based seeding techniques, or using the clusters produced by a clustering method of different characteristics as the initial partition. Interestingly, the k-sp method is *self-refined* by simply using different values for method parameters, since spk-means is a special k-sp case. The clustering improvement achieved by k-sp refinement phase also confirms that self-similarity and feature over-aggregation play a crucial negative role mostly due the clusters' impurity at the initial iterations of the search procedure. The clusters obtained by the basic phase of k-sp need only a few refining reassignments, thus the self-similarity phenomenon is not a very important issue. Moreover, each respective cluster centroid would have a high δ-ratio (Eq. 15) that enables the fine-tuning of its $|V|$ feature weights which would lead to an improvement in its class-discrimination.

3.5. Selecting the k-sp parameters

An additional advantage of the refinement phase of k-sp, which uses the centroids as cluster prototypes, is that it enables the direct comparison of the results obtained using different values for k-sp parameters. The latter is a very important aspect of k-sp, since it allows the selection of the best setting for parameters p_{docs} and p_{terms}. More specifically, the user could specify two sets of candidate parameter values, the set $S_{p_{docs}}$ for p_{docs} and the set $S_{p_{terms}}$ for p_{terms}. Then, using the same random initial conditions, k-sp runs several times for each combination of the two parameter values and by monitoring the average value of the *refined objective function* (Eq. 5), we can determine which parameter values provide the best average performance. The procedure can be summarized by the following steps:

1. The sets of values $S_{p_{docs}}$ and $S_{p_{terms}}$ are specified by the user.
2. Run k-sp with refinement (Algorithm 1) several times for each combination of parameter values $p_{docs} \in S_{p_{docs}}$ and $p_{terms} \in S_{p_{terms}}$.
3. Compare the average value of the refined objective function of each set to determine the best k-sp average performance and the corresponding parameter values.

Furthermore, the above procedure may reveal important information about the dataset characteristics. As we will see in the experimental section, the observation of better performance provided by smaller synthetic prototypes may indicate that the data clusters are overlapping in many dimensions (i.e. vocabulary terms in common), or that there are a lot of noisy objects/terms.

3.6. Implementation and Complexity

In the present context, where document vectors and cluster centroids are normalized with respect to L_2-norm, it is easy to show that the medoid of a cluster is the cluster object with maximum cosine similarity (dot product) to the centroid of that cluster. Let $u_j = \sum_{d_i \in c_j} d_i \,/\, \left\| \sum_{d_i \in c_j} d_i \right\|_2$ the normalized centroid of cluster c_j with respect to L_2-norm, then Eq. 4 can be expressed as:

$$m_j = \arg\max_{d \in c_j} \left\{ d^{\mathsf{T}} \cdot \sum_{d_i \in c_j} d_i \right\} = \arg\max_{d \in c_j} \left\{ d^{\mathsf{T}} \cdot u_j \right\}. \tag{18}$$

Hence we can determine the medoids of all clusters with linear cost $O(N)$ to the size of the corpus. Thus, both 'spherical' version of k-medoids and k-means method have the same asymptotic cost. It must be noted that it is possible for a cluster to have more than one 'medoid', i.e. objects whose total similarity to the other cluster objects has exactly the same maximum value. Moreover, those objects are equally distant to the cluster centroid. None of them can be considered superior to the others, hence, we can randomly select any of them to construct our synthetic prototype.

Suppose we are given for every object d an ordered list containing the other $N-1$ objects in descending order with respect to their similarity to d. Then it is possible to determine the $K-1$ objects in a cluster that are nearest to its medoid by linearly traversing the respective list ($K-1 \leq N$). By taking advantage of the intracluster rNN-consistency property, we can precompute off-line a number of K_{nn} ($K-1 \leq K_{nn} \leq N$) nearest neighbors for each document in the dataset. If a list has less than $K-1$ objects that are assigned to the same cluster with the medoid object d, we have to necessarily apply greedy search in cluster to locate the rest nearest neighbors to d, up to the desired $K-1$. Supposing that we have set a proper K_{nn} value that eliminates the previously mentioned greedy search, then the non-incremental (λ=1) construction of a MedoidKNN$^{(r)}$ prototype costs $O(n_j + K_{nn} + K\,|V|)$. This includes the cost: i) to determine the medoid document: $O(n_j)$, ii) to locate medoid's $K-1$ nearest neighbors in the cluster: $O(K_{nn})$, and iii) to compute the centroid of the K objects: $O(K\,|V|)$. The latter is the first step of the incremental MedoidKNN^r construction (λ>1). For the steps other than the first we have to seek the nearest documents to the partial centroid (synthetic prototype) computed so far. For the j-th cluster, this can be done by computing and then sorting the pairwise similarities between the $n_j^{(i)}$ data objects and its synthetic prototype in step i, where $i=2,\ldots,\lambda$. Thus, if a subset of $K^{(i)}$ cluster objects are used to construct the MedoidKNN^r for cluster c_j at step i>1, then the construction complexity is $\mathrm{O}(n_j^{(i)}|V| + n_j^{(i)} log(n_j^{(i)}) + K^{(i)}|V|)$.

4. Experimental Evaluation

4.1. Clustering Methods

To provide a comparison of k-sp performance to other clustering methods, we implemented spk-means, k-medoids, hierarchical agglomerative clustering (HAC), and spectral clustering. HAC has been extensively tested on text data [23], herein we have used the *average-link* cluster merging criterion based on the cosine similarity [24]. In addition, we compare k-sp with feature weighting k-means (fwk-means) [39] and entropy weighting k-means (ewk-means) [31] which, according to the comparative result in the latter work, performs better than a series of other soft and hard subspace clustering methods. It is noteworthy that these two methods use the Euclidean distance measure instead of the cosine similarity, whereas for normalized document vectors with respect to the L_2-norm, euclidean and cosine measures determine the same proximity ordering between data objects. The parameters h and γ, respectively, were both set to 1.5 for all datasets. This value was used as well in [39] to apply fwk-means on the 20-Newsgroups dataset that we also use in our experiments. In addition, in [33] it is also reported ewk-means to perform well on the same dataset with γ=1.5. Besides, it is also illustrated that ewk-means is not sensitive to the setting of γ value. Actually, we conducted a number of preliminary tests for these algorithms using parameter values within a wide range, but the observed differences in clustering performance was insignificant.

The spk-means [22] is the baseline approach, the same algorithm is also utilized to refine the solution produced by HAC and k-medoids. In order to show that in the HDS feature space

Table 1: Datasets used in the experimental evaluation

Dataset	Source Docs/Topic	Classes	Docs	Class	$\lvert V\rvert$	consistency		OS	CS
				Balance		1NN	10NN		
$Talk_3$	20-NGs: *guns*, *mideast*, *religion.misc*	3	900	1.0	7051	.952	.854	98.8	98.2
$RS_4^{(S)}$	20-NGs: *autos*, *motorcycles*, *crypt*,	4	800	1.0	3451	.853	.694	98.5	97.2
$RS_4^{(M)}$	*electronics*		1600	1.0	7818	.939	.807	99.3	98.7
$RS_4^{(L)}$			3928	.980	12708	.963	.872	99.6	99.2
$M_6^{(S)}$	20-NGs: *pc.hardware*, *autos*, *baseball*,	6	1200	1.0	7154	.885	.767	99.3	98.2
$M_6^{(M)}$	*hockey*, *electronics*, *med*		3000	1.0	12082	.932	.816	99.6	98.9
$M_6^{(L)}$			5891	.980	17955	.953	.862	99.7	99.2
$M_8^{(S)}$	20-NGs: *atheism*(50,795), *hockey*(100,989),	8	600	.500	4350	.767	.578	98.9	96.9
$M_8^{(M)}$	*windows.x*(100,959), *forsale*(100,957),		2000	1.0	9608	.824	.690	99.4	98.4
$M_8^{(L)}$	*electronics*(100,975), *politics.misc*(100,770)		7355	.780	20592	.912	.783	99.7	99.2
	mac.hardware(50,955), *graphics*(50,955)								
NG_4	20-NGs: *comp.**, *rec.**, *sci.**, *talk.**	4	12000	.985	31498	.954	.877	99.8	99.6
$Mini_{20}$	20-NGs: *from all of the 20 newsgroups*	20	1870	.970	10463	.666	.494	99.4	97.5
Wap_{20}	WebACE	20	1560	.015	8460	.696	.636	98.6	95.8
$K1_6$	WebACE	6	2340	.043	13879	.954	.909	99.1	98.1
Rev_5	TREC	5	4069	.043	23220	.878	.834	99.2	98.
$A_4^{(1)}$	Artificial	4	4000	1.0	9401	.951	.916	99.7	99.5
$A_4^{(2)}$	dataset		4000	1.0	9461	.922	.875	99.7	99.5
$A_4^{(3)}$	generator		4000	1.0	9437	.849	.792	99.6	99.5
$A_4^{(4)}$			4000	1.0	9469	.693	.630	99.6	99.3

marginal clustering improvement should be expected by the careful selection of objects as initial seeds for spk-means, since as explained single objects are inappropriate for representing groups of many objects, some spk-means initialization techniques were also tested: i) the *random clusters* where each object is randomly assigned to one cluster, ii) the *Forgy's approach* were k objects are randomly selected as cluster centroids, and iii) the effective *k-means++* method [41] that try to spread the initial centroids away from each other.

Spectral clustering is based on spectral analysis of the similarity matrix of the dataset. We have used the standard algorithm described in [25]. The basic idea is to project the data in the subspace spanned by the k largest eigenvectors of the Laplacian matrix L, which is computed from the similarity matrix $A^{(N \times N)}$ of pairwise document similarities. The similarity matrix A is computed using the cosine similarity measure. The Laplacian matrix is computed as $L{=}D^{-1/2}AD^{-1/2}$, where D is a diagonal matrix with $D_{ii}{=}\sum_{j=1}^{N} A_{ij}$ the sum of i-th row of similarities. To solve for k clusters, the algorithm proceeds with the construction of a matrix $X^{(N \times M)}{=}\ \{x_i\ : i{=}1, \dots, k\}$ whose columns correspond to the k largest eigenvectors of L. X is then normalized so that each row has unit length in Euclidean space, let $Z^{(N \times k)}$ be the obtained normalized matrix. Finally, the clustering procedure takes place in the embedding space, i.e. the rows of Z are clustered using the standard k-means algorithm, assuming that i-th row of Z represents the i-th document.

Generally, the proposed k-sp variants are denoted by the respective synthetic prototypes they consider, e.g. Centroid-P(p_{terms}), MedoidK(p_{docs})NN-P(p_{terms})[2]. The set of values considered for p_{docs} are: $S_{p_{docs}}{=}\{.90, .80, .60, .40\}$, and for p_{terms}: $S_{p_{terms}}{=}\{.98, .95, .90, .80, .60, .40\}$.

[2]MedoidK($\cdot$)NN is also denoted as K($\cdot$)NN for brevity

In all cases, MedoidKNN$^{(r)}$ has been constructed incrementally in three steps (λ=3) with β_1=0.2, β_2=0.6, β_3=1 (see Section 3.2). In Table 2, we provide the percentage of the original features retained after computing various synthetic prototypes for a specific cluster example to provide a notion of the feature selection that is caused by object selection in a HDS feature space.

4.2. Datasets

4.2.1. Real Data

In order to conduct controlled experiments with respect to the corpus size, cluster sizes and overlap, both real and artificial datasets were used (see Table 1). We constructed a series of clustering problems from real collections, by first selecting certain topics from a collection and then by producing different instances of these problems. In particular, we considered several subsets of the popular 20-Newsgroups[3] collection using as ground truth the provided class label of each document. As an example, $M_6^{(S)}$, $M_6^{(M)}$, $M_6^{(L)}$ are three datasets generated from same topics but with increasing cluster sizes: small, medium, and large that includes all the documents of the selected topics. Mini$_{20}$[4] contains 100 documents from each one of the twenty newsgroups, while NG$_4$ is a subset containing all the four largest subjects in collection, namely *computer*, *records*, *science* and *talk*. Moreover, we used three datasets from the Cluto package[5]: K1$_6$ and Wap$_{20}$ are from the WebACE project and contain web pages from different directories of Yahoo!, Rev5 is derived from the San Jose Mercury newspaper articles that are distributed as part of the TREC collection (TIPSTER Vol. 3).

In brief, in the preprocessing of each dataset, we eliminated trivial terms (stopwords), headers and special tags, we applied *Porter's stemming transformation* [15] and *document frequency thresholding* (*DF*) [17] to discard terms that appear in only one document (dft=1). Thus, all rare terms that have high *discriminating power* were maintained. Finally, we used only documents having more than five terms. In Table 1, we report for each dataset the balance of class sizes, the 1NN and 10NN-consistency (*leave one out* classification accuracy), the overall sparsity (OS) of each dataset which is the average number of zero dimensions that a data vector presents, and the sparsity of each class (CS) when considering only the vocabulary used by the class members (note that OS $\geq$ CS). We also report for the datasets we constructed the number of documents per class that were used in cases of sensible imbalance of class sizes (Docs/Topic).

4.2.2. Artificial Data

In order to construct the artificial text collections we implemented a corpus generator. To generate a corpus with k clusters, our algorithm assumes that the terms (the feature space) are partitioned into k+1 disjoint *topic vocabulary bags*. Each bag B_i, i=1, . . . , k contains the terms related to i-th topic, while an additional bag B_{k+1} contains general terms that could be used in the documents of any cluster.

Each text document is considered to be a sequence of terms and each term of a sequence is generated in two steps: 1) selecting a vocabulary bag, and then 2) selecting a term from that bag. The correlation between a data cluster and the vocabulary bags is user-defined in a $k\times(k+1)$ matrix W, where each element W_{ji} is the probability of selecting the bag B_i when producing a term for a document of the j-th cluster. To sample a term from an already selected bag (step

[3] Available at: http://people.csail.mit.edu/jrennie/20Newsgroups/
[4] Available at: http://kdd.ics.uci.edu/databases/20newsgroups/
[5] Available at: http://www.cs.umn.edu/~karypis/cluto

2) we used the *Zanette-Montemurro stochastic process* (ZM) [32] that has been proposed for generating a single long artificial text that has similar statistical characteristics to real texts, such as the *Zipf's power law* [16] of term frequencies and the sublinear increase of the vocabulary size as the text becomes longer. To achieve these goals the ZM process considers a time decreasing probability controlled by a parameter v of inserting a previously unseen term in the text, i.e. $p_t = \alpha \cdot t^{v-1}$. Otherwise an already selected term of a bag is reselected with a probability proportional to the number of times that has already been used in the created sequence. This property of the process (called 'memory') permits high frequencies for some terms, while the majority of terms present low frequency. In our algorithm, the generation of documents is conducted in cluster order, i.e. the documents of the first cluster then that of the second etc. The memory of the general bag B_{k+1} is maintained during the whole procedure, but the memory of all the other bags is reset when starting the generation of the documents of a new cluster. Using this strategy, in the documents of each cluster a (generally) different set of terms from all the bags would present high frequencies.

To demonstrate the superiority of k-sp performance under situations of clusters that overlap in many dimensions we constructed four artificial datasets called $\mathrm{A}_4^{(i)}$, i=1, . . . , 4 using the above algorithm. All datasets have four clusters (k=4), each of them containing 1000 documents and five topic vocabulary bags were considered with 2000 terms each. The datasets exhibit increasing cluster overlap (from $\mathrm{A}_4^{(1)}$ to $\mathrm{A}_4^{(4)}$), by lowering the probabilities W_{ii} (i=1, . . . , 4) and increasing the probabilities W_{ij} ($j{\neq}i$) of selecting a term from the rest of the bags. The probability matrices W are presented in Fig. 5 (the fifth bag contains the general vocabulary). The length of each document was randomly set by an exponential distribution with mean value λ_{exp}=1/100. The parameter values that we used for the ZM process are α=0.3 and v=0.9.

4.3. Cluster Evaluation Measures

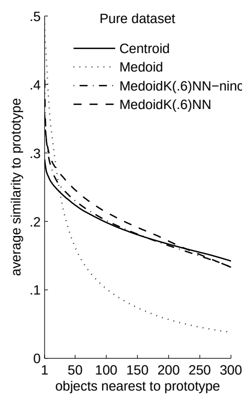
Since we are given the ground truth labeling of the documents in all datasets, clustering evaluation is based on the two popular *supervised* measures *Normalized Mutual Information* (*NMI*) and *Purity*. At this point, we denote: C the clustering solution of k clusters, $c_1, \ldots, c_k$, $C^{(L)}$ the grouping based on ground truth document labels $c_1^{(L)}, \ldots, c_k^{(L)}$ (true classes), N the number of documents in a dataset, N_i the size of $c_i^{(L)}$, n_j the size of c_j, and n_{ij} the number of documents belonging to $c_i^{(L)}$ that are clustered in c_j. Let us further denote the probabilities $p(c_j)$=n_j/N, $p(c_i^{(L)})$=$n_i^{(L)}/N$, and $p(c_i^{(L)}, c_j)$=n_{ij}/N. The [0, 1]-Normalized MI measure, as used in [26], is computed by dividing the MI by the maximum between the cluster and class entropy:

$$NMI(C^{(L)}, C) = \frac{\sum_{\substack{c_i \in C^{(L)},\\ c_j \in C}} p(c_i, c_j^{(L)}) \log_2 \dfrac{p(c_i^{(L)}, c_j)}{p(c_i^{(L)}) p(c_j)}}{\max\left\{H(C^{(L)}),\ H(C)\right\}}. \tag{19}$$

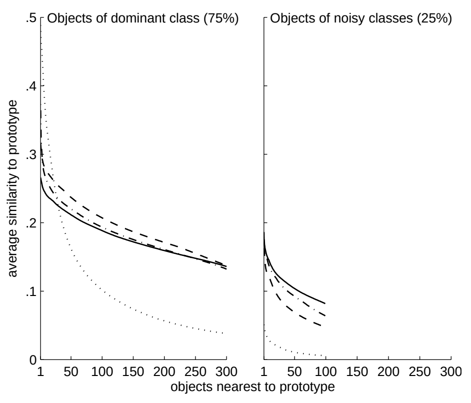
When C and $C^{(L)}$ are independent the value of NMI equals to zero, and to one if these groups contain identical clusters.

The Purity of a cluster can be interpreted as the classification accuracy by assuming that all objects of a cluster are assigned to its dominant class. The clustering Purity is the weighted average of cluster-wise purity:

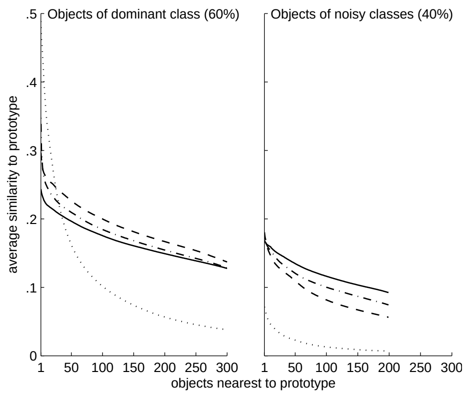
$$Purity(C) = \frac{1}{N} \sum_{j=1}^{k} \max_i \left\{ n_{ij} \right\}. \tag{20}$$



(a)



(b)



(c)

Figure 3: The decrease of average similarity between different types of cluster prototypes and the nearest objects around them as the number of neighbors increase. The datasets consist of objects belonging to a dominant class and two other classes corresponding to noise. We considered three percentages for the objects of the noisy classes: (a) a pure dataset (0%), (b) 25%, and (c) 40%. MedoidK(.6)NN-nincr denotes the reference prototype constructed non-incrementally using the 60% of the objects of each dataset.

Generally, we seek to find a clustering solution that maximizes both NMI and Purity to values close to unit. For each dataset and method we report the values of these indexes. For the methods depending on initialization we also report the average value of each index over the runs on a dataset, while we also report (denoted as 'best') the value of each index (NMI or purity) corresponding to the solution with the highest clustering objective function Φ_{coh} among the 50 runs.

Moreover, in order to evaluate a method's behavior during iterations, we introduce the *Q-index*:

$$Q_t = 1 - \Phi_{ics}^{(t)}(C) \Big/ \Phi_{ics}^{(t-1)}(C), \ t > 0, \tag{21}$$

where $\Phi_{ics}^{(t)}(C)$ is the *intracluster similarity measure* defined as the sum of pairwise cosine simi-

Table 2: The percentage of features retained in the synthetic cluster prototypes for a cluster containing 300 documents from the first topic of Talk$_3$ dataset. The centroid contains all the 4264 non-zero dimensions of the cluster.

Reference Prototype	p_{terms} 1.0	.98	.95	.90	.80	.60	.40
Centroid	100	84.0	72.5	59.3	42.0	21.5	9.8
Medoid	4.6	–	–	–	–	–	–
MedoidK(.9)NN	98.0	82.2	71.1	58.0	40.8	20.7	9.4
MedoidK(.8)NN	95.7	80.6	69.5	56.7	39.6	20.0	9.1
MedoidK(.6)NN	89.5	75.8	65.4	53.1	36.7	18.5	8.5
MedoidK(.4)NN	76.7	65.5	56.4	45.8	31.9	16.2	7.3

larities between objects in the same cluster at iteration t:

$$\Phi_{ics}^{(t)}(C) = \sum_{j=1}^{k} \left[\frac{2}{N\,(n_j - 1)} \sum_{d_i \in c_j} \sum_{d_r \in c_j,\ i<j} d_i^{\top} \cdot d_r \right], \tag{22}$$

where n_j the size of cluster c_j. Initially, we assume that Q_0=0 holds. Higher values of Q-index indicate greater relative improvement of the clustering quality after one iteration.

Finally, the *statistical t-test* was applied to estimate the significance of the average performance difference between k-sp and the methods under comparison for each dataset, except for HAC that is deterministic. Within a confidence interval of 95% and for the value of degrees of freedom equal to $2 \cdot$*number_of_runs*-2 we can test if our method is significantly superior, otherwise the *null hypothesis* is accepted.

4.4. Experimental Results

4.4.1. Robust Cluster Representation

Our first intention in the experiments is to demonstrate the robustness and effectiveness of synthetic prototypes in favoring the representation of the dominant class in a cluster that contains documents from more than one class. To this end we constructed three sets of documents from the topics of Talk$_3$ dataset: a) a pure set of 300 documents from the first topic (0% noisy objects), b) the previous set along with 50 documents from each of the other two topics (25% noisy objects), c) a set of 300, 130, and 70 documents from each topic (40% noisy objects). In all three cases the medoid of the complete dataset belongs to the dominant class (i.e. the first topic). Fig. 3 demonstrates the decrease of average similarity between different types of cluster prototypes considered for the above cases and the nearest objects around them as the number of neighbors increase. We can observe the high average similarity of the medoid with its very close neighbors that decreases rapidly as we consider wider neighborhoods. This indicates that the medoid exhibits high intracluster rNN-consistency (see Section 3.2) and empirically explains why the medoid-based construction of synthetic prototype is more class-discriminative than the centroid-based. The result is the higher average similarity to the members of the dominant class, and the lower similarity values to the documents of other classes (considered as noisy). Furthermore, the incremental construction of MedoidKNN performs better than the direct construction based on the K nearest neighbors of the medoid. Table 2 reports the percentage of features that have non-zero weights after the implicit (i.e. features retained in the reference prototype) and explicit (i.e. additional feature selection on reference prototype) feature selection. We can see the extent to which synthetic prototypes can summarize the characteristics of the document clusters, as well as that synthetic prototypes can discover feature subspaces to represent data clusters.

Table 3: Clustering results on the $M_6^{(S)}$ dataset using k-sp variants.

Reference Prototype	p_{terms}	**NMI** *avg.*	*best*	**Purity** *avg.*	*best*	$\bar{t}$
Centroid	1.0	.480	.564	.630	.751	17.1
Centroid	0.8	.484	.644	.632	.798	16.1
Centroid	0.4	.528	.679	.655	.807	16.3
Medoid	1.0	.286	.424	.504	.648	2.5
MedoidK(.4)NN	1.0	.564	.681	.688	.833	6.9
MedoidK(.8)NN	1.0	.686	.792	.777	.899	13.9

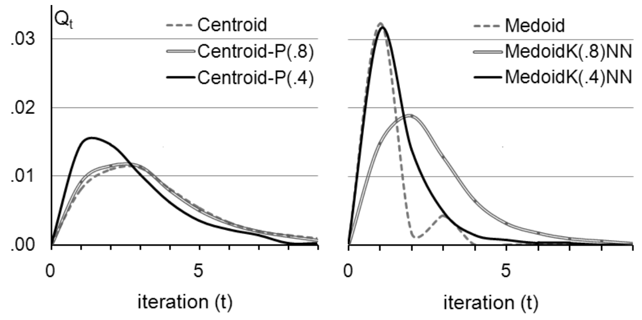


Figure 4: The evolution of the average Q-index with clustering iterations for 50 randomly initialized runs using the $M_6^{(S)}$ dataset.

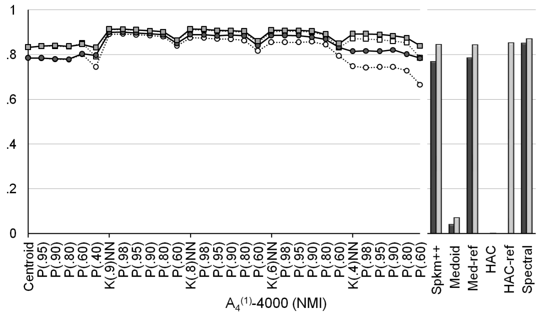
In another experiment we intend to demonstrate the robustness of k-sp under adverse initial conditions. We considered the $M_6^{(S)}$ dataset and examined the case where clusters are initialized by *randomly assigning each document to a cluster*. Table 3 reports the average and best values of the evaluation measures, and the average number of iterations until convergence ($\bar{t}$) for 50 random restarts without refinement. Fig. 4 illustrates how the average Q-index value evolves with iterations for each method. An efficient approach should maximize the area under its corresponding curve, either by executing many iterations or by making larger improvements in shorter time. Fig. 4 indicates the weakness of centroid representation: it defines an optimal cluster representative assuming that all its documents should stay in that cluster. This constrains to a great extent the representation flexibility and forces the procedure to reach poor locally optimal solutions not far from the bad initial clusters. As k-sp becomes more selective on the cluster's features, as in the case of Centroids$^{(r)}$ (e.g. with P(.4)), we observe immediate clustering improvement in the first iterations. However, the main problem remains: the features are selected from the centroids of impure clusters. Despite the fact that medoids lead to a major initial improvement related to a sharper preference to represent one class out of many others in a cluster, subsequently, the procedure converges too early (2.5 iterations on average). On the other hand, the k-sp with MedoidK(.8)NN is a more balanced choice that combines efficiently the advantages of keeping a compact cluster representation and that of considering a wider set of objects around medoid for computing cluster representatives.

4.4.2. Clustering Performance Results

In this section, we provide experimental results using the procedure described in Section 3.5 for the datasets of Table 1 for the two sets of values $S_{p_{docs}}$ and $S_{p_{terms}}$ mentioned in Section 4.1. The results are displayed using the line-plots presented in Fig. 5, 6, 7. The reported 'refined' solutions

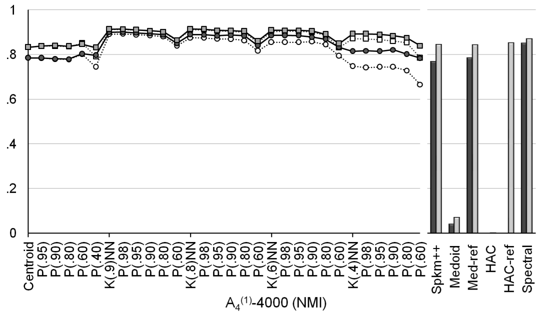
Vocabulary Bags

Clusters	1	2	3	4	5
1	**0.65**	0.04	0.02	0.04	0.25
2	0.13	**0.65**	0.04	0.02	0.16
3	0.01	0.08	**0.63**	0.12	0.16
4	0.08	0.02	0.08	**0.66**	0.16



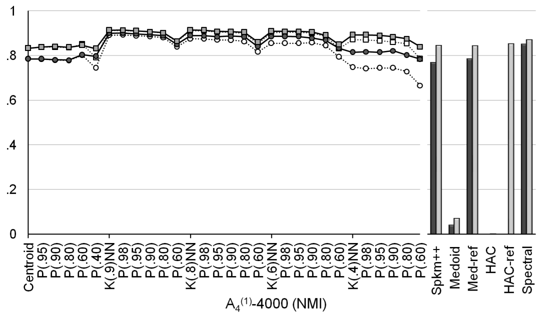
Vocabulary Bags

Clusters	1	2	3	4	5
1	**0.65**	0.04	0.02	0.04	0.25
2	0.13	**0.65**	0.04	0.02	0.16
3	0.01	0.08	**0.63**	0.12	0.16
4	0.08	0.02	0.08	**0.66**	0.16



Vocabulary Bags

Clusters	1	2	3	4	5
1	**0.65**	0.04	0.02	0.04	0.25
2	0.13	**0.65**	0.04	0.02	0.16
3	0.01	0.08	**0.63**	0.12	0.16
4	0.08	0.02	0.08	**0.66**	0.16



Vocabulary Bags

Clusters	1	2	3	4	5
1	**0.65**	0.04	0.02	0.04	0.25
2	0.13	**0.65**	0.04	0.02	0.16
3	0.01	0.08	**0.63**	0.12	0.16
4	0.08	0.02	0.08	**0.66**	0.16

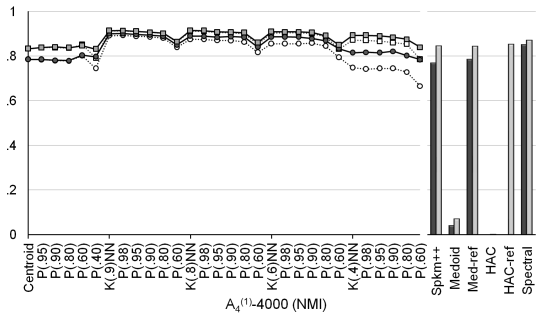


Figure 5: Experimental results on four artificial datasets of increasing cluster overlap, from $A_4^{(1)}$ to $A_4^{(4)}$, where the line-plots indicate the solutions of k-sp method with different parameter values. The respective results for the refined solutions are also reported.

are obtained by k-sp refinement phase using centroids as cluster prototypes (see Section 3.4) on the final clusters of each of the 50 runs of basic k-sp. The bar-graphs in each row of plots present the results for spk-means initialized with the k-means++ heuristic (Spkm++), k-medoids

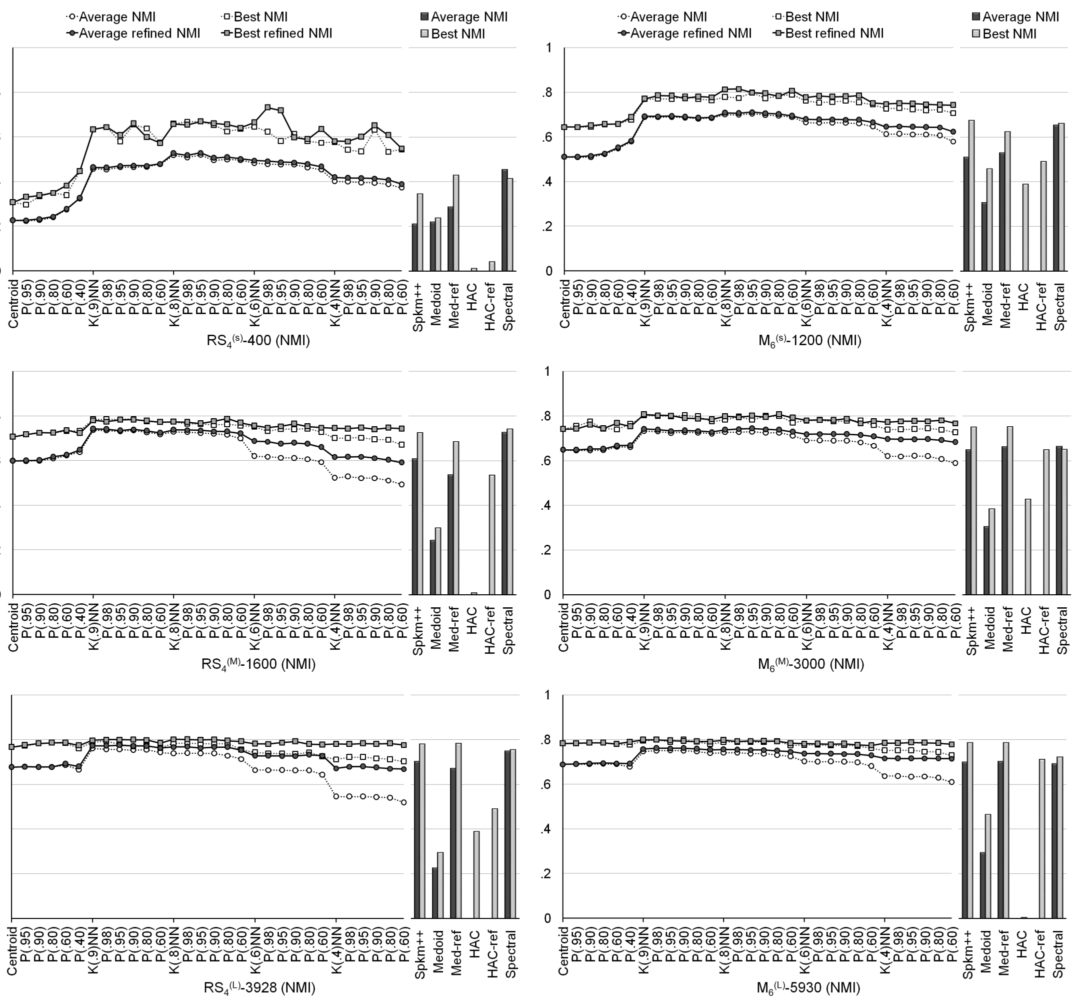


Figure 6: Experimental results for instances of the RS$_4$ and M$_6$ problems with different cluster sizes.

(Medoid), the refined k-medoids (Med-ref), HAC, refined HAC (HAC-ref) using spk-means, and finally the spectral clustering method.

The results on artificial datasets are presented in Fig. 5. For a dataset of small cluster overlap, such as the $A_4^{(1)}$, the performance of k-sp and spectral clustering are quite similar. However, in a more confused setting, such as the $A_4^{(3)}$ and $A_4^{(4)}$ datasets the superiority of k-sp becomes more clear. Moreover, as the overlap between clusters increases, k-sp performs significantly better than the other methods even with lower values of p_{docs} parameter (e.g. 0.6 or 0.4) where the best result is closer to the average performance of the method.

The results on real datasets that are displayed in Fig. 6 and 7 support as well the main idea of this paper. In all cases the k-sp method produced much better results than spk-means. Using MedoidKNN$^{(s)}$ prototypes, the best results for larger datasets are obtained for the K(.9)NN and K(.8)NN cases. Especially for the experiments where we considered three instances of the same problem with increasing size of clusters from small to large (datasets RS$_4$, M$_6$, and M$_8$), it is clear that k-sp using synthetic prototypes manages to overcome the issues arising in the case of

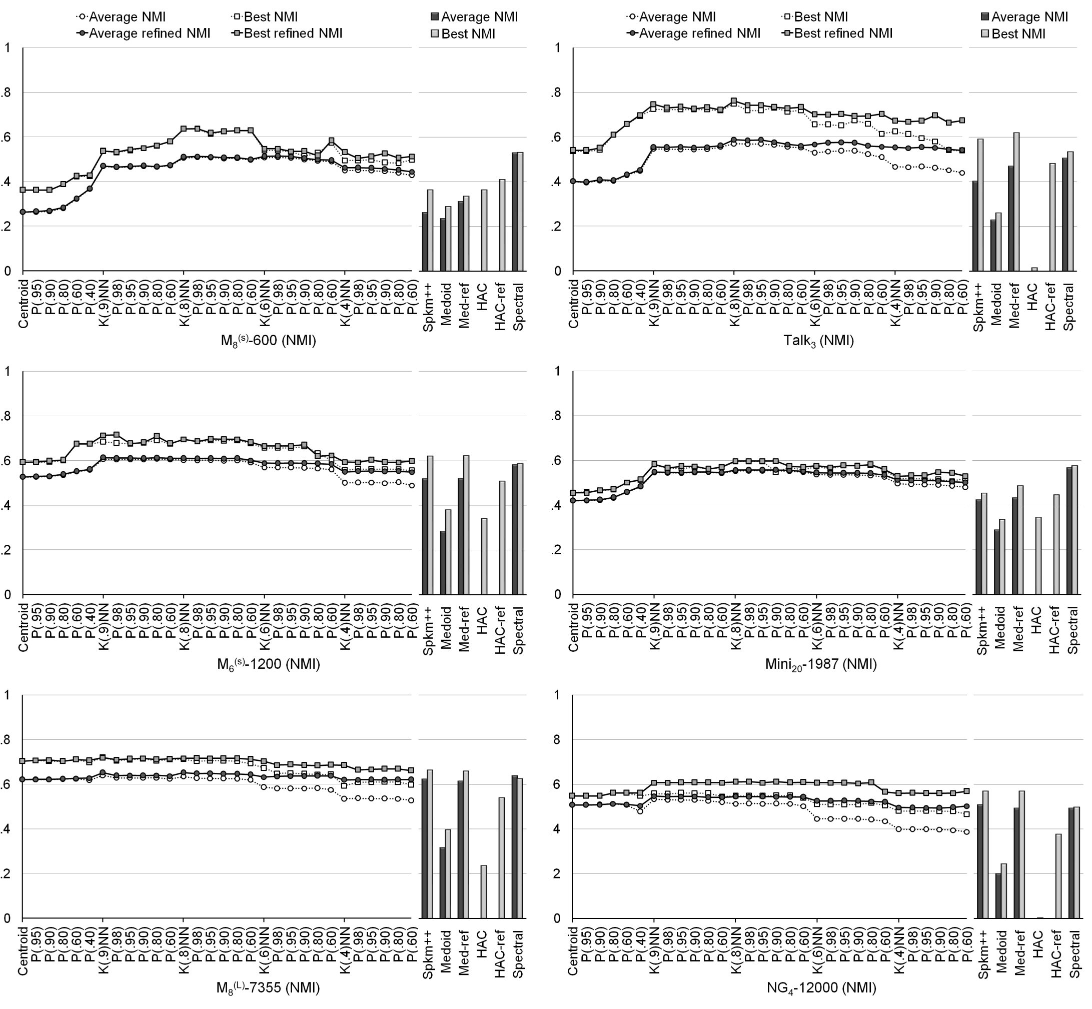


Figure 7: Experimental results for instances of the M_8 problem with different cluster sizes, $Talk_3$, $Mini_{20}$ and NG_4 datasets.

small datasets where the number of objects per cluster is not sufficient, such as self-similarity and feature over-aggregation. The proposed refinement phase leads to even better results, while reducing the sensitivity of setting improper values for k-sp parameters. All the experimentally compared clustering methods performed better when more data objects became available for a specific problem, but the proposed k-sp remained the best among them.

By observing both curves of average and best values of the evaluation measures, we can realize the trade-off in setting k-sp parameters. When limiting the size of synthetic prototypes, k-sp avoids the bad solutions and produces much better clusterings. On the other hand, as synthetic prototypes discard too much information 'detail' from clusters, the basic k-sp procedure becomes unable to identify the fine differences between data classes. This explains the sudden drop of the performance of K(.6) and K(.4) synthetic prototypes for medium and large datasets (e.g. $RS_4^{(M)}$, $RS_4^{(L)}$, $RS_4^{(M)}$, $RS_4^{(L)}$, $M_6^{(M)}$, $M_6^{(L)}$, and $M_8^{(L)}$) when no refinement is applied. The information of the formed clusters can be further exploited by larger synthetic representatives in the refinement

Table 4: The NMI, Purity measures for the refined solutions found for each dataset. Bold values indicate the best result per column. The underlined *t*-values denote the cases where according to the statistical t-test k-sp appears not to be significantly better (0<*t*-val<1.999), or appears to be worse than the compared method (*t*-val<0).

	$A_4^{(1)}$ – k-sp: KNN(.90)-P(.98)						**$A_4^{(2)}$ – k-sp: KNN(.90)-P(1.0)**						**$A_4^{(3)}$ – k-sp: KNN(.80)-P(.98)**						**$A_4^{(4)}$ – k-sp: KNN(.60)-P(.98)**					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
Method	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.901**	**.914**		**.968**	**.978**		**.866**	**.880**		**.960**	**.968**		**.756**	**.774**		**.916**	**.930**		**.433**	**.527**		**.740**	**.816**	
Centroid-P(.6)	.803	.846	06.46	.917	.958	03.49	.714	.801	07.25	.866	.941	05.10	.483	.665	11.84	.730	.886	08.89	.056	.193	25.05	.376	.518	24.29
spk-means	.785	.832	07.51	.909	.950	11.66	.674	.775	09.12	.847	.928	06.48	.394	.626	16.00	.668	.868	12.07	.042	.176	26.34	.357	.512	25.94
spk-means++	.768	.843	07.81	.894	.955	04.55	.692	.779	08.63	.860	.933	05.77	.416	.624	13.98	.691	.862	10.37	.038	.157	27.05	.350	.491	27.19
Medoid-ref	.784	.843	08.08	.911	.955	04.11	.699	.769	08.97	.868	.930	05.86	.423	.628	14.52	.690	.865	10.72	.055	.176	24.96	.373	.512	24.18
fwk-means	.051	.262	80.48	.366	.574	58.60	.289	.160	97.58	.334	.311	81.48	.016	.032	99.83	.314	.360	81.96	.006	.007	30.38	.286	.290	34.75
ewk-means	.131	.302	43.50	.400	.460	34.62	.073	.274	63.02	.372	.540	45.02	.032	.003	57.87	.336	.266	64.44	.009	.006	30.10	.296	.283	33.50
HAC-ref		.851			.936			.802			.936			.450			.659			.156			.418	
Spectral	.850	.869	04.86	.942	.965	02.15	.849	.869	02.05	.941	.965	02.47	.738	.763	02.41	.891	.926	02.55	.021	.021	29.01	.230	.305	32.84

	$RS_4^{(S)}$ – k-sp: KNN(.80)-P(.95)						**$RS_4^{(M)}$ – k-sp: KNN(.80)-P(1.0)**						**$RS_4^{(L)}$ – k-sp: KNN(.90)-P(.80)**						**$Talk_3$ – k-sp: KNN(.80)-P(1.0)**					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
Method	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.529**	.689		**.760**	**.875**		**.738**	**.773**		**.900**	**.926**		**.771**	**.798**		**.916**	**.935**		**.587**	**.762**		**.816**	**.935**	
Centroid-P(.6)	.277	.383	14.92	.565	.695	11.65	.625	.737	06.65	.813	.910	05.12	.691	.786	05.23	.851	.931	04.12	.431	.657	05.78	.728	.900	04.23
spk-means	.226	.307	17.97	.532	.605	13.78	.598	.706	07.93	.798	.892	05.80	.677	.766	07.18	.838	.921	04.79	.401	.540	06.97	.715	.875	04.88
spk-means++	.209	.343	18.35	.508	.623	15.08	.606	.723	08.11	.801	.899	05.97	.700	.778	04.54	.864	.926	03.36	.400	.588	06.68	.717	.823	04.54
Medoid-ref	.285	.427	15.59	.550	.675	13.77	.535	.682	12.91	.730	.876	10.88	.669	.781	06.59	.823	.929	05.50	.468	.617	04.58	.751	.916	03.34
fwk-means	.095	.153	27.86	.420	.493	21.57	.116	.196	53.89	.448	.548	37.55	.140	.257	56.67	.470	.610	39.73	.082	.119	24.20	.510	.551	17.77
ewk-means	.134	.219	24.34	.457	.498	18.86	.219	.357	39.59	.519	.619	29.64	.248	.020	23.85	.499	.288	21.76	.174	.197	17.52	.589	.689	11.90
HAC-ref		.022			.285			.533			.680			.489			.492			.480			.734	
Spectral	.453	.413	04.99	.647	.628	07.91	.725	.740	01.19	.896	.913	00.34	.747	.754	02.77	.911	.919	00.53	.504	.533	04.18	.785	.790	02.05

	$M_6^{(S)}$ – k-sp: KNN(.80)-P(.95)						**$M_6^{(M)}$ – k-sp: KNN(.90)-P(1.0)**						**$M_6^{(L)}$ – k-sp: KNN(.90)-P(.98)**						**Wap_{20} – k-sp: KNN(.80)-P(1.0)**					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
Method	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.711**	**.798**		**.808**	**.904**		**.741**	**.807**		**.835**	**.907**		**.761**	**.799**		**.861**	**.905**		.592	**622**		.658	**.696**	
Centroid-P(.6)	.552	.657	13.12	.670	.814	09.57	.667	.768	05.86	.755	.880	04.97	.693	.780	06.36	.773	.893	06.31	.556	.574	07.56	.621	.637	06.38
spk-means	.510	.644	17.05	.647	.803	11.56	.648	.741	07.36	.742	.870	12.87	.689	.782	06.85	.769	.895	06.74	.538	.544	11.35	.609	.624	08.48
spk-means++	.509	.673	15.85	.641	.831	11.34	.647	.750	07.77	.743	.876	06.12	.698	.785	06.27	.783	.899	05.92	.545	.547	11.15	.616	.609	08.00
Medoid-ref	.527	.622	14.30	.648	.759	10.55	.660	.751	06.28	.753	.876	05.00	.701	.784	05.86	.781	.887	06.17	.548	.576	11.08	.628	.643	06.06
fwk-means	.133	.186	54.59	.372	.443	37.00	.148	.188	53.78	.390	.440	36.52	.160	.241	75.31	.398	.484	51.23	.369	.357	45.34	.486	.487	31.71
ewk-means	.245	.313	49.58	.456	.475	33.78	.323	.295	40.21	.470	.377	29.86	.352	.097	54.27	.461	.271	39.68	.439	.433	09.37	.531	.535	08.52
HAC-ref		.489			.492			.647			.648			.709			.793			.527			.573	
Spectral	.652	.659	06.76	.726	.754	07.03	.662	.649	08.33	.754	.729	06.34	.690	.720	09.37	.771	.821	08.52	**.596**	.602	-1.21	**.664**	.665	-1.43

	$M_8^{(S)}$ – k-sp: KNN(.60)-P(.98)						**$M_8^{(M)}$ – k-sp: KNN(.90)-P(1.0)**						**$M_8^{(L)}$ – k-sp: KNN(.90)-P(.98)**						**Rev_5 – k-sp: KNN(.80)-P(1.0)**					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
Method	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.615**	**.642**		**.706**	**.738**		.692	**.796**		.786	**.904**		**.786**	**.839**		**.854**	**.928**		**.577**	**.676**		**.767**	**.833**	
Centroid-P(.6)	.331	.459	28.81	.511	.582	19.59	.610	.709	08.92	.704	.839	07.45	.734	.828	05.22	.795	.919	04.44	.542	.659	02.67	.745	.828	02.26
spk-means	.275	.367	33.20	.473	.528	23.71	.578	.667	12.14	.688	.812	08.88	.733	.826	05.33	.791	.919	04.74	.535	.651	02.41	.733	.822	02.31
spk-means++	.261	.361	40.56	.450	.537	30.60	.517	.619	20.07	.610	.735	15.66	.622	.661	31.82	.711	.751	20.85	.540	.663	02.42	.739	.819	02.18
Medoid-ref	.332	.445	27.61	.510	.635	18.99	.565	.697	14.13	.668	.833	10.45	.733	.817	05.75	.788	.911	05.14	.526	.653	03.00	.717	.819	03.33
fwk-means	.152	.197	56.08	.360	.400	39.85	.145	.195	59.75	.340	.420	40.94	.156	.246	72.02	.353	.473	45.20	.234	.367	20.82	.566	.700	15.16
ewk-means	.188	.288	49.45	.400	.442	35.13	.281	.279	42.86	.418	.388	32.34	.316	.279	53.99	.418	.364	36.85	.278	.078	05.57	.561	.388	02.27
HAC-ref		.302			.335			.607			.640			.664			.706			.237			.515	
Spectral	**.615**	.620	00.00	.645	.650	08.86	**.733**	.733	-5.48	**.817**	.818	-3.24	.741	.774	05.57	.832	.886	02.26	.406	.411	14.46	.664	.671	11.95

	NG_4 – k-sp: KNN(.90)-P(1.0)						**$Mini_{20}$ – k-sp: KNN(.80)-P(.90)**						**$K1_6$ – k-sp: KNN(.90)-P(1.0)**					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
Method	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.547**	**.607**		**.734**	**.799**		.557	**.597**		**.546**	**.603**		.701	**.802**		.834	**.897**	
Centroid-P(.6)	.510	.561	02.62	.702	.751	02.31	.459	.501	12.59	.442	.484	18.73	.690	.785	01.00	.837	.887	00.37
spk-means	.507	.548	02.73	.699	.748	02.45	.420	.454	30.07	.418	.455	22.24	.675	.725	02.52	.831	.838	01.59
spk-means++	.506	.568	02.70	.696	.755	02.69	.422	.451	30.81	.425	.446	21.99	.680	.770	02.09	.833	.883	01.30
Medoid-ref	.492	.568	03.66	.694	.756	03.00	.431	.484	28.47	.424	.484	20.94	.685	.767	01.59	.829	.885	01.90
fwk-means	.081	.152	42.94	.412	.494	28.20	.081	.152	91.60	.412	.494	64.11	.303	.454	28.07	.715	.756	13.18
ewk-means	.063	.001	27.76	.316	.253	24.93	.286	.312	08.78	.314	.308	46.82	.417	.537	16.36	.762	.827	08.00
HAC-ref		.375			.591			.444			.348			.582			.860	
Spectral	.492	.497	05.43	.711	.714	02.48	**.566**	.573	-04.04	.522	.539	04.66	**.741**	.763	-5.16	**.847**	.860	-1.87

phase (where the centroids are used). Apparently, when larger synthetic prototypes are used in the main phase, the contribution of refinement turns out to be much smaller.

Table 4 summarizes the best and average performance of each method focusing on the refined solutions of k-sp, HAC, and k-medoids. Regarding k-sp, its refinement phase uses the complete feature set and centroids which, as explained in Section 3.5, enables the direct comparison of the solutions corresponding to different parameter values. The supervised evaluation measures that are presented in Table 4 correspond to the set of experiments with the maximum average value of the refined objective function determined by the procedure described in Section 3.5. The k-sp setting that provided this result in each dataset is indicated near the dataset name. The reported best refined k-sp clustering is the best solution using the latter setting of parameter values, whereas it is possible that a different parameter setting may have produced a better solution. The column *t-val* presents the t-value of the significance t-tests between the best k-sp average performance and the average performance of the other methods. For two sets of 50 experiments each, the critical t-value is t_c=1.999 (p_c=5% for p value). This means that if the computed t-value$\geq t_c$, then the null hypothesis is rejected ($p\geq$5%, respectively), i.e. our method is superior, otherwise the *null hypothesis* is accepted indicating a marginal improvement achieved by k-sp. If the t-value is negative, k-sp performs worse than the compared method. In Table 4 the t-values $\leq$ 1.999 are underlined.

According to the significance t-tests, k-sp is clearly superior to the baseline methods such as spk-means, spk-means initialized with the k-means++ technique, k-medoids and HAC as well as their refined solutions using spk-means, and the soft subspace clustering methods fwk-means, ewk-means. Compared to spectral clustering k-sp is superior in most datasets, in terms of both NMI and Purity. Spectral clustering seems to be clearly superior only for datasets $\mathrm{M}_8^{(M)}$ and $\mathrm{K1}_6$. It is also worth mentioning that the computational complexity of spectral clustering is $O(N^3)$ which is significantly higher than that of k-sp. It must be also emphasized that for all datasets the best solutions were provided by the k-sp method.

4.4.3. Discussion

As a general conclusion about the experimental study, it turns out that the refined k-sp approach using medoidKNN with p_{docs}=.9 or .8 seems to be the best method exhibiting superior clustering performance as well as robustness in the case of small, or noisy datasets where the clusters overlap in many dimensions. High values of p_{terms} (e.g. .98 or .95) may also help in some cases. However, we explained in Section 3.5 that the user specifies only the two sets of parameter values $S_{p_{docs}}$, $S_{p_{terms}}$, and the best result can then be identified automatically by examining the values of the objective function of the refined k-sp clusterings. We should also remark that k-sp's feature selection on reference prototypes can efficiently summarize to a great extent the characteristics of the document clusters, since in most cases its application does not deteriorate the clustering performance. When p_{docs} value is kept fixed and small number of features is considered (e.g. p_{terms}=.6 that is expected to be about 20% of cluster's features, see Table 2) then, in most cases, the quality of the clusters produced using MedoidK(p_{docs})NN$^{(s)}$ is comparable to the respective results of the respective unfiltered reference prototypes. As for the Centroid$^{(s)}$, it is the k-sp variant that mostly profits by the prototype filtering. These findings indicate the straightforward applicability of k-sp method to corpus summarization problems or off-line term selection.

In both artificial and real document datasets neither the sophisticated k-means++ initialization, nor the refined k-medoids helped the spk-means to discover much better clusterings. There are also cases where these methods perform equally or worse than typical spk-means. For the refined k-medoids the reason for this observation is explained in Section 3.1 and is related to the inability of any data object to represent a large group of objects in HDS feature space. Thus, spk-means is seeded in a little better way than Forgy's random selection. The fact that spk-means++ and the refined k-medoids perform similarly implies that the probability introduced by the former in order to select objects that are far from each other may not reflect their respective *semantic distance*, since it does not take into account the special properties of text feature space, such as sparsity.

An interesting remark is that the soft subspace clustering methods tested, fwk-means and ewk-means, did not manage to provide satisfactory solutions. In Sections 2.4 and 3.1, we reported as one of their disadvantages the fact that, by introducing explicit feature weights per cluster, the parameters to be estimated are doubled. This becomes more problematic for the very high dimensional datasets used in our experiments. It is worth mentioning that in the experiments in [39] and [33] at most 2000 features were used to represent the documents of datasets containing 2000 to 15905 objects. Apparently, this experimental setting focuses on high dimensional data but of lower scale. The very large scale of dimensionality in our experiments seems to reveal their weakness regarding the number of parameters they use. In most cases, ewk-means presented better results to that of fwk-means with respect to the average evaluation measures. At the same time for many datasets, e.g. $A_4^{(3)}$, $A_4^{(4)}$, and $RS_4^{(L)}$, the best clustering of ewk-means is evaluated to be of lower quality than the average clustering found by the algorithm. This observation indicates that the feature weight entropy term e_j introduced in Eq. 10 may dominate the value of the objective function. We tried to lower down the γ value without observing any improvement. This implies that the feature weight entropy may not always capture the quality of a cluster, whereas numerical issues may also arise for the entropy computation in a HDS feature space.

5. Conclusions

In this paper we have proposed the k-synthetic prototypes (k-sp) clustering method that incorporates the synthetic prototypes into the spherical k-means (spk-means) procedure for document clustering. Through the computation of synthetic prototypes (such as MedoidKNN) cluster-based dynamic feature selection is achieved that favors the representation of the dominant class of a cluster and enables the reassignment of the improperly clustered documents to other clusters. The proposed method is general, simple and effective and includes spherical k-means as a special case. As indicated by extensive experimental results using several datasets, the method provides robust clustering performance especially in cases of small datasets, or noisy clusters that overlap in many dimensions, and compares favorably against spk-means (with Forgy's and k-means++ initialization), k-medoids, HAC, spectral clustering, and the subspace clustering methods fwk-means and ewk-means. It is remarkable that in the HDS feature spaces of the datasets we used, state of the art soft subspace clustering methods did not manage to achieve better solutions even than baseline methods such as spk-means.

The proposed k-sp approach exhibits similarity to subspace clustering methods, since the introduced synthetic prototypes define different subspaces in which data classes are more distinguishable. Therefore, one could argue that k-sp in high dimensional and sparse spaces is also a subspace clustering method. To clarify the differences, we remark that many of the subspace clustering methods [39, 33, 31] construct each cluster prototype by explicitly computing weights for each dimension using all cluster objects. On the other hand, k-sp first applies object selection to construct a reference prototype (resulting in implicit feature selection), and then proceeds with optional explicit feature selection on the reference prototype. Moreover, the motivation of k-sp is to address the self-similarity and feature over-aggregation phenomena that are very intense in the HDS feature spaces. We have also shown that the solutions obtained from the *basic k-sp phase* can be refined by the *refinement k-sp phase* using the whole feature set, which is in contrast with the traditional idea of subspace clustering.

A direction for future work is to extend the feature selection procedure to a continuous weighting scheme, instead of the current binary weighting. It is interesting to investigate the possibility of developing a gradual adjustment of the k-sp parameters aiming to achieve a gradual change of the prototype behavior from medoid-like to centroid-like. This would also eliminate the separate refinement phase. We also aim to test the proposed method to other problems, such as term selection for cluster summarization, organization of noisy document collections, on-line document clustering, and semi-supervised document clustering [56].

References

[1] A. Schenker, M. Last, H. Bunke, A. Kandel, Clustering of Web Documents Using a Graph Model, in: A. Antonacopoulos and J. Hu (Eds.), Web Document Analysis: Challenges and Opportunities, World Scientific Publishing Company, 2003, pp. 3–18.

[2] A. Kalogeratos, A. Likas, A Significance-based Graph Model for Clustering Web Documents, Proc. 4th Hellenic Conf. on AI (SETN'06), Springer, 2006, pp. 516–519.

[3] K.M. Hammouda, M.S. Kamel, Efficient Phrase-based Document Indexing for Web-Document Clustering, IEEE Trans. on Knowledge and Data Engineering 16:10 (2004) 1279–1296.

[4] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[5] Y. Yang, J.P. Pedersen, A Comparative Study on Feature Selection in Text Categorization, Proc. 14th Intern. Conf. on Machine Learning, 1997, pp. 412–420, .

[6] N.M. Wanas, D.A. Said, N.H. Hegazy, N.M. Darwish, A study of global and local thresholding techniques for text categorization, Proc. 5th Australasian Conf. on Data Mining and Anal., ACS, 2006, pp. 91–101.

[7] R. Xu, D. Wunsch II, Survey of Clustering Algorithms, IEEE Trans. on Neural Networks 16:3 (2005) 645–678.

[8] D.M Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet Allocation, Journal of Machine Learning Research 3 (2003) 993–1022.

[9] S. Zhong, J. Ghosh, Generative model-based document clustering: a comparative study, Knowledge and Information Systems 8:3 (2005) 374–384.

[10] J. McQueen, Some Methods for Classification and Analysis of Multivariate Observations, Proc. 5th Berkley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.

[11] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, 1990.

[12] I. Dhillon, Y. Guan, Iterative Clustering of High Dimensional Text Data Augmented by Local Search, Proc. 2nd IEEE Intern. Conf. on Data Mining, Mining, 2002, pp. 131–138.

[13] C. Ding, X. He, K-Nearest-Neighbor Consistency in Data Clustering: Incorporating Local Information into Global Optimization, Proc. Symposium on Applied Computing, 2004, pp. 584–589.

[14] N. Grira, M.E. Houle, Best of Both: A Hybridized Centroid-Medoid Clustering Heuristic, Proc. 24th Intern. Conf. on Machine Learning, 2007, pp. 313–320.

[15] M.F. Porter, An algorithm for suffix stripping, Program 14:3 (1980) 130–137.

[16] G.K. Zipf, *The Psycho-biology of Language, an Introduction to Dynamic Philology*, MIT Press, 1936.

[17] D.D. Lewis, Feature Selection and Feature Extraction for Text Categorization, Proc. Speech and Natural Language Workshop, 1992, pp. 212–217.

[18] G. Salton, A. Wong, C. Yang, A Vector Space Model for Automatic Indexing, Communications of the ACM 18:11 (1975) 613–620.

[19] A. Strehl, J. Ghosh, R. Mooney, Impact of similarity measures on web-page clustering, Proc. AAAI 2000 Workshop on AI for Web Search, 2000, pp. 58–64.

[20] J. Ghosh, A. Strehl, Similarity-Based Text Clustering: A Comparative Study, Grouping Multidimensional Data, Springer, 2006, pp. 73–97.

[21] I.S. Dhillon, J. Fan, Y. Guan, Efficient Clustering of Very Large Document Collections, R. Grossman, G. Kamath, R. Naburu (Eds.), Data Mining for Scientific and Engineering Applications, Kluwer, 2001.

[22] I.S. Dhillon, D.S. Modha, Concept Decomposition for Large Sparse Text Data using Clustering, Machine Learning 42 (2001) 143–175.

[23] Y. Zhao, G. Karypis, Hierarchical Clustering Algorithms for Document Datasets, Data Mining and Knowledge Discovery 10 (2005) 141–168.

[24] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, Proc. KDD Workshop on Text Mining, 2000, pp. 20–23, 2000.

[25] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, In Advances in Neural Information Processing Systems 14 (2001) 849–864.

[26] D. Cai, X. He, J. Han, Document Clustering Using Locality Preserving Indexing, IEEE Trans. on Knowledge and Data Engineering 17:12 (2005) 1624–1637.

[27] K. Beyer, J. Goldstein, R. Ramakrisnan, U. Shaft. When is Nearest Neighbors Meaningful?, Proc. Intern. Conf. on Database Theory (ICDT99), 1999, pp. 217–235.

[28] A. Hinneburg, C.C. Aggarwal, D.A. Keim, What Is the Nearest Neighbor in High Dimensional Spaces?, Proc. 26th Intern. Conf. on Very Large Data Bases (VLDB00), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 506–515.

[29] H.P. Kriegel, P. Kröger, A. Zimek, Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering, ACM Trans. Knowledge Discovery from Data 3:1 (2009) 1–58.

[30] D.S. Modha, W.S. Spangler, Feature Weighting in k-Means Clustering, Machine Learning 52:3 (2003) 217–237.

[31] L. Jing, N. Liping, K. Michael, J.Z. Huang, An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data, IEEE Trans. on Knowledge and Data Engineering 19:8 (2007) 1026–1041.

[32] D.H. Zanette, M.A. Montemurro, Dynamics of Text Generation with Realistic Zipf's Distribution, Journal of Quantitative Linguistics 12:1 (2005) 29–40.

[33] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, D. Papadopoulos, Locally adaptive metrics for clustering high dimensional data, Data Mining and Knowledge Discovery 14:1 (2007) 63–97.

[34] C.Y. Tsai, C.C. Chiu, Developing a feature weight self-adjustment mechanism for a K-means clustering algorithm, Computational Statistics & Data Analysis 52:10 (2008) 4658–4672.

[35] J.M. Pe na, J.A. Lozano, P. Larra naga, An empirical comparison of four initialization methods for the K-Means algorithm, Pattern Recognition Letters 20:10 (1999) 1027–1040.

[36] L. Kaufman, P.J. Rousseeuw, *Finding Groups in Data. An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., Canada, 1990.

[37] J.H. Friedman, J.J. Meulman, *Clustering objects on subsets of attributes*, Journal of the Royal Statistical Society, 66:1 (2004) 815–849.

[38] H. Cheng, K.A. Hua, K. Vu, *Constrained locally weighted clustering*, Proc. Intern. Conf. on Very Large Data Bases (VLDB08), Auckland, New Zealand, 2008, pp. 90–101.

[39] L. Jing, M.K. Ng, J. Xu, J.Z. Huang, *Subspace clustering of text documents with feature weighting k-means algorithm*, Proc. 9th Pacific-Asia Conf. on Knowl. Disc. and Data Mining (PAKDD05), Vietnam, 2005, pp. 802–812.

[40] H.S. Heap, *Information Retrieval: Computational and Theoretical Aspects*, Academic Press, Orlando, USA, 1978.

[41] I.S. Dhillon and Y.Guan and J. Kogan, *Refining clusters in high-dimensional text data*, Proc. Workshop on Clustering High Dimensional Data and Its Applications at the 2nd ACM-SIAM Intern. Conf. on Data Mining (ICML98), Philadelphia, USA, 2002, pp. 71–78.

[42] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, *A local search approximation algorithm for k-means clustering*, Computational Geometry: Theory and Applications, 28:2-3 (2004) 89–112.

[43] P.S. Bradley and U.M. Fayyad, *Refining initial points for k-means clustering*, Proc. 15th Intern. Conf. on Machine Learning, 1998, pp. 91–99.

[44] D. Arthur, S. Vassilvitskii, *k-means++: the advantages of careful seeding*, Proc. 18th ACM-SIAM symposium on Discrete algorithms (SODA07), New Orleans, Louisiana, 2007, pp. 1027–1035.

[45] R. Maitra, *Initializing Partition-Optimization Algorithms*, IEEE/ACM Trans. Computational Biology and Bioinformatics (TCBB09), 6:1 (2009) 144–157.

[46] L. Parsons, E. Haque, H. Liu, *Subspace clustering for high dimensional data: a review*, ACM SIGKDD Explorations Newsletter, 6:1 (2004) 90–105.

[47] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman, *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, 41 (1990) 391–407.

[48] D.M. Blei, A.Y. Ng, M.I. Jordan, *Latent dirichlet allocation*, Journal of Machine Learning Research, 3 (2003) 993–1022.

[49] P. Mitra, C.A. Murthy, S.K. Pal, *Unsupervised Feature selection using feature similarity*, IEEE Trans. Pattern Analysis and Machine Intelligence, 24:3 (2002), 301–312.

[50] N. Wiratunga, R. Lothian, S. Massie, *Unsupervised feature selection for text data*, Proc. 8th European Conf. on Case-Based Reasoning, 2006, pp. 340–354.

[51] Q. Wu, Y. Ye, M. Ng, H. Su, Hanjing, J. Huang, *Exploiting word cluster information for unsupervised feature selection*, Proc. Trends in Artificial Intelligence (PRICAI10), 2010, pp. 292–303.

[52] D. Cai, C. Zhang, X. He, *Unsupervised feature selection for multi-cluster data*, Proc. 16th ACM SIGKDD Intern.

Conf. on Knowledge Discovery and Data Mining (KDD10), 2010, pp. 333–342.
[53] Y. Zhang, Z.H. Zhou, *Multilabel dimensionality reduction via dependence maximization*, ACM Trans. on Knowledge Discovery from Data, 4:3 (2010), 1–21.
[54] Y.B. Liu, J.R. Cai, J. Yin, A.W.C. Fu, *Clustering Text Data Streams*, Journ. of Comp. Sci. and Tech., 23:1 (2008), 112–128.
[55] J.W. Leea, N.H. Park, W.S. Lee, *Efficiently tracing clusters over high-dimensional on-line data streams*, Data and Knowledge Engineering, 68:3 (2009), 362–379.
[56] R. Huang, W. Lam, *An active learning framework for semi-supervised document clustering with language modeling*, Data and Knowledge Engineering, 68:1 (2009), 49–67.