Clustering and classification in Information Retrieval: from standard techniques towards the state of the art

Vincenzo Russo (vincenzo.russo@neminis.org)

Department of Physics
Università degli Studi di Napoli "Federico II"
Complesso Universitario di Monte Sant'Angelo
Via Cinthia, I-80126 Naples, Italy

September 2008

Technical Report TR-9-2008 – SoLCo Project

Abstract

This document is an overview about the clustering and the classification techniques in the Information Retrieval (IR) application domain. The first part of the document covers classical and affirmed techniques both in clustering and in classification for information retrieval. The second part is about the most recent development in the area of the machine learning applied to the document mining. For every technique we cite experiments found in the most important literature.

Contents

List of abbreviations

BVMs	Ball Vector Machines
CCL	Cone Cluster Labeling
CVMs	Core Vector Machines
DF	Document Frequency
EM	Expectation-Maximization
FN	False Negative
FP	False Positive
IDF	Inverse Document Frequency
IR	Information Retrieval
KG	Kernel Grower
k**-NN**	k-Nearest Neighbors
MBI	Minimum Bregman Information
MEB	Minimum Enclosing Ball
MSVC	Multi-sphere Support Vector Clustering
NG20	NewsGroup20
P	Precision
QP	Quadratic Programming
R	Recall
SMO	Sequential Minimal Optmization
SVC	Support Vector Clustering
SVDD	Support Vector Domain Description
SVM	Support Vector Machine
SVMs	Support Vector Machines
UPGMA	Unweighted Pair Group Method with Arithmetic Mean
TF	Term Frequency
TN	True Negative
TP	True Positive

1 Introduction

Information retrieval systems provide access to collection of thousands, millions, even billions of documents.[1] By providing an appropriate description, users can retrieve any of such documents. Usually, users refine their description to satisfy their own needs.

However, many operations in the information retrieval systems can be automated. Processes such as document indexing and query refinement are usually accomplished by computers, but also more complicated tasks like document classification and index term selection could be automatised by machine learning techniques.

In this document we are interested in the machine learning techniques related to clustering and classification in information retrieval.

2 Definitions and document representation

Before going deeper into the matter, we have to establish a coherent terminology, as well as state the basic concepts and the *de facto* standard in representing documents.

2.1 Bag of words model

The *bag-of-words* model is a simplifying assumption used in natural language processing and information retrieval. In this model, a text (such as a sentence or a document) is represented as an unordered collection of words, disregarding grammar and even word order. According to the purpose and the application domain, the size of the vocabulary can be reduced by applying some common techniques, such as the removal of the so called *stop words*, the *stemming* and/or the *lemmatization*.

2.1.1 Stop words

Stop words is the name given to words which are filtered out prior to, or after, processing of natural language data (text). It is controlled by human input and not automated, and it consists of deleting some very common words, such as prepositions, articles. There is no definite list of stop words (also known as *stop-list*) which all natural language processing and information retrieval tools incorporate, but we can find[2] a number of different stop-lists[3] that one may use for his own purpose. Anyway, some tools

[1] Google's servers process 1 petabyte of data every 72 minutes [41].

[2] The SnowBall project provides stop-lists for a number of languages. See `http://snowball.tartarus.org/` for more details.

[3] Obviously, the stop-lists are different for each natural language.

specifically avoid using stop-list to support phrase searching.[4]

2.1.2 Stemming

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The process is helpful for reducing the size of the vocabulary, i.e. the number of words that represents a text. Stemming is also useful in search engines for query expansion or indexing and other natural language processing and information retrieval problems.

2.1.3 Lemmatization

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. In computing, lemmatisation is the algorithmic process of determining the lemma for a given word. Since the process may involve complex tasks such as understanding context and determining the part of speech of a word in a sentence (requiring, for example, knowledge of the grammar of a language) it can be a hard task to implement a lemmatiser for a new language. Lemmatisation is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech. However, stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications.

2.2 The vector space model

Even though the vector space model was first used in an information retrieval system[5], the model is actually used for representing different objects as vectors in many application domains. Therefore, let us give a more general definition of *vector space model*.

Let $o_i \in \mathcal{D}$ be the i-th object of a data set $\mathcal{D}$, we denote with $\vec{x}_i$ the vector derived from the object o_i, where $\vec{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, i.e. the set $\mathcal{X}$ usually is a linear subspace of $\mathbb{R}^d$. Each vector component is called *feature*[6] or *attribute*.

[4]Phrase searching is used to search for words as phrases. That is, the words must be side by side and in the order given. Web search engines like Google run a phrase search when the input query is included between double quotes.

[5]The SMART system developed at the Cornell University in the 1960s.

[6]It actually is an abuse of notation, because the vector component is an *instance* of a feature which is often represented by an *encoding*. For instance, a chair is an object represented by a variety of features, such as the number of legs, the color, the material etc. In a vector space model, a chair will be represented by a real-valued vector whose components will be

With a slight abuse of notation, we call the set $\mathcal{X}$ the *dataset*, notwithstanding the real dataset is the set of objects $\mathcal{D}$, whereas the elements of $\mathcal{X}$ are usually called *points*[7]. Hence we have

$$\mathcal{X} = \{\vec{x}_i | \vec{x}_i = (f_{i_j})_{j=1..d}\}_{i=1..n}$$

where n is the cardinality of the set $\mathcal{X}$, $\vec{x}_i$ are the points, d is the dimensionality of the linear subspace (and so the number of features) and f_{i_j} are the features, where $j = 1, 2, .., d$.

Such point-by-feature data format conceptually corresponds to an $n \times d$ data matrix used by the majority of the clustering algorithms; each matrix row represents an object, whereas in each matrix column we find the value of a specific feature for each objects.

2.3 The vector space model in the information retrieval

In the information retrieval systems the vector space model is widely adopted as now. When we deal with such a system, the vector space model is nothing but that a statement of the bag-of-words model using a mathematical language. This leads to change the basic terminology: in text mining the features are represented by the words in the documents, so the information retrieval experts call they *keywords*, *words* or *terms*, rather than features. The objects are obviously the *documents* and the data matrix mentioned in subsection 2.2 is called *document-term matrix*. Finally, since the number of keywords is often larger than the number of documents, some information retrieval systems represent the documents as transposed vectors. Therefore, the document-term matrix becomes a transposed matrix, i.e. the documents are on the columns of the matrix whereas the keywords are on the rows, and the name accordingly changes in *term-document matrix*.

2.3.1 Term frequency and weighting

Now, we know that documents are represented as vectors of terms, but in which way are these terms expressed? The most simple way is to use boolean vectors, i.e. vectors where each component can only assume the value 0 or 1. In this way we only have a boolean information about terms occurrences in a document, i.e. we only know whether the term is in the document or not. This is not so useful, because it can easily lead to misclassification, since the number of occurrences is more significative than the boolean information "present/not present".

Therefore, the most sophisticated methods rely on the concept of assigning a weight to each term. The weight is always a function of the number

an encoding of each feature instance.

[7] Synonymously, *objects*, *instances*, *patterns*, etc.

of term occurrences, in fact the most simple weighting scheme is to assign the weight of a term t to be equal to the number of occurrences of t in a document x. This weighting scheme is called *Term Frequency (TF)*.

However, the most common weighting scheme adopted in the information retrieval systems is the TF-IDF. As already stated, the TF of a term t is simply the number of occurrences of t in a document. Alternatively, the $\log$-TF of t can be used, that is just the natural logarithm of the TF. The Document Frequency (DF) can be used to attenuate the effect of terms that occur too often in the collection to be meaningful for relevance determination. The DF of a term t is the number of documents in the collection that contain that term. The DF cannot be used as it is to scale the term weight. The Inverse Document Frequency (IDF) is calculated instead. The IDF of a term t is

$$\text{IDF}(t) = \log \frac{n}{\text{DF}(t)}.$$

Therefore, the IDF of a rare term is high, whereas the IDF of a frequent term is likely to be low [34]. The TF-IDF weighting system is simply the product of the TF and IDF.

3 Clustering

Clustering is a technique to group a set of objects into subsets or *clusters*. The goal is to create clusters that are coherent internally, but substantially different from each other, on a per-feature basis. In plain words, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters.

In clustering there is no human supervision: it is the distribution and the nature of data that will determine cluster membership, in opposition to the classification (see section 4) where the classifier learns the association between objects and classes from a so called *training set*, i.e. a set of data correctly labeled by hand, and then replicates the learnt behavior on unlabeled data.

From a practical perspective clustering plays an outstanding role in *data mining* applications.

3.1 Problem statement

Like all machine learning techniques, the clustering problem can be formalized as an optimization problem, i.e. the minimization or maximization of a function subject to a set of constraints. We can generally define the goal of a clustering algorithm as follows.

Given

1. a dataset $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$
2. the desired number of clusters k
3. a function f that evaluates the quality of clustering

we want to compute a mapping

$$\gamma : \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, k\} \tag{1}$$

that minimizes[8] the function f subject to some constraints.

The function f that evaluates the clustering quality are often defined in terms of *similarity*[9] between objects and it is also called *distortion function* or *divergence*. The similarity measure is the key input to a clustering algorithm.

Hence we build an objective function by means of divergence function and some constraints (including the number of aimed clusters) and our goal is to minimize[10] it finding the suitable γ application. In most cases, we also demand that the mapping γ is surjective, i.e. no cluster have to be empty. In this case we can formally state that the goal of a clustering algorithm is to build a partition of the initial dataset $\mathcal{X}$ which have cardinality k.

3.2 Clustering in Information Retrieval

The *cluster hypothesis* states the fundamental assumption we make when using clustering in *information retrieval*.[11]

> **Cluster hypothesis.** Documents in the same cluster behave similarly with respect to relevance to information needs.

The hypothesis states that if there is a document from a cluster that is relevant to search request, then it is likely that other documents from the same cluster are also relevant. This is because the clustering puts together documents that share many features. There are different applications of clustering in information retrieval. They differ in the set of documents that they cluster (collection, result sets, etc.) and the aspect of an information retrieval system that they try to improve (effectiveness, accuracy, user experience, etc.). But they are all based on the aforementioned cluster hypothesis [34].

[8] Or in other cases maximizes.

[9] The similarity could be a proximity measure, a probability, etc.

[10] We know that every maximization problem can be translated in an equivalent minimization problem.

[11] The *cluster hypothesis* holds in case the documents are encoded as *bag of words*, i.e. by using the *vector space model*. See section 2 for details about the vector space model.

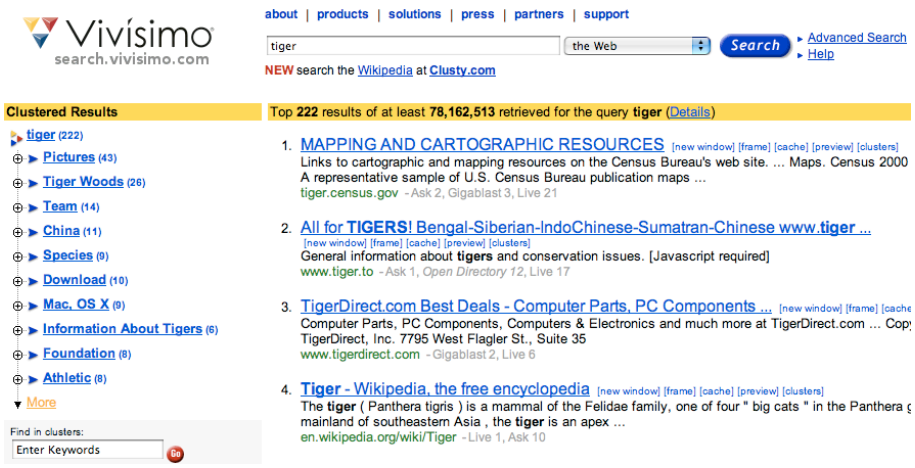

Figure 1: Clustering of search results to improve user recall. None of the top hits cover the "Tiger Woods" sense of *tiger*, but users can easily access it by clicking on the "Tiger Woods" cluster in the *Clustered Results* panel provided by *Vivisimo* search engine.

The first application is the *result set clustering*. In this case the goal is to improve the user experience by grouping together similar search results. Usually it is easier to scan few coherent groups rather than many individual documents, especially if the search terms have different meanings. For instance, let us consider the search term *tiger*; three frequent senses refer to the animal, the golf player "Tiger Woods" and a version of the Apple operating system. If the search engine presents the results showing a number of clusters like "animals", "Tiger Woods", "Mac", and so on, it will allow the end-user to easily discard the results that are not relevant to his search query (see Figure 1).[12]

Another application of clustering in information retrieval is the *scatter-gather* interface. Also in this case, the goal is to improve the *user experience*. Scatter-gather clusters the whole collection to get groups of documents that the user can select ("gather"). The selected groups are merged and the resulting set is again clustered. This process is repeated until a cluster of interest is found.

In the *web mining*, we can find several applications of the clustering in information retrieval. The web mining is a special case of the text documents mining. Currently the web applications that perform the so-called *knowledge aggregation* are proliferating on Internet. Such web softwares heavily rely on data mining techniques, especially clustering. Anyway,

[12]To see such a search engine in action, visit `http://vivisimo.com/` or `http://clusty.com`.

in most of these cases the clustering is semi-supervised because the Web are moving towards a more "semantic" conception. With the term "semi-supervised" we mean that the content have some representative attributes which depends on human operations.[13]

Clustering is also used to increase the precision and/or the recall. The standard inverted index is used to identify an initial set of documents that match the query, but then other documents from the same clusters is added even if they have low similarity to the query. For example, if the query is "car" and several car documents are taken from a cluster of automobile documents, then we can add documents from this cluster that use terms other than "car" (e.g. "automobile", "vehicle", etc.). This can increase recall since a group of documents with high mutual similarity is often relevant as a whole.

This idea has been also used for *language modeling*. It has been showed that to avoid sparse data problems in the language modeling approach to the information retrieval, the model of a document d can be interpolated with a collection model [34, chap. 12]. But the collection contains many documents with words untypical of d. By replacing the collection model with a model derived from the document cluster, we get more accurate estimates of the occurrence probabilities of words in the document d.

Finally, the clustering was recently used also for address the missing values issue: each document usually contains only a small number of the words chosen for representing the documents in a set. This makes the data matrix (see section 2) very sparse. So, the clustering is used to cluster also the words with respect to the documents.

The words clustering is also used for dimensionality reductions, since another distinctive characteristic of the text data are the huge amount of features which describe an object, i.e. the number of words are usually very large (thousands or even millions).[14]

3.3 Hierarchical approaches

The problem stated in subsection 3.1 is only one of the possible formal definition for the clustering problem, though it is the most common and widespread one; it is called *flat partitional clustering*, because it provides a flat partitioning of the original data set, without any explicit structure or information that relate clusters to each other (see Figure 2).

On the contrary, hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as a *dendrogram*. Such an approach could be very useful in document clustering applications.

[13] For instance, tagging a content by means of keywords.

[14] A special version of the NewsGroup20 (NG20) dataset used in [29] has up to $1,355,191$ features. It is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#news20.binary`.

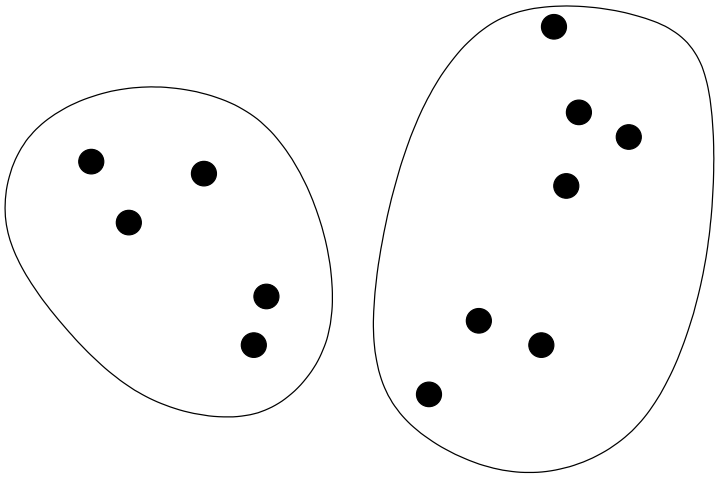

Figure 2: A classic flat clustering results.

Every cluster node contains child clusters; sibling clusters partition the points covered by their common parent (see Figure 3). One of the advantages of the hierarchical clustering is the possibility of examining the clustering results at different levels of granularity, which results in more flexibility. A disadvantage could be the necessity of extending similarity measures to handle similarity among clusters in order to have a way to choose what cluster is appropriate to merge or to split.

4 Classification

Classification is a procedure in which individual items are placed into groups based on quantitative information on one or more characteristics inherent in the items. In plain words, given a set of classes, we seek to determine which class(es) a given object belongs to.

A classifier learns the association between objects and classes by means of a training set, which is a set of objects previously labeled by human beings. Once the training process ends, a classifier is able to classify unlabeled objects assigning them to one (or more) of the classes previously learnt. Anyway, even though a classifier rely on supervised information, it is obviously not perfect.

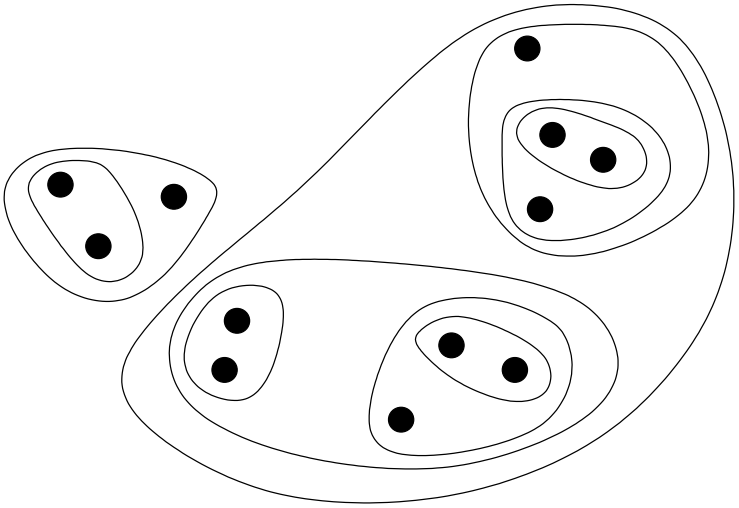

Figure 3: An illustration of typical hierarchical clustering results.

4.1 Problem statement

In classification we are given a description $x \in \mathcal{X}$ of a datum, where $\mathcal{X}$ is the data space, and a fixed set of *classes* $\mathcal{C} = \{c_1, c_2, \cdots, c_l\}$. Typically, the data space is a high-dimensional space, and the classes are human defined for the need of an application. We are also given a *training set* $\mathcal{D}$ of labeled data $\langle x, c\rangle \in \mathcal{X} \times \mathcal{C}$.

Therefore, we wish to learn a *classification function* ϕ that maps the new data to classes

$$\phi\colon \mathcal{X} \to \mathcal{C} \tag{2}$$

In this kind of problem, we have a human being defining classes and labels, and we call such a human being *supervisor*, which is the reason why this type of learning is called *supervised learning*.

4.2 Classification in Information Retrieval

The task of document classification is often a particular type of the classification, the one called "Multi-label classification". In multi-label classification, the examples are associated with a set of labels, that is a very common situation in document mining. Let us take a newspaper as example. The news about the *Iraq war* logically belongs to the category of the "war news" as well as to the category of the "history". The classes in information retrieval field are often called *topics*, and the document classification can be

referred also as to *text classification*, *text categorization* or *topic classification* [34].

Anyway, the notion of classification is very general and has many applications in the information retrieval other than the document classification. For example, the classification can be used in the preprocessing steps necessary for indexing: detecting a document's encoding, word segmentation, documents' language detection, and so forth. Moreover, we can use classification for the detection of the spam texts: this is an example of binary classification in information retrieval, where only two classes are concerned (spam and not-spam). Also, the detection of the explicit contents in textual documents, or the building of a vertical search engine, i.e. search engines which restrict its search process to a particular topic.

4.3 Hierarchical approaches

Hierarchical classification is less usual then the hierarchical clustering: there are only few methods, and most of them are "clustering guided", i.e. they rely on hierarchical clustering techniques for performing a hierarchical classification.[15] This is because the hierarchical classification is just a special case of the multi-label classification.

5 Final goal

The final goal for both document clustering and classification is maximizing the quality of the results. The measure of such a quality is different for clustering and classification.

5.1 Quality of the classification results

When we deal with document classification, the main objective is maximizing the accuracy of the results.

The accuracy is defined as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{3}$$

where the *True Positive (TP)* is the number of documents correctly assigned to a class; the *False Positive (FP)* is the number of documents incorrectly assigned to a class; the *True Negative (TN)* is the number of documents correctly not assigned to a class; the *False Negative (FN)* is the number of documents incorrectly not assigned to a class.

[15] A very recent approach about hierarchical document classification can be found in [26], which performs a hierarchical SVM classification aided by Support Vector Clustering (SVC) method.

As far as the supervised document classification is concerned, three quality measures are used other than the overall accuracy defined above: *Precision (P)*, *Recall (R)* and *F-measure*. All these measures are also known as *external criteria* for the results evaluation.

The precision is the percentage of positive predictions that are correct

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4}$$

whereas the recall is the percentage of positive labeled instances that were predicted as positive

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5}$$

So, the F-measure is the harmonic mean of *P* and *R*

$$\mathrm{F}_\beta = \frac{1}{\alpha\frac{1}{\mathrm{P}} + (1-\alpha)\frac{1}{\mathrm{R}}} = \frac{(\beta^2+1)\mathrm{PR}}{\beta^2\mathrm{P} + \mathrm{R}} \tag{6}$$

where $\beta^2 = 1 - \alpha/\alpha$. Choosing a $\beta > 1$ we can penalize false negatives more strongly than false positives, i.e. we can give more weight to recall. The most common F-measure used in IR is the F1, i.e. the F-measure with $\beta = 1$.

5.2 Quality of the clustering results

The accuracy, the precision, the recall, and the F-measure are all intended for supervised learning. They suppose the availability of labeled data in order to check the quality of the final results against those data. Obviously, this is not possible in real-world clustering application, where the data are completely unlabeled. In this case, we can replace the external criteria with the so-called *relative criteria* also known as *validity indices* [37, sec. 3.3.3]. With relative criteria the challenge is to characterize the clustering result in a way that tells us the quality of the clustering. Since almost each clustering algorithm uses an internal criteria (an internal quality function) to stop the clustering process and yield the result, there is a grey line between measures used by clustering algorithms to determine where to join or split clusters and indices proposed to determine if that was good. A number of validity indices have been developed and proposed in literature and some of them could be embedded in the clustering process to perform an auto-refinement during the clustering itself [34, sec. 16.4.1]. For more information see [37, sec. 3.3.3] and references therein.

6 Issues in document mining

In document mining there are many issues that have to be addressed from both classification and clustering algorithms. The most important issues are:

- *High dimensionality*: documents are often represented by thousand, even millions words. This leads to the *curse of dimensionality* [6];
- *Sparseness of the data matrix*: as stated above, the whole vocabulary could contain million words. Anyway, a single document is not likely to contain all the terms. This is why the final data matrix suffers the sparseness issue;
- *Scalability*: real world data sets may contain hundreds of thousands of documents. Many algorithms work fine on small data sets, but fail to handle large data sets efficiently.

The above issues are about both clustering and classification. Another important issue, that regards *only* clustering techniques, is the *estimation of number of clusters*. In fact, the majority of (flat) clustering techniques requires the number of target clusters as input parameter and it is often a simply guess based on the knowledge of the application domain. This is obviously a limitation and in literature there exists a number of techniques for automatically detecting the number of clusters, and some of them could be directly embedded in the clustering algorithms. Moreover, there exist some clustering algorithms that are able to automatically find the right number of clusters.

7 Towards the state-of-the-art clustering

Currently, in information retrieval there some affirmed clustering techniques, both flat and hierarchical.

7.1 Flat clustering

The most widespread flat clustering techniques are the *K-means*[16] and the *Expectation-Maximization (EM)* [16, 23, 33, 34, 36]. The former is a partitional clustering algorithm: the clustering provided is a partition of the original dataset, i.e. it provides a flat clustering where each document belongs to a single cluster. The K-means is perhaps the most used clustering algorithm across multiple application domains, and it was (and currently is) subject of active research and many improvements, especially for its main issues: the detection of the number of clusters and the initialization of the centroids.

[16] And the highly related *K-medoids* algorithms.

The EM can be considered a generalization of the K-means algorithm. It is a *model based* clustering algorithm, i.e. it assumes that the data were generated by a model and tries to recover the original model from the data. The EM can be tuned for performing both a partitional clustering and a soft clustering, i.e. a clustering process that does not yield a partition of the original dataset as result (the same document can be assigned to more clusters).

Anyway, even though both K-means and EM[17] behave averagely very well in the field of information retrieval, they have their limitation and do not address the already mentioned issues (see section 6).

7.2 Hierarchical clustering

Hierarchical clustering could be very useful in document clustering applications. Anyway, most of the algorithms are not fully reliable in real world applications, and they can also be not scalable. *Divisive algorithms* use a *top-down* approach[18] to build the cluster hierarchy. Such algorithms use a second (flat) clustering algorithm as sub-routine and are not flexible in performing adjustment once a merge or split has been performed. This inflexibility often lowers the clustering accuracy.

As far as *agglomerative algorithms* are concerned, they are more flexible in generating the cluster hierarchy. Anyway, the most accurate agglomerative hierarchical algorithm in the classical literature (the Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [28]) is also the less scalable [23, 42].

On the other hand, an accurate and scalable hierarchical algorithm for document clustering was recently proposed [32], even though it uses neither a divisive approach nor an agglomerative one, but an alternative approach called *frequent itemsets*.

7.3 Proposed state-of-the-art techniques

In this section we wish to propose the utilization of some recently published clustering techniques. Such state-of-the-art clustering methods address some (or all) issues outlined in section 6. The first technique is the *Bregman Co-clustering* [4, 15], whereas the second technique is the *SVC* [37, ch. 6].

[17] And all their variations and improvements too.

[18] They start from at the top with all documents in one cluster, then this cluster is split using a flat clustering algorithm. This procedure is applied recursively until each document is in its own singleton cluster or (actually) when a certain stopping criterion is fulfilled.

7.3.1 Bregman Co-clustering

The Bregman Co-clustering relies on a new clustering paradigm, namely the *co-clustering*. Recently, the research effort in the co-clustering direction is increased, for several reasons. One motivation is the need of contextually performing the dimensionality reduction during the document clustering,[19] instead of performing a generic dimensionality reduction as a preprocessing step for clustering. Another statistically important reason is the necessity of revealing the interplay between similar documents and their words clusters. This is a statistical information we can obtain only by solving a co-clustering problem.

The main goal of the co-clustering problem is to find co-clusters instead of simple clusters. A co-cluster is a group of documents and words that are interrelated. The words in a co-cluster are the terms used for choosing the documents to put in that co-cluster.

There are different approaches to perform the co-clustering, and some of them have the great limitation of finding just one co-cluster for each algorithm execution. The most interesting approaches are a framework based on the information theory [17, 19, 20] and a minimum sum-squared residue framework [13]. The generalization of these two methods lead to the Bregman Co-clustering framework.

All the above three frameworks use a generalized K-means like strategy, i.e. an alternate minimization scheme for clustering both dimensions of a document-term matrix. The main improvement of the Bregman framework is the employment of a large class of well-behaved distance functions, the *Bregman divergences* [9], which also include well known distances, such as the *Squared Euclidean*, the *KL divergence* (relative entropy), etc. Such a general framework allows to make not assumptions on the data matrices. In fact, the fundamental properties (e.g. the convergence of the algorithm which solves the problem) hold in the general case, i.e. for the problem stated with a "general" Bregman divergence. Then, we can choose the most appropriate divergence according to the application domain.[20]

Furthermore, the Bregman Co-clustering addresses the *high dimensionality* and the *data matrix sparseness* issues mentioned in section 6, as well as the problem of clustering in presence of missing data values [37, ch. 10]. A very recent improvement of Bregman Co-clustering, called *Bregman Bubble Co-clustering* [15], is also able to deal with *outliers* and automatically estimate the number of (co-)clusters.

Finally, this framework allows performing other interesting operations on data matrices, such as the *missing values prediction* and the *matrix approx-*

[19] We recall that in subsection 3.2 we stated that the clustering can be used also for dimensionality reduction by means of words clustering.

[20] As we will see in the next part of this document, the divergences that seems to fit the document clustering problem well are the KL-divergence and the I-divergence.

imation (or compression). These peculiarities make this framework a more general unsupervised tool for data analysis.

7.3.2 Support Vector Clustering

The *Support Vector Clustering (SVC)* was the first proposed clustering methods that relies on Support Vector Machines (SVMs) [8]. SVC relies on the *Support Vector Domain Description (SVDD)*[21] problem [38] and is able to work with a variety of kernel functions. In [37, ch. 6] there is an exhaustive review of almost all the literature regarding the SVC, plus some additional contributions.

This clustering algorithm addresses the issues mentioned in section 6, such as the *sparseness of the data matrix*, the *automatic detection of the number of clusters*; and it even shown a well behavior with *very high dimensional data*. Moreover, it deals with *outliers* and with cluster of *arbitrary shapes*. Finally, it can be also used as a *divisive hierarchical* clustering algorithm, if we drop the constraint of having a strict hierarchy [7, 25].

SVC is more experimental than the Bregman Co-clustering framework and some other issues are still open, but could provide very interesting features and yield very good results too.

As far as alternative support vector methods for clustering are concerned [37, ch. 7], it worths to explore the *Kernel Grower (KG)* algorithm. Kernel Grower also relies on the SVDD problem, but uses a K-means-like strategy too: it tries to find K spherical-like clusters in a higher dimensional features space, where the centroids are the centers of those hyper-spheres. By adopting a K-means-like strategy in a higher dimensional space, KG is more likely to be successful with respect to those problems that are non-separable in the data space. The main drawback is the need of the number clusters K as input parameter, but it can be solved with some heuristics like the approach used by *Multi-sphere Support Vector Clustering (MSVC)* [12].

7.4 Proposed state-of-the-art techniques: computational complexity

Achieving better results is surely the main goal in every data mining application. Anyway, we have to make a trade-off between the quality of the results and the computational complexity of the employed algorithms, i.e. the algorithms need to scale in real-world applications which means a gigantic amount of data when the application domain is the information retrieval.

Therefore, let us analyze the complexity of the chosen state-of-the-art algorithms.

[21] Which finds a hypersphere in a higher dimensional feature space.

7.4.1 Bregman Co-clustering

According to the Minimum Bregman Information (MBI) solution form (closed or not), we distinguish two cases for calculating the computational complexity. As stated in [37, sec. 5.4.6.4], when the instance of the MBI problem at hand has a closed form solution,[22] the resulting algorithm involves a computational effort that is linear per iteration in the size of the data and are hence scalable, because the complexity of the MBI solution computation can be considered $O(1)$. In this case it is enough to calculate the computational complexity of the Bregman block average co-clustering. Let I be the number of iterations, let k and l be the number of row clusters and column clusters respectively, let d be the dimensionality of the space and let m and n be the number of rows and columns of a data matrix respectively. **The worst-case running time complexity in the case of a closed form MBI solution is**

$$O(I(km + ln)d)$$

When we deal with an instance that leads to a non-closed form MBI solution (e.g. with the Itakura-Saito distance), we have to consider also the complexity of the algorithm we have chosen to analytically compute the MBI solution, i.e. either the complexity of the *Bregman's Algorithm* [9] or the complexity of the *iterative scaling algorithm* [14].

7.4.2 Support Vector Clustering

We recall that the SVC is composed of two steps: the *cluster description* and the *cluster labeling*. To calculate the overall time complexity we have to analyze each step separately.

The complexity of the cluster description is the complexity of the Quadratic Programming (QP) problem [38] we have to solve for finding the Minimum Enclosing Ball (MEB). Such a problem has $O(n^3)$ worst-case running time complexity. Anyway, the QP problem can be solved through efficient approximation algorithms like *Sequential Minimal Optmization (SMO)* [35] and many other decomposition methods. These methods can practically scale down the worst-case running time complexity to (approximately) $O(n^2)$ [8, sec. 5].

Without going deeper into the matter, if we use the original cluster labeling algorithm [8] the complexity of this step is $O(\overline{n}^2 n_{sv} m)$, where $\overline{n} = n - n_{bsv}$, n_{sv} is the number of support vectors, n_{bsv} is the number of bounded support vectors, n is the number of items, and m is a value usually between 10 and 20.

[22] We recall that this is definitely true for (i) the co-clustering basis $\mathcal{C}_2$ and all Bregman divergences; (ii) for the I-divergence and all co-clustering bases; (iii) for the Euclidean distance and all co-clustering bases.

Anyway, if we employ the *Cone Cluster Labeling (CCL)* (which is the best cluster labeling algorithm in terms of accuracy/speed trade-off), the complexity of the labeling stage becomes $O((n - n_{sv})n_{sv})$, which can be absorbed in the computational complexity of the cluster descriptions stage, so we have $O(n^2)$ overall computational complexity.

We could further scale down such a complexity by using different Support Vector Machine (SVM) formalisms like Core Vector Machines (CVMs) [40] and Ball Vector Machines (BVMs) [39].

7.5 Proposed state-of-the-art techniques: experimental results

In this section we cite a number of experiments executed all around the world by different researchers, but no new (our own) experiments will be provided.[23] The experiments concerns both Bregman Co-clustering and support vector methods for clustering.

7.5.1 Bregman Co-clustering

The experiments mentioned here (look at Table 1, Table 2, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8) are from a number of publications [2, 3, 4, 17, 19, 20, 37] and we have not performed those experiments again so far. The datasets used are

- CLASSIC3 [2, 3, 4, 17, 19, 20, 37];
- PORE [37, ch. 9];
- SCI3 [37, ch. 9];
- Binary, Binary_subject [20];
- Multi5, Multi5_subject [20];
- Multi10, Multi10_subject [20];
- Different-1000 [3].

Since the good results obtained in above cited papers, it worths to spend resources and time to better explore the applicability of the Bregman Co-clustering to the text clustering: its ability to deal with high dimensional and sparse data are two important peculiarities necessary to a clustering algorithm to work in text mining application domain. Moreover, by studying in depth the Bregman Bubble Co-clustering [15], we may be able to need not "guessing" the number of clusters.

[23] An experimental session is a future-scheduled task.

CLASSIC3 - Euclidean - No feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	72.47%	53.01%	61.24%	25.31%	43.27%	31.94%	39.09%	29.57%	33.68%
$\mathcal{C}_{2,4}$	72.5%	53.08%	61.28%	25.41%	43.47%	32.08%	39.17%	29.57%	33.7%
$\mathcal{C}_3$	43.13%	87.67%	57.82%	55%	19.17%	28.44%	30.09%	12.14%	17.3%
$\mathcal{C}_5$	37.6%	37.26%	37.42%	29.05%	33.88%	31.28%	36.5%	32.36%	34.3%
$\mathcal{C}_6$	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%	**100**%
CLASSIC3 - Information-Theoretic - No feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	37.77%	33.42%	35.46%	24.77%	31.46%	27.72%	37.39%	34.43%	35.84%
$\mathcal{C}_2$	38.14%	32.6%	35.16%	26.82%	34.27%	30.1%	36.45%	34.5%	35.44%
$\mathcal{C}_3$	36.87%	31.64%	34.06%	24.48%	30.59%	27.2%	36.03%	34.71%	35.36%
$\mathcal{C}_4$	36.48%	31.23%	33.66%	27.34%	34.56%	30.52%	36.95%	35.29%	36.1%
$\mathcal{C}_5$	37.74%	33.22%	35.34%	28.66%	37.75%	32.58%	35.93%	32%	33.86%
$\mathcal{C}_6$	36.76%	34.52%	35.6%	27.92%	33.3%	30.38%	37.6%	34.64%	36.06%

CLASSIC3 - Euclidean - No feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	41.998%	49.782%	42.332%	42.076%	34.601%	**100%**
Macroaveraging	42.282%	40.643%	34.515%	42.354%	34.337%	**100**%
CLASSIC3 - Information-Theoretic - No feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	33.265%	33.727%	32.469%	33.573%	33.984%	34.241%
Macroaveraging	33.011%	33.565%	32.203%	33.426%	33.923%	34.014%

Table 1: From [37]. Bregman Co-clustering of the CLASSIC3, without feature clustering. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. The Euclidean $\mathcal{C}_6$ scheme yields the best results.

7.5.2 Support Vector Clustering

The experiments mentioned here (look at Table 3[24], Figure 9[25]) are from [37] which is the only source where we found SVC applied to the document clustering. In addition we provide the results of a Kernel Grower (KG) experiment. We have not performed again those experiments, so far. The datasets used are

- CLASSIC3 [37, ch. 9];
- PORE [37, ch. 9];
- SCI3 [37, ch. 9];
- Spambase [11].

SVC failed on SCI3 and PORE data. The reasons could be a number. Above all, the SCI3 dataset dimensionality is about three times of the CLASSIC3 dataset dimensionality, and the PORE dimensionality is about

[24]Here LC, GC, and EC means the SVC with Laplace, Gauss and Exponential kernels respectively.

[25]Here the KG is called KMC.

CLASSIC3 - Euclidean - With feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	31.88%	39.52%	35.3%	50.8%	49.27%	50.02%	22.66%	17.5%	19.74%
$\mathcal{C}_{2-6}$	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

CLASSIC3 - Information-Theoretic - With feature clustering									
Basis	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
$\mathcal{C}_1$	39.16%	33.15%	35.9%	28.06%	36.21%	31.62%	38.9%	36.79%	37.82%
$\mathcal{C}_2$	35.61%	31.37%	33.36%	26.72%	34.66%	30.18%	37.25%	33.71%	35.4%
$\mathcal{C}_3$	37.59%	31.64%	34.36%	25.08%	31.85%	28.06%	36.76%	35.5%	36.12%
$\mathcal{C}_4$	36.5%	32.12%	34.18%	25.24%	31.17%	27.9%	35.96%	34.21%	35.06%
$\mathcal{C}_5$	**99.86%**	**100%**	**99.92%**	**99.81%**	**99.9%**	**99.86%**	**99.93%**	**99.71%**	**99.82%**
$\mathcal{C}_6$	37.9%	33.56%	35.6%	26.04%	32.14%	28.78%	35.7%	33.79%	34.72%

CLASSIC3 - Euclidean - With feature clustering		
	$\mathcal{C}_1$	$\mathcal{C}_{2-6}$
Accuracy	34.19%	**100%**
Macroaveraging	35.021%	**100%**

CLASSIC3 - Information-Theoretic - With feature clustering						
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Accuracy	35.268%	33.085%	33.085%	32.623%	**99.872%**	33.265%
Macroaveraging	35.111%	32.975%	32.847%	32.376%	**99.869%**	33.028%

Table 2: From [37]. Bregman Co-clustering of the CLASSIC3, with 10 feature clusters. The first table shows the Precision (P), Recall (R) and F1 for each class and for each co-clustering basis. The second one shows the Accuracy and the Macroaveraging. Best results are in bold.

four times. Hence, it is likely that authors found a dimensionality limit for the CCL.[26]

Moreover the SCI3 and PORE datasets were built by [37] and neither stemming or lemmatization were applied. They used just the *MC Toolkit* [18] feature selection on the whole dictionary. Therefore, it is likely that the construction of these two datasets could be improved (also employing a more sophisticated feature selection strategy) so that the SVC with the CCL could separate them.

However, it worths to spend resources and time to better explore the applicability of the SVC to the text clustering: its ability to deal with high dimensional[27] and sparse data are two important peculiarities necessary to a clustering algorithm to work in text mining application domain. Furthermore, due to its pseudo-hierarchical behavior, the SVC could also be employed for hierarchical automatic organization of text documents, a common task in such application domain.

8 Towards the state-of-the-art classification

In information retrieval there are a number of classification algorithms that worked well in the past, like the *Naive Bayes*, *Rocchio* algorithm, the k-

[26] The cluster labeling algorithm used for the experiments.

[27] This capability can be improved.

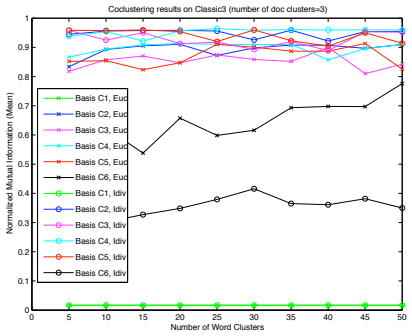

Figure 7.6: Co-clustering results from *CLASSIC3*—6 bases and 2 divergences. Bases $\mathcal{C}_2 - \mathcal{C}_5$ perform very well in getting back the hidden true labels. Basis $\mathcal{C}_1$ performs the worst as it has access to minimal amount of information. Interestingly, basis $\mathcal{C}_6$, in spite of having the maximal information, performs poorly according to NMI. Possibly $\mathcal{C}_6$ is overfitting, i.e., finding some additional structure in the data that goes beyond what is needed to get the labels right. There is no significant difference between the two loss functions used.

Figure 4: CLASSIC3 data, from [4]

Nearest Neighbors (k-NN) [34], *Centroid-based classification algorithms* [24], *Support Vector Machines (SVMs)* [5].

8.1 Support Vector Machines

The research effort around the SVMs is huge. SVMs has been largely employed for text classification too, and they have rapidly become the state of the art tools for such an application domain. The information retrieval research community made a number of contributions to the SVMs research, like kernels that are specific for the text classification (the *string kernel*, the *lexical kernel* and the *tree kernel* [5]).

SVMs address the issues stated in section 6 [1, 27].

8.2 Proposed state-of-the-art technique

In this section we wish to propose the use of the *Infinite Ensemble Learning via SVMs* in document classification.

The *ensemble learning* area encloses some techniques like *boosting* or *bagging* [21, 10, 22]. These techniques are intended for both speeding up and creating more stable classifiers: conceptually, they consider a set of simpler classifiers (also known as hypotheses) and then make a linear combination of them. The main drawback of such algorithms is that the number of hypotheses is finite.

A recent proposed technique [31, 30] uses particular kernels for achieving infinite ensemble learning via SVMs: it can be shown that some kernels

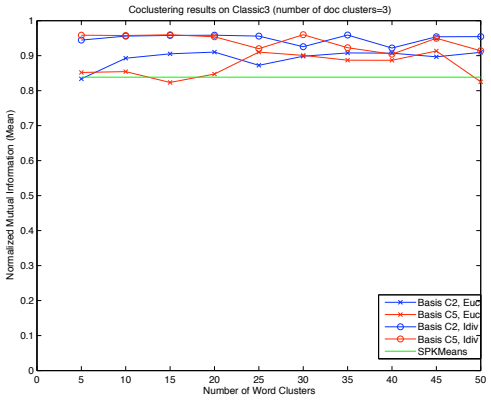

Figure 7.7: Co-clustering on *CLASSIC3*—Bases $\mathcal{C}_2$ and $\mathcal{C}_5$ using squared Euclidean distance and I-divergence compared with SPKmeans. The co-clustering results compare favorably to SPKmeans.

Figure 5: CLASSIC3 data, from [4]

embed an infinite number of hypotheses.[28] Let us refer to these kernels as to "infinite ensemble kernels"; some of them are the *Stump Kernel* and the *Perceptron Kernel* [31, 30].

This kind of approach has two major advantages: it provides more stable and accurate classifiers (because of the ensemble learning paradigm) and the speed of an SVM classifier that uses the infinite ensemble kernels is generally greater than the speed of SVMs that use classical kernels (like Gaussian, Laplacian, etc.). The latter is due to the infinite ensemble kernels, which usually have a simpler form than the classical kernels, and this implies a faster parameters selection.[29]

8.3 Proposed state-of-the-art technique: computational complexity

The computational complexity of the infinite ensemble SVMs is the same complexity of a classical SVM: it scales between $O(n)$ and $O(n^{2.3})$ for state-of-the-art implementations. Moreover, by using the infinite ensemble kernels, the parameters selection is considerably faster, e.g. the parameters selection for the stump kernel and the perceptron kernel can be even ten times faster than the parameters selection for the Gaussian kernel or the Exponential kernel.

[28] Moreover, the set of the hypotheses can be even uncountable.

[29] SVM parameters selection is usually one of the main causes of the SVM slowness.

Table 6: Effect of Implicit Dimensionality Reduction by Co-clustering on Classic3. Each subtable is for a fixed number of (document,word) co-cluster.

(3,20)			(3,500)			(3,2500)		
1389	1	2	1364	3	18	920	49	292
9	1455	33	5	1446	21	31	1239	404
0	4	998	29	11	994	447	172	337

Table 7: Effect of Implicit Dimensionality Reduction by Co-clustering on Different-1000. Each subtable is for a fixed number of (document,word) co-cluster.

(3,20)			(3,500)			(3,2500)		
949	15	32	435	364	146	376	368	283
31	925	80	393	454	85	251	367	363
20	60	888	172	182	769	373	265	354

Figure 6: CLASSIC3 and Different-1000 data, from [3]

8.4 Proposed state-of-the-art technique: experimental results

To the best of our knowledge, there are no available results of this technique applied to the document clustering. The only results available are on other kind of datasets [31, 30]. Anyway, such results plus the stability and the speed of the suggested technique are remarkable enough that worth to try it for document classification.

Co-clustering			1D-clustering		
992	4	8	**944**	9	98
40	**1452**	7	71	**1431**	5
1	4	**1387**	18	20	**1297**

Table 3: Co-clustering accurately recovers original clusters in the *CLASSIC3* data set.

Binary				Binary_subject			
Co-clustering		1D-clustering		Co-clustering		1D-clustering	
244	4	**178**	104	**241**	11	**179**	94
6	**246**	72	**146**	9	**239**	71	**156**

Table 4: Co-clustering obtains better clustering results compared to one dimensional document clustering on Binary and Binary_subject data sets

Figure 7: CLASSIC3 and Binary data, from [20]

	Co-clustering	1D-clustering	IB-Double	IDC
Binary	0.98	0.64	0.70	
Binary_subject	0.96	0.67		0.85
Multi5	0.87	0.34	0.5	
Multi5_subject	0.89	0.37		0.88
Multi10	0.56	0.17	0.35	
Multi10_subject	0.54	0.19		0.55

Table 5: Co-clustering obtains better micro-averaged-precision values on different newsgroup data sets compared to other algorithms.

Figure 8: Binary, Multi5, Multi10 data, from [20]

CLASSIC3 - Support Vector Clustering									
Type	CISI			MEDLINE			CRANFIELD		
	P	R	F1	P	R	F1	P	R	F1
LC	no separation								
GC	64.72%	100%	78.58%	63.11%	100%	77.38%	n.a.n.		
EC	**100**%	**99.8**%	**99.9**%	**100**%	**100**%	**100**%	**100**%	**99.6**%	**99.8**%

	LC	GC	EC
Accuracy	37.50%	64.038%	**99.8**%
Macro-AVG	n/a	n/a	**99.9**%

Details of Support Vector Clustering instances					
Type	Kernel	q	C	softening	# of runs
LC	Laplacian	any	any	any	any
GC	Gaussian	0.527325	0.00256871	any	3
EC	**Exponential**	**1.71498**	**0.00256871**	**1**	**4**

Table 3: From [37]. Support Vector Clustering of the CLASSIC3. The first table shows the Precision (P), Recall (R), F1, Accuracy and Macroaveraging for each class and for each SVC instance. The second table shows the details about the SVC instances. In bold the best results.

Algorithm	Iris Data	Wisconsin Database	Spam Database
SOM	121.5 ± 1.5 (81.0%)	660.5 ± 0.5 (96.7%)	1210 ± 30 (78.9%)
K-Means	133.5 ± 0.5 (89.0%)	656.5 ± 0.5 (96.1%)	1083 ± 153 (70.6%)
Neural Gas	137.5 ± 1.5 (91.7%)	656.5 ± 0.5 (96.1%)	1050 ± 120 (68.4%)
Ng-Jordan Algorithm	126.5 ± 7.5 (84.3%)	652 ± 2 (95.5%)	929 ± 0 (60.6%)
KMC	142 ± 1 (94.7%)	662.5 ± 0.5 (97.0%)	1247 ± 3 (81.3%)

Table 1. SOM, K-Means, Neural Gas, Ng-Jordan algorithm and KMC average performances, in terms of correctly classified points, on Iris, Wisconsin and Spam database. The variances σ of the gaussian Kernel are: 1.1 (Iris Data), 0.9 (Wisconsin Data), 2.0 (Spam Database)

Figure 9: Spambase data, from [11]

References

[1] N. Ancona, R. Maglietta, and E. Stella. On sparsity of data representation in support vector machines. In *Signal and Image Processing*, volume 444, Via Amendola 122/D-I - 70126 Bari, Italy, 2004. Istituto di Studi sui Sistemi Intelligenti per l'Automazione - C.N.R.

[2] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized Maximum Entropy approach to Bregman co-clustering and matrix approximation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, pages 509–514, August 2004.

[3] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized Maximum Entropy approach to Bregman co-clustering and matrix approximation. Technical report, UTCS TR04-24, UT, Austin, 2004.

[4] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized Maximum Entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, August 2007.

[5] R. Basili and A. Moschitti. *Automatic Text Categorization: From Information Retrieval to Support Vector Learning*. Aracne Editrice, 2005.

[6] R. E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[7] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik. A support vector method for hierarchical clustering. In *Fourteenth Annual Conference on Neural Information Processing Systems*, Denver, Colorado, November 2000.

[8] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[9] L. M. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[10] L. Breiman and L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[11] F. Camastra. Kernel methods for clustering. In *Proceedings of 16th Workshop of Italian Neural Network Society (WIRN05)*, Lectures Notes

on Computer Science Series, Vietri sul Mare, Italy, June 2005. Springer-Verlag.

[12] J.-H. Chiang and P.-Y. Hao. A new kernel-based fuzzy clustering approach: support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems*, 11(4):518–527, August 2003.

[13] H. Cho, I. Dhillon, Y. Guan, and S. Sra. Minimum sum squared residue co-clustering of gene expression data. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, pages 114–125, April 2004.

[14] S. Della Pietra, V. Della Pietra, and J. Lafferty. Duality and auxiliary functions for Bregman distances. TR CMU-CS-01-109 TR CMU-CS-01-109, Carnegie Mellon University, 2001.

[15] M. Deodhar, H. Cho, G. Gupta, J. Ghosh, and I. Dhillon. Bregman Bubble Co-clustering. Technical report, Department of Electrical and Computer Engineering, The University of Texas at Austin, October 2007.

[16] P. Dhanalakshmi, S. Ravichandran, and M. Sindhuja. The role of clustering in the field of information retrieval. In *National Conference on research prospects in knowledge mining (NCKM-2008)*, pages 71–77, 2008.

[17] I. Dhillon and Y. Guan. Information theoretic clustering of sparse co-occurrence data. In *Proceedings of The Third IEEE International Conference on Data Mining*, pages 517–520, November 2003.

[18] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. K. R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001. Invited book chapter.

[19] I. S. Dhillon and Y. Guan. Information theoretic clustering of sparse co-occurrence data. Technical report tr-03-39, The University of Texas at Austin, Department of Computer Sciences, September 2003.

[20] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 89–98, 2003.

[21] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer, and G. Dietterich. An efficient boosting algorithm for combining preferences. In *Journal of Machine Learning Research*, pages 170–178. Morgan Kaufmann, 1998.

[22] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95:*

Proceedings of the Second European Conference on Computational Learning Theory, pages 23–37, London, UK, 1995. Springer-Verlag.

[23] B. C. M. Fung, K. Wang, and M. Ester. Hierarchical document clustering. In *Encyclopedia of Data Warehousing and Mining*, volume Volume 1. Idea Group Publishing, July 2005.

[24] E.-H. S. Han and G. Karypis. Centroid-based document classification algorithms: Analysis & experimental results. Technical report, niversity of Minnesota - Computer Science and Engineering, 2000.

[25] M. S. Hansen, K. Sjöstrand, H. Ólafsdóttir, H. B. W. Larsson, M. B. Stegmann, and R. Larsen. Robust pseudo-hierarchical support vector clustering. In B. K. Ersbøll and K. S. Pedersen, editors, *SCIA*, volume 4522 of *Lecture Notes in Computer Science*, pages 808–817. Springer, 2007.

[26] P.-Y. Hao, J.-H. Chiang, and Y.-K. Tu. Hierarchically svm classification based on support vector clustering method and its application to document categorization. *Expert Syst. Appl.*, 33(3):627–635, 2007.

[27] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[28] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.

[29] S. S. Keerthi and D. DeCoste. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *J. Mach. Learn. Res.*, 6:341–361, 2005.

[30] H.-T. Lin. Infinite ensemble learning with support vector machines. Master's thesis, California Institute of Technology, May 2005.

[31] H.-T. Lin. Support vector machinery for infinite ensemble learning. *Journal of Machine Learning Research*, 9(2):285–312, January 2008.

[32] B. C. M, F. Ke, and W. M. Ester. Hierarchical document clustering using frequent itemsets. In *In Proc. SIAM International Conference on Data Mining 2003 (SDM 2003*, 2003.

[33] K. Machová, V. Maták, and P. Bednár. The role of the clustering in the field of information retrieval. In *Fourth Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence (SAMI-2006)*, 2006.

[34] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2007.

[35] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, April 1998.

[36] M. Rosell. Introduction to information retrieval and text clustering. Paper used as introduction text for a Swedish course of Information Retrieval, 2006.

[37] V. Russo. State-of-the-art clustering techniques: Support Vector Methods and Minimum Bregman Information Principle. Master's thesis, Università degli studi di Napoli "Federico II", Corso Umberto I, 80100 Naples, Italy, Febbraio 2008. (Download from `http://thesis.neminis.org/2008/04/03/thesis-and-talk/`).

[38] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[39] I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. In *Twenty-Fourth International Conference on Machine Learning (ICML)*, Corvallis, Oregon, USA, June 2007.

[40] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[41] Wired.com. The petabyte age: Because more isn't just more — more is different (`http://www.wired.com/science/discoveries/magazine/16-07/pb_intro`), June 2008.

[42] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Data Mining and Knowledge Discovery*, pages 515–524. ACM Press, 2002.