Clustering high dimensional data: Examining differences and commonalities between subspace clustering and text clustering - A position paper

Hans-Peter Kriegel
Ludwig-Maximilians-University of Munich
Germany
kriegel@dbs.ifi.lmu.de

Eirini Ntoutsi
Ludwig-Maximilians-University of Munich
Germany
ntoutsi@dbs.ifi.lmu.de

ABSTRACT

The goal of this position paper is to contribute to a clear understanding of the commonalities and differences between subspace clustering and text clustering. Often text data is foisted as an ideal fit for subspace clustering due to its high dimensional nature and sparsity of the data. Indeed, the areas of subspace clustering and text clustering share similar challenges and the same goal, the simultaneous extraction of both clusters and the dimensions where these clusters are defined. However, there are fundamental differences between the two areas w.r.t object feature representation, dimension weighting and incorporation of these weights in the dissimilarity computation. We make an attempt to bridge these two domains in order to facilitate the exchange of ideas and best practices between them.

Keywords

high dimensional data, subspace clustering, text clustering

1. INTRODUCTION

Subspace Clustering is a new research area receiving a lot of attention from the community [23; 18; 19; 8]. In contrast to traditional clustering that partitions objects in the full dimensional feature space, the subspace clustering methods simultaneously partition both objects and dimensions. A subspace cluster is defined in terms of both its members and the dimensions where these members are grouped together. As with traditional full dimensional clustering, subspace clustering is not confined to specific data types or application domains. Related work often mentions *text data* as a candidate application for subspace clustering [23; 8] due to the high dimensionality and sparsity of the text data [26]. Typically, documents are represented as vectors in a very high dimensional space where the entries of this vector represent words appearing in the document collection.

Text data have been studied extensively in the fields of Information Retrieval and Text Mining. The *text clustering* domain [4] is among the most important ones with a lot of applications like organization, summarization and indexing of document content. Nowadays, it is a very active research field due to the abundance of textual data.

Both subspace clustering and text clustering domains face the challenge of high dimensionality and aim at extracting both clusters and the dimensions upon which these clusters are defined. To this end, both domains evaluate the importance of the dimensions for the clustering task and appropriately incorporate this importance in the distance function and consequently, in the clustering algorithm. Besides the common goal and the similar approach though, there are fundamental differences between the two domains. To the best of our knowledge, no work that elaborates on how the two areas are related has been proposed so far. Our goal of this work is to serve as a point for fruitful exchange of ideas and techniques between the two domains.

The rest of the paper is organized as follows: A short overview of subspace clustering and text clustering domains is presented in Sections 2 and 3, respectively. The differences and commonalities between the two domains are discussed in Section 4 and refer to the following aspects: object-feature representation (Section 4.1), dimension weighting (Section 4.2) and incorporating dimension weights in the similarity/distance function (Section 4.3). In Section 5, we overview a few existing approaches that make use of concepts from both domains. Section 6 concludes our work.

2. SUBSPACE CLUSTERING IN A NUTSHELL

The area of subspace clustering has lately emerged as a solution to the problem of *high dimensional* data, as it is difficult to find meaningful clusters in hundreds or even thousands of dimensions. Different features might be relevant for different clusters and therefore, the goal of subspace clustering is to find both the cluster members and the dimensions upon which these members form a cluster.

This is in contrast to traditional clustering that searches for clusters in the full dimensional feature space [15; 13]. Also, this is different from global dimensionality reduction techniques like PCA [16] that reduce the dimensionality of the feature space and search for clusters in the reduced (though still full) dimensional space. Several subspace clustering methods have been proposed in the literature so far (for a comprehensive overview of the area see e.g., [23], [18], [21]).

A crucial point in subspace clustering is the decision about the relevant dimensions that should be considered for clustering. Given a d-dimensional data set, the number of possible subspaces is 2^d-1, therefore it is infeasible to examine all possible subspaces for clusters. The solution is to efficiently navigate through the search space of all possible subspaces; to this end, two different approaches have been proposed in

the literature [18]:

- *Bottom-up approaches*: They start with 1–D subspaces (i.e., single dimensions) and iteratively merge lower dimensional subspaces in order to compute higher dimensional subspaces. The downward-closure property of Apriori [6] is usually employed to prune non-appropriate subspaces. The key in this category is the merging procedure, i.e., how low dimensional subspaces would be merged to yield higher dimensional subspaces. Once the appropriate subspaces are detected, the clustering algorithm is applied over each subspace similarly to full dimensional clustering.

  To this category belong methods like CLIQUE [5], MAFIA [22] and SUBCLU [17]. These methods result in overlapping clusters; an object might be assigned to more than one clusters defined in different subspaces of the original feature space.

- *Top-down approaches*: They start searching in the full dimensional space and iteratively learn the "correct" subspace for each point or cluster. The key in this category is how to learn the "correct" subspace for a point or cluster.

  Representatives of this category are methods like PROCLUS [3], DOC[25], COSA[11] and PreDeCon [9].

Top-down subspace clustering methods are closer to the text clustering methods since they simultaneously search for the best non-overlapping partitioning of the data and the best subspace for each partition. Therefore, we refer to the *top-down subspace clustering* methods from now on.

3. TEXT CLUSTERING IN A NUTSHELL

The goal of text clustering is to organize documents into clusters of similar content, the so-called topics. Documents are typically represented in terms of their component words, through the *vector space model* [27]. Usually, no information about the order of the words is considered in this model and thus this model is also known as the *bag of words* representation model. Although more elaborate representations have been proposed such as multi-word terms and N-Grams [28], the vector space model remains the most popular one. This representation though results in a high dimensional feature space, where features correspond to words appearing in the document collection. Moreover, only a small subset of all words appearing in the collection appears in each document. As a result, the document representation is *very sparse*.

Typically, before applying any mining technique in the text collection, preprocessing takes place in order to remove non–informative words. Preprocessing consists of several steps [7]:

- **Filtering:** Special characters and punctuation are removed since they are considered not to hold any discriminative power.
- **Stemming:** The words are reduced to their base form or stem or root with the goal of eliminating multiple occurrences of a word (e.g., the words "fish", "fishing", "fisher", "fished" are all derived from the same root "fish"). Stemming is language dependent. The Porter's algorithm [24] is the best known stemmer for the English language.
- **Stopword removal:** A stopword is a word which is not thought to convey any meaning as a dimension in the vector space (i.e., without context), e.g., the words "the", "and". Usually the document words are compared with respect to a known list of stopwords and the detected stopwords are removed from the document.
- **Rare words removal:** Rare words, i.e., words that appear in very few documents are usually removed since they are considered not to capture much information about some category of documents [30]. A threshold on the number of documents is used that determines whether a word is rare and should be removed from the feature space as an outlier.

Although preprocessing results in some sort of dimensionality reduction, the number of dimensions remains very high (hundreds or thousands of keywords[1]). To deal with this problem several dimensionality reduction techniques have been proposed which can be distinguished into feature transformation and feature selection techniques. The *feature transformation* techniques try to reduce the dimensionality to a fewer new dimensions which are combinations of the original dimensions. In this category belongs methods like Principal Component Analysis (PCA) [16] and Latent Semantic Indexing (LSI) [20]. Because they are global methods though, they might be problematic for topics defined upon different keywords. *Feature selection* methods aim at removing dimensions which seem irrelevant for modeling, e.g., words appearing very often in a collection or very rarely, aka outliers. Nevertheless, the resulting feature space is still high dimensional and therefore dimensions' weighting takes place as we will explain in the following sections.

Typical examples of distance-based text clustering algorithms are hierarchical clustering and k-Means [4], while cosine similarity is the commonly used similarity function.

4. DIFFERENCES BETWEEN SUBSPACE CLUSTERING AND TEXT CLUSTERING

In the previous sections, we presented a short introduction to the areas of subspace clustering and text clustering. Hereafter, we show how the two areas are related by pointing out their commonalities and differences.

As already mentioned, both areas deal with *high dimensional* data: *Subspace clustering* targets high dimensional data, though not focusing on specific application domains or data types. The data instances are described in terms of all dimensions, in the full (high) dimensional feature space. The vast majority of the algorithms does not deal with missing values, though recently fault tolerant subspace clustering [12] has been proposed to accommodate data with missing values. From a *text clustering* perspective, the keywords of a document comprise the dimensions of the feature space and documents are points or vectors in this space. There are hundreds or thousands of keywords (thus, high dimensional space) and the vectors describing each document are very sparse (most of the entries are null). A null entry in this case might mean that the corresponding keyword is irrelevant (e.g., consider the word "Ukraine" in a document

[1]The terms "word", "keyword" "term" are used interchangeably.

about the importance of vegetables in our daily diet) or the corresponding word might be missing because it is implied by the context (e.g., in a document about "Greece" the word "country" might not be reported since it is known that Greece is a country).

Note also that both areas share the same goal, namely the *simultaneous partitioning* of the data points and the dimensions (points/ instances and dimensions/ features, respectively according to the subspace clustering terminology; documents and keywords, respectively according to the text clustering terminology). In both cases, the notion of a cluster includes, except for the cluster members, and the dimensions where these members are similar enough to form the cluster. In subspace clustering, these clusters are known as subspace clusters, whereas in text clustering as topics.

The problem of simultaneously partitioning both the data and the dimensions is tackled in a similar way by both subspace clustering and text clustering areas. In particular, the solution involves an appropriate weighting of the dimensions according to their relevance for the clustering task and the incorporation of these weights in the dissimilarity function[2] and consequently, in the clustering algorithm.

We present the commonalities and differences between the two domains with respect to the following aspects: (i) the object–feature representation (Section 4.1), (ii) the weighting of the dimensions (Section 4.2) and (iii) the incorporation of dimensions' weighting in the dissimilarity functions and consequently, in the clustering algorithms (Section 4.3).

4.1 Object–feature representation differences

Let $\mathcal{D} = \{p_1, p_2, ..., p_n\}$ be a dataset of n objects and let A=(A_1, A_2, ..., A_d) be the d-dimensional feature space. Let V be a $|\mathcal{D}| \times |A|$ matrix, referred hereafter as the *object–feature value matrix*, where the rows are the objects and the columns are the dimensions. Each entry $v_{i,j}$ in this matrix corresponds to the value of the object $p_i \in \mathcal{D}$ in the dimension $A_j \in A$. We explain hereafter what these values are for each domain.

4.1.1 Object–feature representation in subspace clustering

In subspace clustering, each object is considered to have values for all dimensions[3]. So, the construction of the object–feature value matrix V is rather straightforward. The rows correspond to the objects in the database and the columns correspond to the dimensions. Each entry $v_{i,j}$ in the matrix contains the value of the object $p_i \in \mathcal{D}$ in dimension $A_j \in A$. The original values of the dimensions may refer to different scales of reference. To prevent attributes with initially large ranges (e.g., salary) from outweighing attributes with initial smaller ranges (e.g., age), a *normalization* process takes place prior to clustering. In this way, different dimensions can be compared meaningfully.

4.1.2 Object–feature representation in text clustering

In text clustering, the database D corresponds to a collection of documents and the dimensions A correspond to the distinct keywords in this collection. We assume that the preprocessing step (c.f., Section 3) has been already applied.

[2]We use the term "dissimilarity" to refer to either distance or similarity function.

[3]Recently, subspace clustering over data with missing values has been considered [12].

The matrix V is the result of the vector space model representation, called *document–term matrix* in this settings. Each entry $v_{i,j}$ in the matrix corresponds to the value of keyword $A_j \in A$ in document $p_i \in \mathcal{D}$. Usually $v_{i,j}$ equals the number of times that keyword A_j appears in document p_i. To prevent biasing towards longer documents, the number of occurrences of a keyword in a document is *normalized* with respect to the total number of keyword occurrences in the document, resulting in the so-called term frequency:

***Definition** 1.* Term Frequency (TF)
The term frequency of a term/keyword $A_j \in A$ in a document $p_i \in \mathcal{D}$ is defined as follows:

$$TF_{j,i} = \frac{n_{j,i}}{\sum_k n_{k,i}}$$

where the numerator represents the number of occurrences of keyword A_j in document p_i and the denominator represents the occurrences of all keywords $A_k \in A$ in p_i.

The TF score expresses how important a keyword is within a document. Its values lie in the $[0, 1]$ range with larger values indicating more important keywords. Hereafter, we consider the entries of matrix V to be the TF values of the keywords in the documents of the collection. Note that since not all keywords in the collection appear in each single document, there are a lot of null entries in this matrix corresponding to non-appearing words in a document.

4.1.3 Discussion on object-feature representation differences

There is a crucial difference in the object-feature representation of the two domains. In subspace clustering, there are no semantics on the values of the dimensions and all values are considered to be of the same importance. On the contrary, in text clustering greater values indicate more important dimensions/keywords. For example, consider a keyword A_1 appearing in two documents p_1, p_2 with term frequencies $0.8, 0.2$, respectively. Since $0.8 > 0.2$, A_1 is considered to be more important for document p_1 compared to document p_2. Such a value differentiation though does not take place in the subspace clustering domain.

Although, as already mentioned, recently subspace clustering over missing data has been proposed [12], missing data are conceptually different from non–appearing words in a document. A missing value in subspace clustering indicates a value that is unknown rather than a value that is not important for the description of a document or redundant w.r.t. already existing words in the document (as it is usually the case for non appearing words in a document).

4.2 Dimension weighting differences

Both subspace clustering and text clustering domains rely on some notion of weighting for the different dimensions based on their importance for the clustering task.

In subspace clustering, the important dimensions are learned either for an instance/object (*instance–based approaches*) or for a cluster (*cluster–based approaches*). Representative weighting schemes for both approaches are presented in Section *4.2.1*. In text clustering, the most well known weighting schema is *inverse document frequency (IDF)* described in Section *4.2.2*.

We can model the dimension weighting in terms of a matrix W referred to hereafter as the *object–feature weight matrix*.

In the general case, W is a $|\mathcal{D}| \times |A|$ matrix where the rows correspond to objects and the columns correspond to dimensions. Each entry $w_{i,j}$ in this matrix represents the weight (or importance) of the dimension $A_j \in A$ for the object $p_i \in \mathcal{D}$. We explain hereafter how these entries are filled for each domain.

4.2.1 Dimension weighting in subspace clustering

The importance of a dimension is evaluated either with respect to some instances (instance–based approaches) or with respect to some cluster (cluster–based approaches). Both approaches rely on the so called "locality assumption" according to which, the subspace preference for an object or a cluster can be learned from its local neighborhood in the full dimensional space. We describe each case below.

(i) Dimension weighting in instance–based approaches
The preferred dimensions subspace is defined *per instance* and is learned by evaluating the local neighborhood of the instance/object in the full dimensional feature space. The definitions and details below are from the algorithm PreDeCon [9].

Locality of an instance: For an object $p \in \mathcal{D}$, its locality or neighborhood ($\mathcal{N}_\varepsilon(p)$) consists of all objects that fall within distance ϵ from p in the full dimensional space.

Preferred dimensions of an instance: Roughly speaking, an object prefers a dimension if it builds "compact" neighborhoods, i.e., neighborhoods of small variance, across this dimension. The variance in the neighborhood of p along some dimension A_j is defined as follows:

Definition 2. Variance along a dimension
Let $p \in \mathcal{D}$ and $\varepsilon \in \mathbb{R}$. The *variance in the neighborhood of* p $\mathcal{N}_\varepsilon(p)$ *along a dimension* $A_j \in \mathcal{A}$ is given by:

$$\text{VAR}_{A_j}(\mathcal{N}_\varepsilon(p)) = \frac{\sum_{q \in \mathcal{N}_\varepsilon(p)} (dist_{A_j}(p,q))^2}{|\mathcal{N}_\varepsilon(p)|}$$

where $dist_{A_j}(p,q)$ is the distance of p, q in dimension A_j.

A small variance, with respect to a given variance threshold δ, indicates a preferable dimension. For each $p \in \mathcal{D}$, its *subspace preference vector* $\bar{\mathbf{w}}_p$ is built [9] which distinguishes between preferable and non-preferable dimensions.

Definition 3. Subspace preference vector
Let $p \in \mathcal{D}$, $\delta \in \mathbb{R}$ and $\kappa \in \mathbb{R}$ be a constant with $\kappa \gg 1$. The *subspace preference vector* of p is defined as:

$$\bar{\mathbf{w}}_p = (w_1, w_2, ...w_d)$$

where:

$$w_i = \begin{cases} 1 & \text{if} \quad \text{VAR}_{A_i}(\mathcal{N}_\varepsilon(p)) > \delta \\ \kappa & \text{if} \quad \text{VAR}_{A_i}(\mathcal{N}_\varepsilon(p)) \leq \delta \end{cases}$$

The values of the subspace preference vector are the entries of the object–feature weight matrix W: k-value entries indicate preferable dimensions while 1-value entries indicate non-preferable ones.

(ii) Dimension weighting in cluster–based approaches
The subspace of the preferred dimensions is defined *per cluster* and is learned by evaluating the local neighborhood of the cluster in the full dimensional space. The number of clusters k is given as input to the algorithms of this category. The definitions and details below are from the algorithm PROCLUS [3]. Each cluster (called *projected cluster*) is represented by its medoid and it is assigned a subspace of preferred[4] dimensions. Let $C = \{c_1, c_2, \ldots, c_k\}$ be a set of k medoids (for more details on the initial selection and refinement of this set, please refer to [3]).

Locality of a cluster: The locality L_i of a cluster $c_i \in C$ is defined as the set of objects in $\mathcal{D}$ that are within distance δ_i from its medoid c_i. The distance is computed in the full dimensional space, whereas the distance threshold δ_i is the minimum distance of c_i from any other medoid $c_j \in C$, i.e., $\delta_i = min\{dist(c_i, c_j)\}$, $i \neq j$.

Preferred dimensions of a cluster: For each medoid c_i, the average distance between c_i and the objects in its locality L_i is computed along each dimension $A_j \in A$, denoted by $X_{i,j}$. If this value is as small as possible w.r.t. the statistical expectation, then A_j is a preferred dimension for cluster c_i. The statistical expectation is evaluated in terms of the mean Y_i and the standard deviation σ_i. The value $Z_{i,j} = \frac{X_{i,j} - Y_i}{\sigma_i}$ is computed which indicates how the average distance between cluster c_i and its locality L_i in dimension A_j is related to the average distance in all dimensions. The lowest $Z_{i,j}$ values are picked leading to a total of $k * l$ dimensions, where l is the average dimensionality per cluster given as input to the algorithm.

After the weighting, each cluster in C is assigned a subspace of preferred dimensions. Usually a bitvector of all dimensions is used to model the preferences and to distinguish between preferred (value 1) and non preferred (value 0) dimensions for a cluster [2].

The objects assigned to a cluster "inherit" the subspace preferences of the cluster, therefore each object can be assumed to have a bitvector of dimension preferences. This way, we can fill the entries of the object–feature weight matrix W with values 1 and 0 indicating preferred and non–preferred, respectively, dimensions for an object. Note that this way the objects belonging to the same cluster, would have the same bitvectors of subspace preferences.

4.2.2 Dimension weighting in text clustering

In text clustering, the importance of a keyword is evaluated in terms of the whole collection of documents, rather than per document or per topic/cluster. Usually, the Inverse Document Frequency (IDF) is employed towards this end.

Definition 4. Inverse Document Frequency (IDF)
The inverse document frequency of a keyword A_j in a collection of documents D is given by:

$$IDF_{A_j} = \frac{|D|}{|\{p_i \in \mathcal{D} : A_j \in p_i\}|}$$

where the denominator represents the number of documents in D which contain the specific keyword/dimension A_j and $|D|$ is the cardinality of the collection.

The basic intuition behind IDF is that common keywords (i.e., keywords appearing very often in the collection) have no discriminative power and thus, they are assigned a small

[4]We use both terms "preferred" and "projected" to describe a dimension that is important.

IDF score. Larger IDF scores indicate more discriminative keywords.
Using the IDF scores of the keywords we can fill the entries of the object–feature weight matrix W. Note that since IDF is defined per keyword, each column of the matrix corresponding to a single keyword would be filled with the same IDF value.

4.2.3 Discussion on dimension weighting differences

Although, the weighting of the dimensions is performed in both subspace clustering and text clustering domains, there are core differences in the weighting schemes.
In subspace clustering, the dimension weighting is *local* relying on the neighborhood of each object or cluster. In text clustering, the weighting of the keywords is *global* and is based upon the whole collection of documents.
Also, the dimension weights in subspace clustering act mainly as a filter in order to distinguish between preferred and non–preferred dimensions. For example, PreDeCon [9] uses the dimension weights κ and 1 to model preferred and non-preferred dimensions, respectively. Similarly, PROCLUS [3] employs two dimension weights, 0 and 1 with 1 indicating a preferred dimension. That is, the actual weights of the dimensions are not considered in subspace clustering, but rather the information about which dimension is preferred or not. This is in contrast to IDF weighting in text.

4.3 Incorporating dimension weighting in similarity/distance computation differences

The weighting of the dimensions is incorporated in the dissimilarity function in both subpsace clustering and text clustering domains. We model this contribution in terms of a generic dimension–weighted dissimilarity function.

***Definition** 5.* Dimension–weighted dissimilarity
Let $p, q \in \mathcal{D}$. We denote by $V[p, A_j]$ ($V[q, A_j]$) the value of object p (q, respectively) with respect to dimension A_j and by $W[p, A_j]$ ($W[q, A_j]$) the importance of A_j for the object p (q, respectively). The dissimilarity between p and q is a combination of their values and dimensions' weights:

$$diss(p, q) = \textsc{Aggr}_{A_j \in A}\ \text{f}(V[p, A_j], V[q, A_j], W[p, A_j], W[q, A_j])$$

The function f() combines the value and weights in each dimension. The total score is an aggregation of the corresponding dimensions' scores, through some aggregation function Aggr().

We already discussed what the object values and dimension weights stand for in each domain and how the object–feature value matrix V and the object–feature weight matrix W are filled. We explain hereafter how the $diss()$ function is instantiated per domain.

4.3.1 Dissimilarity computation in subspace clustering

We distinguish between instance–based and cluster–based approaches, as with the weighting of dimensions case (cf., Section 4.2).

(i) Dissimilarity in instance–based approaches We adapt the distance function of PreDeCon [9] to the generic $diss()$ function notation.

***Definition** 6.* Preference weighted distance
Let $p, q \in \mathcal{D}$, then their preference weighted distance is given by:

$$dist_p(p, q) = \sqrt{\sum_{A_j \in A} \frac{1}{W[p, A_j]} \cdot (V[p, A_j] - V[q, A_j])^2}$$

The above formula considers only the preferences of p. The symmetric version is:

$$dist(p, q) = max(dist_p(p, q), dist_q(p, q))$$

where $V[p, A_j]$ ($V[q, A_j]$) is the value of object p (q, respectively) in dimension A_j and $W[p, A_j]$ ($W[q, A_j]$) is the weight of A_j for point p (q, respectively).

Note, that the dimension weights are either 1 or κ ($\kappa >> 1$). Therefore, the similarity function weights attributes with low variance (dimension weight κ) considerably lower (by a factor $1/\kappa$) than attributes with a high variance (dimension weight 1).

(ii) Dissimilarity in cluster–based approaches We adapt the distance function of PROCLUS to the generic $diss()$ function notation.

***Definition** 7.* Projected distance
Let $p, q \in \mathcal{D}$ with q being the medoid of a cluster. The projected distance of p with respect to the medoid q is given by:

$$dist(p, q) = \frac{\sum_{A_j \in A} W[q, A_j] * |V[p, A_j] - V[q, A_j]|}{|\{A_j : W[q, A_j] = 1\}|}$$

$W[q, A_j]$ is the importance of dimension d for the cluster represented by the medoid q. Note that in this case, weights take values in $\{0, 1\}$. That is, the non important dimensions are not considered at all during distance computation and the important ones are equally taken into account.

4.3.2 Dissimilarity computation in text clustering

$TF \times IDF$ is the most common schema for combining keyword weights and their values across different documents. According to this schema, the value of a keyword in a document (TF) is multiplied by the importance of the keyword in the whole collection (IDF).

***Definition** 8.* TF×IDF score
Let $p \in \mathcal{D}$ be a document and let $A_j \in A$ be one of its keywords. The TF×IDF score of A_j in p is given by:

$$(TF \times IDF)_{p,A_j} = TF_{p,A_j} \times IDF_{A_j}$$

Typically, the dissimilarity function is applied upon the TFIDF values of the documents. A commonly used dissimilarity function is the cosine similarity that finds the cosine of the angle between the two documents. It becomes 1 if the documents are identical and 0 if there is nothing in common between them (i.e., the vectors are orthogonal to each other).
The cosine similarity can be re-written in terms of the generic formula $diss()$ as follows:

***Definition** 9.* Cosine similarity
Let $p, q \in \mathcal{D}$ be two documents. Their similarity is defined as:

$$cos(p,q) = \frac{\sum_{A_j \in A} V[p,A_j] * W[p,A_j] * V[q,A_j] * W[q,A_j]}{\sqrt{\sum_{A_j \in A} (V[p,A_j] * W[p,A_j])^2} * \sqrt{\sum_{A_j \in A} (V[q,A_j] * W[q,A_j])^2}}$$

Note that the weight of a keyword A_j is defined over the whole collection $\mathcal{D}$ thus, documents p and q share the same weight for this keyword, i.e.,$W[p, A_j] = W[q, A_j] = IDF_{A_j}$.

4.3.3 Discussion on incorporation of dimension weighting in distance function differences

Although, both domains consider objects' values and dimensions' weights for dissimilarity computation, there is a clear difference between the two domains. The difference lies in the fact that in text clustering the value of a dimension/ keyword in a document, i.e.,TF, also expresses some notion of importance for the keyword in the document. This is in contrast to subspace clustering where all values are treated similarly and there is no importance discrimination.

This fact is also reflected in the chosen dissimilarity functions. In subspace clustering, where the values carry no semantics on their importance, the value difference in each dimension (e.g.,, absolute as in PROCLUS [3] or squared as in PreDeCon [9]) is considered. In text clustering, though, the widely used cosine similarity function multiplies the actual object values at each dimension, so that higher values result in higher scores. For example, if we consider two objects with values 0.8 and 0.6 for a specific dimension (case 1), the contribution of this dimension to the dissimilarity score will be $0.8 - 0.6 = 0.2$ for PROCLUS and $(0.8 - 0.6)^2 = 0.04$ for PreDeCon, whereas for the cosine similarity it will be $0.8 * 0.6 = 0.48$. For two other objects, with values $0.4, 0.2$ in the same dimension (case 2), the result would be the same for PROCLUS and PreDeCon since they rely on their value difference which is again 0.2, however it will be different for cosine similarity; the contribution of this dimension this time would be: $0.4 * 0.2 = 0.08$ counting for the fact that the actual object values are lower in case 2 than in case 1.

Also, in subspace clustering the dimension weighting predominantly indicates whether the corresponding object values should be considered or not. For example, in PROCLUS [3] only projected dimensions (having weight 1) are considered during dissimilarity assessment and there is no special weighting per preferred dimension. In PreDeCon [9] also, the value differences in the preferred dimensions (those having weight κ) are down-weighted by $1/\kappa$, but again this holds for all preferred dimensions. On the contrary, in text clustering the actual weights of the dimensions as expressed by their IDF values over the whole collection of documents are considered and contribute proportionally to the dissimilarity score.

5. TOWARDS COMBINING THE BEST OF BOTH WORLDS

In the previous section, we elaborated on the differences and commonalities between subspace clustering and text clustering domains with respect to data representation, dimension weighting and incorporation of these weights in the dissimilarity functions. In this section, we overview approaches that make use of concepts from both domains.

In [1], the authors study the problem of semi-supervised text classification and use clustering to create the set of categories for the classification of documents. To improve clustering quality, they use a modified version of K–Means where they iteratively refine both the clusters and the dimensions inside each cluster. In particular, at each iteration they filter out some of the words of the feature space ensuring that only words that frequently occur within the cluster are used for the assignment process. The number of words, i.e.,projected dimensions for each cluster, is reduced at each iteration by a geometric factor. In their experiments, they started with a projected dimensionality of 500 words which was finally reduced to 200 words. This work introduces subspace clustering concepts to text clustering, in particular, the local (within each cluster) weighting of the words and selection of the most prominent ones for cluster centroid representation. The proposed algorithm resembles PROCLUS [3] (c.f., Section *4.2.1*), however the distance function and the selection of projected dimensions at each iteration is adopted to the text domain. In particular, cosine function is used for dissimilarity assessment and the selection of projected dimensions is done on the basis of their occurrences in each cluster.

In [14], the authors propose a modification of W–k–Means [10], that calculates cluster specific weights for each feature during the clustering process, for text documents. Totally $m \times k$ weights are produced by the algorithm where m is the number of features and k the number of clusters. The initial weights are randomly generated and refined during the iteration phase, similarly to K–Means. Based on these weights, the subset of keywords that can serve as cluster label can be identified. In their experiments, the proposed subspace method performed better than standard K–Means and Bisecting–KMeans [29].

There are also methods that extract an initial clustering with respect to the whole feature space of documents (usually through $TF \times IDF$ weighting) and refine the clustering result by applying clustering again inside each extracted cluster but using a refined feature space. For example, in [31] the authors propose the extraction and maintenance of a two level hierarchy of global and local topics: the global topics are extracted by applying clustering over the whole collection of documents in the feature space derived from the whole collection through $TF \times IDF$. To extract the local topics, clustering is applied again inside each global cluster using the cluster population to generate the new cluster specific feature space (the IDF scores are based on the cluster population, rather than on the whole collection), instead of using the generic feature space extracted from the whole dataset. This way, the feature space is refined per cluster.

Discussion Although there are some methods for text clustering that make use of subspace clustering concepts, like [1], [14] and methods that refine the feature space through global and local weighting like [31], there are still many things that the two domains could exchange and benefit from each other, like for example a concurrent incorporation of both global and local weighing of features in the clustering process. There is a bunch of subspace clustering algorithms in the literature, a thorough report on their performance over text data would be very useful as a starting point. Of course, the weighting of the dimensions and the distance function should respect the peculiarities of the text data as discussed in Section 4. There are also other fields which resemble the semantics of the data values in text; for example, in recommendation applications higher ratings indicate more desired items,e.g.,movies or restaurants. Subspace clustering is relevant to such kind of data too, since groups of users with different item preferences might exist, however as with text, the data semantics should be also taken into account.

6. CONCLUSIONS

We outlined the differences and commonalities between subspace clustering and text clustering and we argued that text data is not a straightforward application domain for subspace clustering as it is often suggested in the literature. Although both domains share the same goal, the concurrent extraction of clusters and dimensions where these clusters are formed, and deal with similar challenges, high dimensional and sparse data, there are key differences between the two domains which we summarize bellow:

- The object values in text clustering also reflect the importance of a keyword within a document (TF scores).
- Missing data in subspace clustering imply unknown data, whereas in text clustering a non-appearing keyword indicates words that are not important or are probably redundant for the description of the document.
- Both domains use some notion of dimension weighting to count for the different importance of the dimensions for the clustering task.
- In subspace clustering, dimension weighting is local and is derived with respect to the local neighborhood of a point or a cluster.
- In text clustering, dimension weighting is global and is defined with respect to the whole collection of documents.
- Dimension weighting in subspace clustering mainly filters important from non-important dimensions by assigning the same weight to all important/non-important dimensions.
- In text clustering, the actual weights of the dimensions are used in the dissimilarity function.
- The dissimilarity function in text clustering also considers the importance of a keyword within a document (TF scores), so as more important keywords are given higher scores compared to less important ones.

Our goal of this work is to point out the differences between the two domains and show their commonalities. Text clustering is an established area of research with a bulk of applications and an increased interest nowadays due to the web and the digitization of information. On the other hand, the subspace clustering area grows fast as high dimensionality comprises one of the basic features of nowadays data. A few works combine concepts from both domains and could serve as a starting point for further exchange of ideas and best practices between these domains that would be fruitful for their further development.

7. ACKNOWLEDGMENTS

The authors would like to thank Arthur Zimek for useful comments and suggestions to improve the quality of the paper.

8. REFERENCES

[1] C. Aggarwal, S. Gates, and P. Yu. On using partial supervision for text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 16(2):245–255, Feb 2004.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for projected clustering of high dimensional data streams. In *VLDB*, pages 852–863, 2004.

[3] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA*, 1999.

[4] C. C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pages 77–128. Springer, 2012.

[5] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA*, 1998.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[7] N. O. Andrews and E. A. Fox. Recent developments in document clustering. Technical report, Computer Science, Virginia Tech, 2007.

[8] I. Assent. Clustering high dimensional data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):340–350, 2012.

[9] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM), Brighton, UK*, pages 27–34, 2004.

[10] E. Y. Chan, W. K. Ching, M. K. Ng, and J. Z. Huang. An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognition*, 37(5):943–952, May 2004.

[11] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(4):825–849, 2004.

[12] S. Günnemann, E. Müller, S. Raubach, and T. Seidl. Flexible fault tolerant subspace clustering for data with missing values. In D. J. Cook, J. Pei, W. Wang, O. R. Zaïane, and X. Wu, editors, *ICDM*, pages 231–240. IEEE, 2011.

[13] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2nd edition, 2006.

[14] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):657–668, 2005.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computer Surveys*, 31:264–323, 1999.

[16] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[17] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Lake Buena Vista, FL*, 2004.

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

[19] H.-P. Kriegel, E. Ntoutsi, M. Spiliopoulou, G. Tsoumakas, and A. Zimek. Mining complex dynamic data. Tutorial at the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Athens, Greece., 2011.

[20] T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch. *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

[21] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. In *Proceedings of the 35nd International Conference on Very Large Data Bases (VLDB), Lyon, France*, 2009.

[22] H. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive data sets. In *Proceedings of the 1st SIAM International Conference on Data Mining (SDM), Chicago, IL*, 2001.

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

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

[25] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI*, 2002.

[26] M. Radovanovic, A. Nanopoulos, and M. Ivanovic. On the existence of obstinate results in vector space models. In *SIGIR*, pages 186–193, 2010.

[27] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[28] M. Shafiei, S. Wang, R. Zhang, E. Milios, B. Tang, J. Tougas, and R. Spiteri. A systematic study of document representation and dimension reduction for text clustering. Technical Report CS-2006-05, Faculty of Computer Science, Dalhousie University, Canada, July 2006.

[29] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[30] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.

[31] M. Zimmermann, E. Ntoutsi, and M. Spiliopoulou. Extracting opinionated (sub)features from a stream of product reviews. In *Discovery Science - 16th International Conference (DS), Singapore*, 2013.