Volume 15, Number 2
December 2013

Published by the Association for Computing Machinery
Special Interest Group on Knowledge Discovery and Data Mining

SIG**KDD** *explorations*

TABLE OF CONTENTS

Association for
Computing Machinery

Advancing Computing as a Science & Profession

http://www.kdd.org/explorations/

About *SIGKDD Explorations*

Explorations is published twice yearly, in June/July and December/January each year. After the first two volumes, frequency may increase to quarterly. The newsletter is distributed in hardcopy form to all members of the ACM SIGKDD. It is also sent to ACM's network of libraries. Additionally, issues are published on the web and are free to the general public (http://www.acm.org/sigkdd/explorations/).

Our goal is to make *SIGKDD Explorations* an informative, rapid means of publication and a dynamic forum for communication with the Knowledge Discovery and Data Mining community. SIGKDD membership is growing at a very fast pace, and with KDD being a multi-disciplinary field, we hope that *Explorations* will facilitate its fusion and enhance the sense of community. Submissions will be reviewed by the editor and/or associate and guest editors as appropriate. We are particularly interested in short research and survey articles on various aspects of data mining and KDD. *Explorations* is also a forum for publishing position papers, controversial positions, challenges to the community, product reviews, book reviews, news items and other items of interest to the field. Please see:

http://www.acm.org/sigkdd/explorations/instructions.htm

Advertiser Information:

Explorations accepts advertisements related to data mining and KDD, including company, book, vendor, and service advertisements. For rates and instructions on submitting an ad, please see:

http://www.acm.org/sigkdd/explorations/instructions.htm#advertise

Notice to Contributing Authors of SIGKDD Explorations:

By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library
- to allow users to copy and distribute the article for noncommercial, educational or research purposes.

However, as a contributing author, you retain the copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.

Notice to Past Authors of ACM-Published Articles:

ACM intends to create a complete electronic archive of all articles and/or other material previously published by ACM. If you have written work that was previously published by ACM in any journal or conference proceedings prior to 1978, or any SIG newsletter at any time, and you do NOT want this work to appear in the ACM Digital Library, please inform you respective editors and permissions@acm.org, stating the title of the work, the author(s), and where and when published.

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:

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

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:

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

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:

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

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 $\textsc{Aggr}()$.

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.

Mining Text and Social Streams: A Review

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY
charu@us.ibm.com

ABSTRACT

The large amount of text data which are continuously produced over time in a variety of large scale applications such as social networks results in massive streams of data. Typically massive text streams are created by very large scale interactions of individuals, or by structured creations of particular kinds of content by dedicated organizations. An example in the latter category would be the massive text streams created by news-wire services. Such text streams provide unprecedented challenges to data mining algorithms from an efficiency perspective. In this paper, we review text stream mining algorithms for a wide variety of problems in data mining such as clustering, classification and topic modeling. A recent challenge arises in the context of *social streams*, which are generated by large social networks such as *Twitter*. We also discuss a number of future challenges in this area of research.

1. INTRODUCTION

Text streams have become ubiquitous in recent years because of a wide variety of applications in social networks, news collection, and other forms of activity which result in the continuous creation of massive streams. Some specific examples of applications which create text streams are as follows:

- In social networks, users continuously communicate with one another with the use of text messages. This results in massive volumes of text streams which can be leveraged for a variety of mining and search purposes in the social network. This is because the text messages are reflective of user interests. A similar observation applies to chat and email networks.
- Many news aggregator services [1] may receive large volumes of news articles continuously over time. Such articles are often longer and more well structured than the kinds of messages which are seen in social, chat, or email networks.
- Many web crawlers may collect a large volume of documents from networks in a small time frame. In many cases, such documents are restricted to those which have been modified in a small time frame. This naturally results in a stream of modified documents.

[1] An example would be the *Google News* service.

Text streams create a huge challenge from the perspective of a wide variety of mining applications, because of the massive volume of the data which must be processed in online fashion. Data streams have been studied extensively in recent years not just in the text domain, but also in the context of a wide variety of multi-dimensional applications such as numerical and categorical data. A detailed discussion of mining algorithms for stream data may be found in [3]. While many of the techniques proposed for multi-dimensional data [3] can be generalized to text data at the high level, the details can be quite different because of the very different format and lack of structure of text data. Since stream mining techniques are generally dependent upon summarization, it follows that methods for online summarization need to be designed which work well for the unstructured nature of text data.

In the case of multi-dimensional and time-series data, such summarization often takes the form of methods such as histograms, wavelets, and sketches which can be used to create a structured summary of the underlying data [3]. However, the unstructured nature of text makes the use of such summaries quite challenging. While sketches have been used to some effect in the text domain [27], it has generally been difficult to generalize wavelet and histogram methods to the text domain. As we will see later in this section, the summarization methods designed for text streaming problems may vary a lot, and may often need to be tailored to the problem at hand. One of the goals of this paper is to provide a broad spectrum of the different methods which are used for text mining, which can provide an overview of the tools which can be most effectively used for the text stream scenario. We will also present the future challenges and research directions associated with text stream mining.

This paper is organized as follows. In section 2, we will present a variety of the well known algorithms for clustering text streams. This includes popular methods for topic detection and tracking in text stream. This is because the process of event detection is closely related to the clustering problem. The methods for classification of text streams are reviewed in section 3. Section 4 presents methods for evolution analysis of text stream. Social streams are discussed in section 5. The conclusions and summary are presented in section 6.

2. CLUSTERING TEXT STREAMS

The problem of clustering text streams has been widely studied in the context of numerical data [3; 6; 15]. Other popular methods which have been studied in the machine learning literature include the COBWEB and CLASSIT methods [24; 29]. The COBWEB algorithm assumes nominal attributes, whereas the CLASSIT algorithm assumes real-valued attributes. Many of these methods [6; 15] are extensions of the k-means method as extended to the stream scenario. This trend has also been applied to the case of text streams. One of the earliest methods for streaming text clustering is discussed in [63]. This technique is referred to as the *Online Spherical k-Means Algorithm (OSKM)*, which reflects the broad approach used by the methodology. This techniques divides up the incoming stream into small segments, each of which can be processed effectively in main memory. A set of k-means iterations are applied to each such data segment to cluster them. The advantage of using a segment-wise approach for clustering is that since each segment can be held in main memory, we can process each data point multiple times as long as it is held in main memory. In addition, the centroids from the previous segment are used in the next iteration for clustering purposes. A decay factor is introduced to age-out the old documents, so that the new documents are considered more important from a clustering perspective. This approach has been shown to be extremely effective in clustering massive text streams in [63].

The method in [63] is designed as a flat clustering algorithm, in which there is a single level to the clustering process. In many applications, it is useful to design hierarchical clustering algorithms in which different levels of the clustering can be explored. In the context of text data, this implies that different levels of topics and subtopics can be explored with the use of a hierarchical clustering process. A distributional modeling method for hierarchical clustering of streaming documents has been proposed in [44]. The method extends the COBWEB and CLASSIT algorithms [24; 29] to the case of text data. The work in [44] studies the different kinds of distributional assumptions of words in documents. We note that these distributional assumptions are required to adapt these algorithms to the case of text data. The approach essentially changes the distributional assumption so that the method can work effectively for text data.

A different method for clustering massive text and categorical data streams is discussed in [5]. The method discussed in [5] uses an approach which examines the relationship between outliers, emerging trends, and clusters in the underlying data. Old clusters may become inactive, and eventually get replaced by new clusters. Similarly, when newly arriving data points do not naturally fit in any particular cluster, these need to be initially classified as outliers. However, as time progresses, these new points may create a distinctive pattern of activity which can be recognized as a new cluster. The temporal locality of the data stream is manifested by these new clusters. For example, the first web page belonging to a particular category in a crawl may be recognized as an outlier, but may later form a cluster of documents of its own. On the other hand, the new outliers may not necessarily result in the formation of new clusters. Such outliers are true short-term abnormalities in the data since they do not result in the emergence of sustainable patterns. The approach discussed in [5] recognizes new clusters by first recognizing them as outliers.

This approach works with the use of a summarization methodology that is motivated by the micro-clustering approach proposed in [6]. While the concept of micro-clustering was designed for numerical data, it can also be extended to the case of text and categorical data streams. This methodology essentially creates summaries from the data points which are used to estimate the assignment of incoming data points to clusters. The concept of micro-clusters is generalized to that of *condensed droplets* in [5].

To ensure greater importance of more recent data, a time-sensitive weightage is assigned to each data point. It is assumed that each data point has a time-dependent weight defined by the function $f(t)$. The function $f(t)$ is also referred to as the *fading function*. The fading function $f(t)$ is a non-monotonic decreasing function which decays uniformly with time t. To formalize this concept, we will define the *half-life* of a point in the data stream.

Definition 2.1. *The half life t_0 of a point is defined as the time at which $f(t_0) = (1/2)f(0)$.*

Conceptually, the aim of defining a half life is to quantify the rate of decay of the importance of each data point in the stream clustering process. The *decay-rate* is defined as the inverse of the half life of the data stream. We denote the decay rate by $\lambda = 1/t_0$. We denote the weight function of each point in the data stream by $f(t) = 2^{-\lambda \cdot t}$. From the perspective of the clustering process, the weight of each data point is $f(t)$. It is easy to see that this decay function creates a half life of $1/\lambda$. It is also evident that by changing the value of λ, it is possible to change the rate at which the importance of the historical information in the data stream decays. The higher the value of λ, the lower the importance of the historical information compared to more recent data. For more stable data streams, it is desirable to pick a smaller value of λ, whereas for rapidly evolving data streams, it is desirable to pick a larger value of λ.

When a cluster is created during the streaming process by a newly arriving data point, it is allowed to remain as a trend-setting outlier for at least one half-life. During that period, if at least one more data point arrives, then the cluster becomes an active and mature cluster. On the other hand, if no new points arrive during a half-life, then the trend-setting outlier is recognized as a true anomaly in the data stream. At this point, this anomaly is removed from the list of current clusters. We refer to the process of removal as *cluster death*. Thus, a new cluster containing one data point dies when the (weighted) number of points in the cluster is 0.5. The same criterion is used to define the death of mature clusters. A necessary condition for this criterion to be met is that the inactivity period in the cluster has exceeded the half life $1/\lambda$. The greater the number of points in the cluster, the greater the level by which the inactivity period would need to exceed its half life to meet the criterion. This is a natural solution, since it is intuitively desirable to have stronger requirements (a longer inactivity period) for the death of a cluster containing a larger number of points.

Next, we describe the process of creation of a condensed-droplet from the underlying text stream. An important point to remember is that a text data set can be treated as a *sparse numeric* data set. This is because most documents contain only a small fraction of the vocabulary with non-zero frequency. For a cluster of documents $\mathcal{C}$ at time t, we denote the corresponding condensed droplet by $\mathcal{D}(t, \mathcal{C})$.

Definition 2.2. *A cluster droplet $\mathcal{D}(t, \mathcal{C})$ for a set of text data points $\mathcal{C}$ at time t is defined to as a tuple $(\overline{DF2}, \overline{DF1}, n, w(t), l)$. Each tuple component is defined as follows:*

- *The vector $\overline{DF2}$ contains $3 \cdot wb \cdot (wb-1)/2$ entries. Here wb is the number of distinct words in the cluster $\mathcal{C}$. For each pair of dimensions, we maintain a list of the pairs of word ids with non-zero counts. We also maintained the sum of the weighted counts for such word pairs.*

- *The vector $\overline{DF1}$ contains $2 \cdot wb$ entries. We maintain the identities of the words with non-zero counts. In addition, we maintain the sum of the weighted counts for each word occurring in the cluster.*

- *The entry n contains the number of data points in the cluster.*

- *The entry $w(t)$ contains the sum of the weights of the data points at time t. We note that the value $w(t)$ is a function of the time t and decays with time unless new data points are added to the droplet $\mathcal{D}(t)$.*

- *The entry l contains the time stamp of the last time that a data point was added to the cluster.*

The concept of cluster droplet has some interesting properties that will be useful during the maintenance process. These properties relate to the additivity and decay behavior of the cluster droplet.

OBSERVATION 2.1. *Consider the cluster droplets $\mathcal{D}(t,\mathcal{C}_1) = (\overline{DF2_1}, \overline{DF1_1}, n_1, w(t)_1, l_1)$ and $\mathcal{D}(t,\mathcal{C}_2) = (\overline{DF2_2}, \overline{DF1_2}, n_2, w(t)_2, l_2)$. Then the cluster droplet $\mathcal{D}(t,\mathcal{C}_1 \cup \mathcal{C}_2)$ is defined by the tuple $(\overline{DF2_1}+\overline{DF2_2}, \overline{DF1_1}+\overline{DF1_2}, n_1+n_2, w(t)_1+w(t)_2, max\{l_1,l_2\})$.*

The cluster droplet for the union of two clusters is the sum of individual entries. The only exception is the last entry which is the maxima of the two last-update times. We note that the additivity property provides considerable convenience for data stream processing since the entries can be updated efficiently using simple additive operations.
The second observation relates to the rate of decay of the condensed droplets. Since the weights of each data point decay with the passage of time, the corresponding entries also decay at the same rate. Correspondingly, we make the following observation:

OBSERVATION 2.2. *Consider the cluster droplet $\mathcal{D}(t,\mathcal{C}) = (\overline{DF2}, \overline{DF1}, n, w(t), l)$. Then the entries of the same cluster droplet $\mathcal{C}$ at a time $t' > t$ are given by $\mathcal{D}(t',\mathcal{C}) = (\overline{DF2} \cdot 2^{-\lambda \cdot (t'-t)}, \overline{DF1} \cdot 2^{-\lambda \cdot (t'-t)}, n, w(t) \cdot 2^{-\lambda \cdot (t'-t)}, l)$.*

The second observation is critical in regulating the rate at which the cluster droplets are updated during the clustering process. Since all cluster droplets decay at essentially the same rate (unless new data points are added), it follows that it is not necessary to update the decay statistics at each time stamp. Rather, each cluster droplet can be updated lazily, whenever new data points are added to it.
The overall algorithm proceeds as follows. At the beginning of algorithmic execution, we start with an empty set of clusters. As new data points arrive, unit clusters containing individual data points are created. Once a maximum number k of such clusters have been created, we can begin the process of online cluster maintenance. Thus, we initially start off with a trivial set of k clusters. These clusters are updated over time with the arrival of new data points.
When a new data point $\overline{X}$ arrives, its similarity to each cluster droplet is computed. In the case of text data sets, the cosine similarity measure [21; 47] between $\overline{DF1}$ and $\overline{X}$ is used. The similarity value $S(\overline{X}, \mathcal{C}_j)$ is computed from the incoming document $\overline{X}$ to every cluster $\mathcal{C}_j$. The cluster with the maximum value of $S(\overline{X}, \mathcal{C}_j)$ is chosen as the relevant cluster for data insertion. Let us assume that this cluster is $\mathcal{C}_{mindex}$. We use a threshold denoted by *thresh* to determine whether the incoming data point is an outlier. If the value of $S(\overline{X}, \mathcal{C}_{mindex})$ is larger than the threshold *thresh*, then the point $\overline{X}$ is assigned to the cluster $\mathcal{C}_{mindex}$. Otherwise, we check if some inactive cluster exists in the current set of cluster droplets. If no such inactive cluster exists, then the data point $\overline{X}$ is added to $\mathcal{C}_{mindex}$. On the other hand, when an inactive cluster does exist, a new cluster is created containing the solitary data point $\overline{X}$. This newly created cluster replaces the inactive cluster. We note that this new cluster is a potential true outlier or the beginning of a new trend of data points. Further understanding of this new cluster may only be obtained with the progress of the data stream.
In the event that $\overline{X}$ is inserted into the cluster $\mathcal{C}_{mindex}$, we need to perform two steps:

- We update the statistics to reflect the decay of the data points at the current moment in time. This updating is performed using the computation discussed in Observation 2.2. Thus, the relevant updates are performed in a "lazy" fashion. In other words, the statistics for a cluster do not decay, until a new point is added to it. Let the corresponding time stamp of the moment of addition be t. The last update time l is available from the cluster droplet statistics. We multiply the entries in the vectors $\overline{DC2}$, $\overline{DC1}$ and $w(t)$ by the factor $2^{-\lambda \cdot (t-l)}$ to update the corresponding statistics. We note that the lazy update mechanism results in stale decay characteristics for most of the clusters. This does not however affect the afore-discussed computation of the similarity measures.

- In the second step, we add the statistics for each newly arriving data point to the statistics for $\mathcal{C}_{mindex}$ by using the computation discussed in Observation 2.2.

In the event that the newly arriving data point does not naturally fit in any of the cluster droplets and an inactive cluster does exist, then we replace the most inactive cluster by a new cluster containing the solitary data point $\overline{X}$. In particular, the replaced cluster is the least recently updated cluster among all inactive clusters. This process is continuously performed over the life of the data stream, as new documents arrive over time. The work in [5] also presents a variety of other applications of the stream clustering technique such as evolution and correlation analysis.
A different way of utilizing the temporal evolution of text documents in the clustering process is described in [30]. Specifically, the work in [30] uses *bursty features* as markers of new topic occurrences in the data stream. This is because the semantics of an up-and-coming topic are often reflected in the frequent presence of a few distinctive words in the text stream. A specific example illustrates the bursty features in two topics corresponding to the two topics of *"US Mid-Term Elections"* and *"Newt Gingrich resigns from House"* respectively. The corresponding bursty features [30]

which occurred frequently in the newsstream during the period for these topics were as follows:

US Mid-term Elections (Nov. 3, 1998 burst):
election, voters, Gingrich, president, Newt, ...

Newt Gingrich resigns from house (Nov 6, 1998 burst):
House, Gingrich, Newt, president, Washington ...

It is evident that at a given period in time, the nature of relevant topics could lead to bursts in specific features of the data stream. Clearly, such features are extremely important from a clustering perspective. Therefore, the method discussed in [30] uses a new representation, which is referred to as the *bursty feature representation* for mining text streams. In this representation, a time-varying weight is associated with the features depending upon its burstiness. This also reflects the varying importance of the feature to the clustering process. Thus, it is important to remember that a particular document representation is dependent upon the particular instant in time at which it is constructed.

Another issue which is handled effectively in this approach is an implicit reduction in dimensionality of the underlying collection. Text is inherently a high dimensional data domain, and the pre-selection of some of the features on the basis of their burstiness can be a natural way to reduce the dimensionality of document representation. This can help in both the effectiveness and efficiency of the underlying algorithm.

The first step in the process is to identify the bursty features in the data stream. To achieve this goal, the approach uses Kleinberg's 2-state finite automaton model [31]. Once these features have been identified, the bursty features are associated with weights which depend upon their level of burstiness. Subsequently, a bursty feature representation is defined to reflect the underlying weight of the feature. Both the identification and the weight of the bursty feature are dependent upon its underlying frequency. A standard k-means approach is applied to the new representation to construct the clustering. It was shown in [30] that the approach of using burstiness improves the cluster quality. Once criticism of the work in [30] is that it is mostly focussed on the issue of improving effectiveness with the use of temporal characteristics of the data stream, and does not address the issue of efficient clustering of the underlying data stream.

In general, it is evident that feature extraction is important for all clustering algorithms. While the work in [30] focusses on using temporal characteristics of the stream for feature extraction, the work in [37] focusses on using *phrase extraction* for effective feature selection. This work is also related to the concept of topic-modeling, which will be discussed somewhat later. This is because the different topics in a collection can be related to the clusters in a collection. The work in [37] uses topic-modeling techniques for clustering. The core idea in the work of [37] is that individual words are not very effective for a clustering algorithm because they miss the context in which the word is used. For example, the word "star" may either refer to a celestial body or to an entertainer. On the other hand, when the phrase "fixed star" is used, it becomes evident that the word "star" refers to a celestial body. The phrases which are extracted from the collection are also referred to as *topic signatures*.

The use of such phrasal clarification for improving the quality of the clustering is referred to as *semantic smoothing* because it reduces the noise which is associated with semantic ambiguity. Therefore, a key part of the approach is to extract phrases from the underlying data stream. After phrase extraction, the training process determines a translation probability of the phrase to terms in the vocabulary. For example, the word "planet" may have high probability of association with the phrase "fixed star", because both refer to celestial bodies. Therefore, for a given document, a rational probability count may also be assigned to all terms. For each document, it is assumed that all terms in it are generated either by a topic-signature model, or a background collection model.

The approach in [37] works by modeling the soft probability $p(w|C_j)$ for word w and cluster C_j. The probability $p(w|C_j)$ is modeled as a linear combination of two factors; (a) A maximum likelihood model which computes the probabilities of generating specific words for each cluster (b) An indirect (translated) word-membership probability which first determines the maximum likelihood probability for each topic-signature, and then multiplying with the conditional probability of each word, given the topic-signature. We note that we can use $p(w|C_j)$ to estimate $p(d|C_j)$ by using the product of the constituent words in the document. For this purpose, we use the frequency $f(w, d)$ of word w in document d.

$$p(d|C_j) = \prod_{w \in d} p(w|C_j)^{f(w,d)} \qquad (1)$$

We note that in the static case, it is also possible to add a background model to improve the robustness of the estimation process. This is however not possible in a data stream because of the fact that the background collection model may require multiple passes to build effectively. The work in [37] maintains these probabilities in online fashion with the use of a *cluster profile*, that weights the probabilities with the use of a fading function. We note that the concept of cluster profile is analogous to the concept of condensed droplet introduced in [5]. The key algorithm (denoted by OCTS) is to maintain a dynamic set of clusters into which documents are progressively assigned with the use of similarity computations. It has been shown in [37] how the cluster profile can be used to efficiently compute $p(d|C_j)$ for each incoming document. This value is then used to determine the similarity of the documents to the different clusters. This is used to assign the documents to their closest cluster. We note that the methods in [5; 37] share a number of similarities in terms of (a) maintenance of cluster profiles, and (b) use of cluster profiles (or condensed droplets) to compute similarity and assignment of documents to most similar clusters. (c) The rules used to decide when a new singleton cluster should be created, or one of the older clusters should be replaced.

The main difference between the two algorithms is the technique which is used to compute cluster similarity. The OCTS algorithm uses the probabilistic computation $p(d|C_j)$ to compute cluster similarity, which takes the phrasal information into account during the computation process. One observation about OCTS is that it may allow for very similar clusters to co-exist in the current set. This reduces the space available for distinct cluster profiles. A second algorithm called OCTSM is also proposed in [37] that allows for merging of very similar clusters. Before each assignment, it checks whether pairs of similar clusters can be merged on the basis of similarity. If this is the case, then we allow the merging of the similar clusters and their corresponding cluster profiles. Detailed experimental results on the different clustering algorithms and their effectiveness are presented in [37].

Another method [49] uses a combination of a spectral partitioning and probabilistic modeling method for novelty detection and topic tracking. This approach uses a HITS-like spectral technique within a probabilistic framework. The probabilistic part is an unsupervised boosting method that is closer to semi-parametric maximum likelihood methods.

A closely related area to clustering is that of topic modeling, which is a problem closely related to that of clustering. In the problem of topic modeling, we perform a *soft* clustering of the data in which each document has a membership probability to one of a universe of topics rather than a deterministic membership. A variety of mixture modeling techniques can be used to determine the topics from the underlying data. Recently, the method has also been extended to the *dynamic* case which is helpful for topic modeling of text streams [13]. A closely related topic is that of topic *detection* and *tracking*, which is discussed below.

Recently, a variety of methods for maintaining topic models in a streaming scenario have been proposed in [58]. The work evaluates a number of different methods for adapting topic models to the streaming scenario. These include methods such as Gibbs sampling and variational inference. In addition, a method is also proposed that is based on text classification. The latter has the advantage of requiring only a single matrix multiplication, and is therefore much more efficient. A method called *SparseLDA* is proposed that is an effective method for evaluating Gibbs sampling distributions. The results in [58] show that this method is 20 times faster than traditional LDA.

In some applications, it is desirable to have clusters of approximately balanced size, in which a particular cluster is not significantly larger than the others. A competitive online learning for determining such balanced clusters have been proposed in [11]. The essential approach in [11] is to design a model in which it is harder for new data points to join larger clusters. This is achieved by penalizing the imbalance into the objective function criterion. It was shown in [11] that such an approach can determine well balanced clusters.

2.1 Topic Detection and Tracking in Text Streams

A closely related problem to clustering is that of *topic detection and tracking* [7; 14; 28; 55; 56; 60]. In this problem, we create unsupervised clusters from a text stream, and then determine the sets of clusters which match real events. These real events may correspond to documents which are identified by a human. Since the problem of online topic detection is closely related to that of clustering, we will discuss this problem as a subsection of our broader discussion on clustering, though not all methods for topic detection use clustering techniques. In this subsection, we will discuss all the methods for topic detection, whether they use clustering or not.

The earliest work on topic detection and tracking was performed in [7; 56]. The work arose out of a DARPA initiative [64] which formally defined this problem and proposed the initial set of algorithms for the task. An interesting technique in [56] designs methods which can be used for both retrospective and online topic tracking and detection. In retrospective event detection, we create groups from a corpus of documents (or stories), and each group corresponds to an event. The online version is applicable to the case of data streams, and in this case, we process documents sequentially to determine whether an incoming document corresponds to a new event. An online clustering algorithm can also be used to track the different events in the data in the form of clusters. For each incoming document $\overline{X}$ we compute its similarity to the last m documents $\overline{Y_{t-1}} \dots \overline{Y_{t-m}}$. The score for the incoming document $\overline{X}$ is computed as follows:

$$score(\overline{X}) = \max_{i \in \{t-m \dots t-1\}}(1 - sim(\overline{X}, \overline{Y_i})) \qquad (2)$$

A document is considered novel, when its score is above a pre-defined threshold. In addition, a decay-weighted version is designed in which the weight of documents depends upon its recency. The idea here is that a document is considered to be a new event when the last occurrence of a similar document did not occur recently "enough". In this case, the corresponding score is designed as follows:

$$score(\overline{X}) = \max_{i \in \{t-m \dots t-1\}}(1 - \frac{(m+i-t+1)}{m} \cdot sim(\overline{X}, \overline{Y_i})) \qquad (3)$$

We note that each $\overline{Y_i}$ need not necessarily represent a single document, but may also represent a cluster or a larger grouping. We also note that the method for detecting a new event can be combined with a cluster tracking task. This is because the determination of a new event is indicative of a new event (or singleton cluster) in the data.

The work in [7] addressed the problem of new event detection by examining the relationship of a current document to the previous documents in the data. The key idea is to use feature extraction and selection techniques to design a query representation of the document content. We determine the query's initial threshold by comparing the document content with the query, and set it as the triggering threshold. Then, we determine if the document triggers any queries from the previous documents in the collection. If no queries are triggered, then the document is deemed to be a new event. Otherwise, this document is not a new event.

One general observation about the online topic detection and tracking problem [8; 57] is that this problem is quite hard in general, and the performance of first event detection can degrade under a variety of scenarios. To improve the effectiveness of first event detection, the work in [57] proposes to use the training data of old events to learn useful statistics for the prediction of new events. The broad approach in [57] contains the following components:

- The documents are classified into broad topics, each of which consists of multiple events.
- Named entities are identified, optimizing their weight relative to normal words for each topic, and computing a stopword list for each topic.
- Measuring the novelty of a new document conditioned on the system-predicted topic for that document.

Clearly such an approach has the tradeoff that it requires prior knowledge about the collection in the form of training data, but provides better accuracy. More details on the approach may be found in [57].

A method proposed in [14] is quite similar to that proposed in [56], except that it proposes a number of improvements in how the tf–idf model is incrementally maintained for computation of similarity. For example, a *source-specific* tf–idf model is maintained in which the statistics are specific to each news source. Similarly, the approach normalizes for the fact that documents which come from the same source tend to have a higher similarity because of the vocabulary conventions which are often used in the similarity computation. To achieve this goal, the approach computes the average similarity between the documents from a particular pair of sources and subtracts this average value while computing the similarity between a pair of documents for the purpose of new event detection. A number of other techniques for improving the quality of event detection (such as using inverse event frequencies) are discussed in [14]. A method for online topic detection and tracking is presented in [60] as an application of stream clustering. This work uses a probabilistic LDA model to create an online model for estimating the growing number of clusters. The general approach in this work is similar to the concepts already proposed in [56]. The main novelty of the work is the design of an online approach for probabilistic clustering.

Another method for fast and parameter-free bursty event detection is proposed in [28]. This approach focusses on finding bursty features which characterize the presence of an event. In order to achieve this goal, the technique in [28] proposes a feature-pivot clustering that groups features on the basis of bursty co-occurrence. The approach is designed to be parameter-free, which gives it an advantage in a number of scenarios.

The problem of event detection has also been studied with the use of *keyword graphs* in [46]. The work in [46] builds a keywords graph from a text stream in which a node corresponds to a keyword, and an edge is added to the keyword graphs when a pair of words occur together in the document. Events are characterized as communities of keywords in this network. This broad approach is used in the context of a *window-based technique* for the case of social streams.

In the context of social network streams, a natural question arises, as to whether one can use any of the *social dimensions* of the underlying stream to improve the underlying event detection process. Such an approach has been proposed in [62], in which events are determined by combining text clustering, social network structure clustering, and temporal segmentation. In addition, a multi-dimensional visualization tool is provided, which discusses ways of visualizing the relationships between the events along the three dimensions. In this case, an event is defined as a set of text documents which are semantically coherent, and are structurally well connected both in terms of social network structure and time. These three different characteristics are used in the following ways:

- First, the content is used to create a hierarchy of topics from the social text stream.
- While the topical patterns are useful for distinguishing content, the temporal segmentation is used to distinguish different events. The assumption is that the communication between different parties happen during a short contiguous time period.
- Since it is assumed that the events occur between connected entities, we use the connectivity between the different events to further segment the events. An edge between a pair of nodes corresponds to a communication between the social entities. Multiple edges are allowed between pairs of nodes. This structure is used to determine the event-dependent communities.

Another method [43] has been proposed in the context of the *Twitter* social network data set. In this paper, a locality sensitive hashing method (LSH) is used to keep track of the different documents. The idea in LSH [17] is to use a hashing scheme in which the probability of hashing two documents into the same bucket is proportional to the cosine of the angle between the two documents. For a given query document, we check all the documents in the same bucket, and then perform the similarity calculation explicitly with all documents in the same bucket. The closest document is returned as the nearest neighbor. We note that the problem of first-story detection can be considered an application of repeated nearest neighbor querying in which an incoming document is compared to the previously seen documents from the data stream. While LSH can be used directly in conjunction with a nearest neighbor search for first story detection, such an approach typically leads to poor results. This is because LSH works effectively only if the true nearest neighbor is close to the query point. Otherwise, such an approach is unable to find the true nearest neighbor. Therefore, the approach in [43] uses the strategy of using LSH only for declaring a document to be sufficiently new on an aggregate basis. For such cases, the document is compared against a fixed number of the most recent documents. In the event that the corresponding distance is above a given threshold, we can declare the underlying story as novel. The main advantage of this technique over many of the aforementioned techniques is that of *speed*, which is especially important, when dealing with social streams of very high volume. This speed is achieved because of the use of the LSH technique, though there is some loss in accuracy because of the approximation process. This technique was compared [43] against the *UMass system* [9], and it was found that more than an order of magnitude improvement in speed was obtained with only a minor loss in accuracy.

Most of the work in text stream mining and topic detection is performed in the context of a *single news stream*. In many cases, we have multiple text streams [53], which are indexed by the same set of time points (called coordinated text streams). For example, when a major event happens, all the news articles published by different agencies in different languages cover the same event in that period. This is referred to as a *correlated bursty topic pattern* in the different news article streams. In some cases, when the correlated streams are multi-lingual, they may even have completely different vocabulary. The discovery of bursty topic patterns can determine the interesting events and associations between different streams. Such an approach can also determine interesting local and stream-specific patterns by factoring out the global noise which is common to the different streams.

To achieve this goal, the technique in [53] aligns the text samples from different streams on the basis of the shared time-stamps. This is used to discover topics from multiple streams simultaneously with a single probabilistic mixture model. The approach of constructing independent topic models from different streams is that the topic models from the different streams may not match each other very well, or at least, create a challenge in matching the different topic models, if it is done at the end of the process of model construction. Therefore, it is important to make the mixture models for the different streams communicate with one another during the modeling process, so that a single mixture model is constructed across the different streams. To achieve this goal, the stream samples at a given point are merged together, while keeping the stream identities. To achieve this, we align the topic models from different streams while keeping their identities. While the topic models from different streams are separate, the global mixture model is designed for the overall text stream sample. Such a mixture model is coordinated, because it would emphasize topics which tend to occur across multiple streams. Once the coordinated mixture model is obtained, the topical models for the different streams can be extracted by fitting the mixture model to the different streams. As the topic models for the different streams ate aligned with one another, we can obtained a correlated bursty topic pattern, when the corresponding topic is bursty during the same period. A key aspect of this approach is that it does not require the different streams to share vocabulary. Rather it is assumed the topics involved in a correlated bursty topic pattern tend to have the same distribution across streams, and this can be used to match topics from different streams. More details on the approach may be found in [53].

2.2 Application to Novelty Detection

Methods for clustering text streams can be used for novelty detection, as illustrated in [5; 60]. In all these methods, the formation of a new cluster corresponds to a new pattern in the underlying data. This represents a novelty with respect to the previous history of the stream. Eventually, this novelty may become a normal pattern, as more data points are added to this cluster. This cluster therefore represents a sudden change in the nature of the data stream, such as a newsworthy event or other cluster. In other cases, the novelty may be isolated, and may not turn into a new pattern. This issue has been discussed in detail in [5]. Such an approach for detecting novelties is not specific to the text domain, but can also be used for other data domains.

3. CLASSIFICATION OF TEXT STREAMS

The problem of classification of data streams has been widely studied by the data mining community in the context of different kinds of data [3; 4; 52; 59]. Many of the methods for classifying numerical data can be easily extended to the case of text data, just as the numerical clustering method in [6] has been extended to a text clustering method in [5]. In particular, the classification method of [4] can be extended to the text domain, by adapting the concept of numerical micro-clusters to condensed droplets as discussed in [5]. With the use of the condensed droplet data structure, it is possible to extend all the methods discussed in [4] to the text stream scenario. Similarly, the core idea in [52] uses an ensemble based approach on chunks of the data stream. This broad approach is essentially a *meta-algorithm* which is not too dependent upon the specifics of the underlying data format. Therefore, the broad method can also be easily extended to the case of text streams.

The problem of text stream classification arises in the context of a number of different IR tasks. The most important of these is *news filtering* [34], in which it is desirable to automatically assign incoming documents to pre-defined categories. A second application is that of email spam filtering [10], in which it is desirable to determine whether incoming email messages are spam or not. We note that the problem of text stream classification can arise in two different contexts, depending upon whether the training or the test data arrives in the form of a stream:

- In the first case, the training data may be available for batch learning, but the test data may arrive in the form of a stream.
- In the second case, both the training and the test data may arrive in the form of a stream. The patterns in the training data may continuously change over time, as a result of which the models need to be updated dynamically.

The first scenario is usually easy to handle, because most classifier models are compact and classify individual test instances efficiently. On the other hand, in the second scenario, the training model needs to be constantly updated to account for changes in the patterns of the underlying training data. The easiest approach to such a problem is to incorporate temporal decay factors into model construction algorithms, so as to age out the old data. This ensures that the new (and more timely data) is weighted more significantly in the classification model. An interesting technique along this direction has been proposed in [45], in which a temporal weighting factor is introduced to modify the classification algorithms. Specifically, the approach has been applied to the Naive Bayes, Rocchio, and k-nearest neighbor classification algorithms. It has been shown that the incorporation of temporal weighting factors is useful in improving the classification accuracy, when the underlying data is evolving over time.

A number of methods have also been designed specifically for the case of text streams. In particular, the method discussed in [27] studies methods for classifying text streams in which the classification model may evolve over time. This problem has been studied extensively in the literature in the context of multi-dimensional data streams [4; 52]. For example, in a spam filtering application, a user may generally delete the spam emails for a particular topic, such as those corresponding to political advertisements. However, in a particular period such as the presidential elections, the user may be interested in the emails for that topic, it may not be appropriate to continue to classify that email as spam.

The work in [27] looks at the particular problem of classification in the context of user-interests. In this problem, the label of a document is considered either *interesting* or *non-interesting.* To achieve this goal, the work in [27] maintains the interesting and non-interesting topics in a text stream together with the evolution of the theme of the interesting topics. A document collection is classified into multiple topics, each of which is labeled either interesting or non-interesting at a given point. A concept refers to the main theme of interesting topics. A concept drift refers to the fact that the main theme of the interesting topic has changed.

The main goals of the work are to maximize the accuracy of classification and minimize the cost of re-classification. To achieve this goal, the method in [27] designs methods for detecting both *concept drift* as well as *model adaptation.* The former refers to the change in the theme of user-interests, whereas the latter refers to the detection of brand new concepts. To detect concept drifts, the method in [27] measures the classification error-rates in the data stream in terms of true and false positives. When the stream evolves, these error rates will increase, if the change in the concepts are not detection. To determine the change in concepts, techniques from statistical quality control are used, in which we determine the mean μ and standard deviation σ of the error rates, and determine whether this error rate remains within a particular tolerance which is $[\mu - k \cdot \sigma, \mu + k \cdot \sigma]$. Here the tolerance is regulated by the parameter k. When the error rate changes, we determine when the concepts should be dropped or included. In addition, the drift rate is measured to determine the rate at which the concepts should be changed for classification purposes. In addition, methods for dynamic construction and removal of sketches are discussed in [27].

Another related work is that of one-class classification of text streams [61], in which only training data for the positive class is available, but there is no training data available for the negative class. This is quite common in many real applications in which it easy to find representative documents for a particular topic, but it is hard to find the representative documents to model the background collection. The method works by designing an ensemble of classifiers in which some of the classifiers corresponds to a recent model, whereas others correspond to a long-term model. This is done to incorporate the fact that the classification should be performed with a combination of short-term and long-term trends.

A rule-based technique that can learn classifiers incrementally from data streams is the *sleeping-experts systems* [19; 25]. One characteristic of this rule-based system is that it uses the position of the words in generating the classification rules. Typically, the rules correspond to sets of words which are placed close together in a given text document. These sets of words are related to a class label. For a given test document, it is determined whether these sets of words occur in the document, and are used for classification. This system essentially learns a set of rules (or sleeping experts) that can be updated incrementally by the system. While the technique was proposed prior to the advent of data stream technology, its online nature ensures that it can be effectively used for the stream scenario.

One of the classes of methods which can be easily adapted to stream classification is the broad category of *neural networks* [48; 54]. This is because neural networks are essentially designed as a classification model with a network of perceptrons and corresponding weights associated with the term-class pairs. Such an incremental update process can be naturally adapted to the streaming context. These weights are incrementally learned as new examples arrive. The first neural network methods for online learning were proposed in [48; 54]. In these methods, the classifier starts off by setting all the weights in the neural network to the same value. The incoming training example is classified with the neural network. In the event that the result of the classification process is correct, then the weights are not modified. On the other hand, if the classification is incorrect, then the weights for the terms are either increased or decreased depending upon which class the training example belongs to. Specifically, if class to which the training example belongs is a positive instance, the weights of the corresponding terms (in the training document) are increased by α. Otherwise, the weights of these terms are reduced by α. The value of α is also known as the *learning rate*. Many other variations are possible in terms of how the weights may be modified. For example, the method in [22] uses a multiplicative update rule, in which two multiplicative constants $\alpha_1 > 1$ and $\alpha_2 < 1$ are used for the classification process. The weights are multiplied by α_1, when the example belongs to the positive class, and is multiplied by α_2 otherwise. Another variation [35] also allows the modification of weights, when the classification process is correct. A number of other online neural network methods for text classification (together with background on the topic) may be found in [20; 26; 40; 42]. A Bayesian method for classification of text streams is discussed in [16]. The method in [16] constructs a Bayesian model of the text which can be used for online classification. The key components of this approach are the design of a Bayesian online perceptron and a Bayesian online Gaussian process that can be used effectively for online learning.

4. EVOLUTION ANALYSIS IN TEXT STREAMS

A key problem in the case of text is to determine evolutionary patterns in temporal text streams. An early survey on the topic of evolution analysis in text streams may be found in [32]. Such evolutionary patterns can be very useful in a wide variety of applications, such as summarizing events in news articles and revealing research trends in the scientific literature. For example, an event may have a life cycle in the underlying theme patterns such as the beginning, duration, and end of a particular event. Similarly, the evolution of a particular topic in the research literature may have a life-cycle in terms of how the different topics affect one another. This problem was defined in [38], and contains three main parts: (a) Discovering the themes in text; (b) creating an evolution graph of themes; and (c) studying the life cycle of themes.

A *theme* is defined as a semantically related set of words, with a corresponding probability distribution that coherently represents a particular topic or sub-topic. This corresponds to a model that is represented by θ. The *span* of such a theme represents the starting and end point of the corresponding theme in terms of the time-interval (s, t). Thus, the theme span is denoted by the triple $\gamma = (\theta, s, t)$. As time passes, a particular theme γ_1 may perform a transition into another theme γ_2. A theme γ_1 is said to have evolved into another theme γ_2, if there is sufficient similarity between their corresponding spans. It is possible to create a *theme evolution graph* $G = (N, A)$, in which each node in N corresponds to a theme, and each edge in A corresponds to a transition between two themes. The overall approach requires three steps:

- In the first step, we segment the text stream into a number of possibly overlapping sub-collections with fixed or variable time spans. This is done in an application-specific way.

- The salient themes are determined from each collection with the use of a probabilistic mixture model. A standard mixture model technique [23] was used for this purpose.

- Finally, all the evolution transitions are determined from these theme patterns. This is done with the use of the KL-divergence measure to compute the evolution distance between two themes. In the event that the similarity is above a given threshold, the transition is considered valid.

In addition, a method is proposed in [38] for analyzing the entire theme life cycle by measuring the strength of the theme over different periods. A method based on HMM is proposed to measure the theme-shifts over the entire period as well.

The problem of tracking new topics, ideas, and memes across the Web has been studied in [33]. This problem is related to that of the topic detection and tracking discussed earlier. However, the rate of evolution in the web and blog scenario is significantly greater than the models which have been discussed in earlier work. In the context of the web and social networks, the content spreads widely and then fades over time scales on the order of days. The work is [33] develops a framework for tracking short, distinctive phrases that persistently appear in on-line text over time. In addition, scalable algorithms were proposed for clustering textual variants of such phrases. In addition, the approach is able to perform local analysis for specific threads. This includes the determination of peak intensity and the rise and decay in the intensity of specific threads. The relationship between the news cycle and blogs is examined in terms of how events propagate from one to the other, and the corresponding time-lag for the propagation process. In the next section, the notion of social streams is discussed, which are often used in the context of online analysis.

5. SOCIAL STREAMS

In recent years, large networks such as *Facebook* and *Twitter* have produced massive streams that can be analyzed for a wide variety of insights [1; 39; 46]. Such streams are referred to as *social streams*. Social streams are particularly rich in terms of the amount of content that they can produce over time. Typically, the kind of data that can be tracked in such streams is massive, and can evolve over time. According to the official Twitter blog, the average number of tweets sent per day was 140 million in 2011, which was around 1620 tweets per second. Analogously, there are about 2 million new friend requests and 2 million messages sent every 20 minutes in *Facebook*. Clearly, the analysis of data on such a scale poses numerous challenges that are unique in streaming text analysis.

1. The data often contains structural information, such as linkage and friendship analysis. Structural data usually contains a much stronger signal than text data. The optimal analysis can often be performed only by using a combination of both structure and content [1].

2. Text stream is usually quite noisy, and the documents are *short*. Typically, a tweet in *Twitter* is space-limited, and may contain no more than a short snippet of text. These challenges are significant, because the noise in the data may be a significant impediment in the analysis. This further emphasizes the importance of using the structural information in the analysis.

3. The patterns in the data may evolve significantly over time. As a result, it is important for the analysis to be done *dynamically* in *real time*.

The analysis of social streams are important applications to key problems such as event detection [1; 46]. Many important news stories are often presented in social media, long before they appear in traditional news media. Unfortunately, the information in social media is untrustworthy, and noisy and methods are required to analyze the trustworthiness of the data in *streaming fashion* [51].

Recently, a significant amount of work has been performed in the social streaming scenario. These pieces of work are as follows.

1. The problem of event detection has been studied in [1; 41; 46]. Event detection is one of the most important problems is social stream analysis, because the nature of the problem is focussed on real-time analysis. The problem of streaming event detection in *Twitter* data has been studied in [43], in the context of first story detection.

2. The work in [36] shows how online topic analysis may be performed from tweet streams. The work explores the problem of smoothing for language models.

3. The problem of performing recommendations in social streams (eg. recommending messages) has been studied in [18]. Because of the sheer volume of such streams, it is crucial to be able to present a small number of target recommenders to the user.

4. The problem of topic-based browsing of social text streams has been studied in [12]. The browsing of large amounts of streams is crucial to enable the understanding of the key topics in the text stream.

5. The problem of influence analysis in social streams has been studied in [50]. In many social media applications, the influential entities may change over time, together with the patterns in the text stream. Therefore, a streaming approach can adjust for such changes.

A survey from the perspective of network data may be found in [2], though the notion of content-centered analysis is somewhat different from network-centered analysis. Nevertheless, a section of this survey discusses how evolving content streams arise in the context of dynamic and evolving networks. The area of social streams is still in its infancy and a significant scope exists in designing methods for different kinds of social streaming applications. This will be discussed in the next section.

6. CONCLUSIONS

This paper studies the problem of mining text streams. The challenge in the case of text stream arises because of its temporal and dynamic nature in which the patterns and trends of the stream may vary rapidly over time. The determination of the changes in the underlying patterns and trends is very useful in the context of a wide variety of applications. A variety of problems in text stream mining are examined such as clustering, classification, evolution analysis, and event detection. In addition, we studied the applications of some of these techniques in the context of new applications such as social networks. A lot of interesting avenues for research remain in the context of social media analytics, and the use of social dimensions to enhance text stream mining. In particular, the incorporation of network structure into the mining of social streams such as *Twitter* remains a relatively unexplored area, which can be a fruitful avenue for future research.

7. REFERENCES

[1] C. C. Aggarwal, and K Subbian. Event Detection in Social Streams, *SDM Conference*, 2012.

[2] C. C. Aggarwal, and K. Subbian. Evolutionary Network Analysis: A Survey, *ACM Computing Surveys*, accepted to appear, 2014.

[3] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[4] C. C. Aggarwal, J. Han, J. Wang, P. Yu. On Demand Classification of Data Streams, *KDD Conference*, 2004.

[5] C. C. Aggarwal, P. S. Yu. On Clustering Massive Text and Categorical Data Streams, *KAIS Journal*, 24(2), pp. 171–196, 2010.

[6] C. C. Aggarwal, J. Han. J. Wang, P. Yu. A Framework for Clustering Evolving Data Streams, *VLDB Conference*, 2003.

[7] J. Allan, R. Papka, V. Lavrenko. On-line new event detection and tracking. *ACM SIGIR Conference*, 1998.

[8] J. Allan, V. Lavrenko, H. Jin. First story detection in tdt is hard. *ACM CIKM Conference*, 2000.

[9] J. Allan, V. Lavrenko, D. Malin, R. Swan. Detections, bounds and timelines: Umass and tdt3, *Proceedings of the Topic Detection and Tracking Workshop*, 2000.

[10] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, C. D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. *Proceedings of the ACM SIGIR Conference*, 2000.

[11] A. Banerjee, J. Ghosh. Competitive learning mechanisms for scalable, balanced and incremental clustering of streaming texts, *NIPS Conference*, 2003.

[12] M. S. Bernstein, B. Suh, L. Hong, J. Chen, S. Kairam, and E. H. Chi. Eddi: interactive topic-based browsing of social status streams. *ACM symposium on User interface software and technology*, 2010.

[13] D. Blei, J. Lafferty. Dynamic topic models. *ICML Conference*, 2006.

[14] T. Brants, F. Chen, A. Farahat. A system for new event detection. *ACM SIGIR Conference*, 2003.

[15] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra, S. Guha. Streaming-Data Algorithms for High-Quality Clustering. *ICDE Conference*, 2002.

[16] K. Chai, H. Ng, H. Chiu. Bayesian Online Classifiers for Text Classification and Filtering, *ACM SIGIR Conference*, 2002.

[17] M. Charikar. Similarity Estimation Techniques from Rounding Algorithms, *STOC Conference*, 2002.

[18] J. Chen, R. Nairn, and E. Chi. Speak little and well: recommending conversations in online social streams. *SIGCHI Conference*, 2011.

[19] W. Cohen, Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2), pp. 141–173, 1999.

[20] K. Crammer, Y. Singer. A New Family of Online Algorithms for category ranking, *ACM SIGIR Conference*, 2002.

[21] D. Cutting, D. Karger, J. Pedersen, J. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. *Proceedings of the SIGIR*, 1992.

[22] I. Dagan, Y. Karov, D. Roth. Mistake-driven learning in text categorization. *Conference Empirical Methods in Natural Language Processing*, 1997.

[23] A. P. Dempster, N. M. Laird, D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society* 39: pp. 1–38, 1977.

[24] D. Fisher. Knowledge Acquisition via incremental conceptual clustering. *Machine Learning*, 2: pp. 139–172, 1987.

[25] Y. Freund, R. Schapire, Y. Singer, M. Warmuth. Using and combining predictors that specialize. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pp. 334–343, 1997.

[26] Y. Freund, R. Schapire. Large Margin Classification using the perceptron Algorithm, *COLT*, 1998.

[27] G. P. C. Fung, J. X. Yu, H. Lu. Classifying text streams in the presence of concept drifts. *PAKDD Conference*, 2004.

[28] G. P. C. Fung, J. X. Yu, P. Yu, H. Lu. Parameter Free Bursty Events Detection in Text Streams, *VLDB Conference*, 2005.

[29] J. H. Gennari, P. Langley, D. Fisher. Models of incremental concept formation. *Journal of Artificial Intelligence*, 40: pp. 11–61, 1989.

[30] Q. He, K. Chang, E.-P. Lim, J. Zhang. Bursty feature representation for clustering text streams. *SDM Conference*, 2007.

[31] J. Kleinberg, Bursty and hierarchical structure in streams, *ACM KDD Conference*, pp. 91–101, 2002.

[32] A. Kontostathis, L. Galitsky, W. M. Pottenger, S. Roy, D. J. Phelps. A survey of emerging trend detection in textual data mining. *Survey of Text Mining*, pp. 185–224, 2003.

[33] J. Leskovec, L. Backstrom, J. Kleinberg. Meme Tracking and the Dynamics of the News Cycle, *KDD Conference*, 2009.

[34] D. Lewis. The TREC-4 filtering track: description and analysis. *Proceedings of TREC-4, 4th Text Retrieval Conference*, pp. 165–180, 1995.

[35] D. Lewis, R. E. Schapire, J. P. Callan, R. Papka. Training algorithms for linear text classifiers. *ACM SIGIR Conference*, 1996.

[36] J. Lin, R. Snow, and W. Morgan. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. *ACM KDD Conference*, 2011.

[37] Y.-B. Liu, J.-R. Cai, J. Yin, A. W.-C. Fu. Clustering Text Data Streams, *Journal of Computer Science and Technology*, Vol. 23(1), pp. 112–128, 2008.

[38] Q. Mei, C.-X. Zhai. Discovering Evolutionary Theme Patterns from Text- An Exploration of Temporal Text Mining, *ACM KDD Conference*, 2005.

[39] M. Naaman, J. Boase, and C. H. Lai. Is it really about me?: Message content in social awareness streams. *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pp. 189–192, 2010.

[40] H. T. Ng, W. B. Goh, K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. *SIGIR Conference*, 1997.

[41] A. Ritter, O. Etzioni, and S. Clark. Open domain event extraction from twitter. *ACM KDD Conference*, 2012.

[42] F. Rosenblatt. The perceptron: A probabilistic model for information and storage organization in the brain, *Psychological Review*, 65: pp. 386–407, 1958.

[43] S. Petrovic, M. Osborne, V. Lavrenko. Streaming First Story Detection with Application to Twitter. *Proceedings of the ACL Conference*, pp. 181–189, 2010.

[44] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, R. Padman. Incremental Hierarchical Clustering of Text Documents, *ACM CIKM Conference*, 2006.

[45] T. Salles, L. Rocha, G. Pappa, G. Mourao, W. Meira Jr., M. Goncalves. Temporally-aware algorithms for document classification. *ACM SIGIR Conference*, 2010.

[46] H. Sayyadi, M. Hurst, A. Maykov. Event Detection in Social Streams, *AAAI*, 2009.

[47] H. Schutze, C. Silverstein. Projections for Efficient Document Clustering, *ACM SIGIR Conference*, 1997.

[48] H. Schutze, D. Hull, J. Pedersen. A comparison of classifiers and document representations for the routing problem. *ACM SIGIR Conference*, 1995.

[49] A. Surendran, S. Sra. Incremental Aspect Models for Mining Document Streams. *PKDD Conference*, 2006.

[50] K. Subbian, C. Aggarwal and J. Srivastava. Content-centric Flow Mining for Influence Analysis in Social Streams, *CIKM Conference*, 2013.

[51] D. Wang, T. Abdelzaher, L. Kaplan, and C. Aggarwal. Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. *ICDCS Conference*, 2013.

[52] H. Wang, W. Fan, P. Yu, J. Han, Mining Concept-Drifting Data Streams with Ensemble Classifiers, *KDD Conference*, 2003.

[53] X. Wang, C.-X. Zhai, X. Hu, R. Sproat. Mining Correlated Bursty Topic Patterns from Correlated Text Streams, *ACM KDD Conference*, 2007.

[54] E. Wiener, J. O. Pedersen, A. S. Weigend. A Neural Network Approach to Topic Spotting. *SDAIR*, pp. 317–332, 1995.

[55] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald, X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[56] Y. Yang, T. Pierce, J. Carbonell. A study on retrospective and on-line event detection. *ACM SIGIR Conference*, 1998.

[57] Y.Yang, J. Carbonell, C. Jin. Topic-conditioned Novelty Detection. *ACM KDD Conference*, 2002.

[58] L. Yao, D. Mimno, A. McCallum. Efficient methods for topic model inference on streaming document collections, *ACM KDD Conference*, 2009.

[59] K. L. Yu, W. Lam. A new on-line learning algorithm for adaptive text filtering. *ACM CIKM Conference*, 1998.

[60] J. Zhang, Z. Ghahramani, Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In *Saul L., Weiss Y., Bottou L. (eds) Advances in Neural Information Processing Letters*, 17, 2005.

[61] Y. Zhang, X. Li, M. Orlowska. One Class Classification of Text Streams with Concept Drift, *ICDMW Workshop*, 2008.

[62] Q. Zhao, P. Mitra. Event Detection and Visualization for Social Text Streams, *ICWSM*, 2007.

[63] S. Zhong. Efficient Streaming Text Clustering. *Neural Networks*, Volume 18, Issue 5-6, 2005.

[64] http://projects.ldc.upenn.edu/TDT/

Mining Social Media with Social Theories: A Survey

Jiliang Tang
Computer Science & Eng
Arizona State University
Tempe, AZ, USA
Jiliang.Tang@asu.edu

Yi Chang
Yahoo!Labs
Yahoo!Inc
Sunnyvale,CA, USA
yichang@yahoo-inc.com

Huan Liu
Computer Science & Eng
Arizona State University
Tempe, AZ, USA
Huan.Liu@asu.edu

ABSTRACT

The increasing popularity of social media encourages more and more users to participate in various online activities and produces data in an unprecedented rate. Social media data is big, linked, noisy, highly unstructured and incomplete, and differs from data in traditional data mining, which cultivates a new research field - social media mining. Social theories from social sciences are helpful to explain social phenomena. The scale and properties of social media data are very different from these of data social sciences use to develop social theories. As a new type of social data, social media data has a fundamental question - *can we apply social theories to social media data?* Recent advances in computer science provide necessary computational tools and techniques for us to verify social theories on large-scale social media data. Social theories have been applied to mining social media. In this article, we review some key social theories in mining social media, their verification approaches, interesting findings, and state-of-the-art algorithms. We also discuss some future directions in this active area of mining social media with social theories.

1. INTRODUCTION

Social media greatly enables people to participate in online activities and shatters the barrier for online users to share and consume information in any place at any time. Social media users can be both passive content consumers and active content producers, and generate data at an unprecedented rate. The nature of social media determines that its data significantly differs from the data in traditional data mining. Social relations are pervasively available in social media data, and play important roles in social media such as mitigating information overload problem [38; 51] and promoting the information propagation process [4; 67].

Social media data is big, noisy, incomplete, highly unstructured and linked with social relations. These unique properties of social media data suggest that naively applying existing techniques may fail or lead to inappropriate understandings about the data. For example, social media data is linked via social relations and contradicts with the underlying independent and identically distributed (IID) assumption of the vast majority of existing techniques [23; 57]. This new type of data calls for novel data mining techniques for a better understanding from the computational perspective. The study and development of these new techniques are under the purview of social media mining, which is the process of representing, analyzing, and extracting actionable patterns from social media data [70].

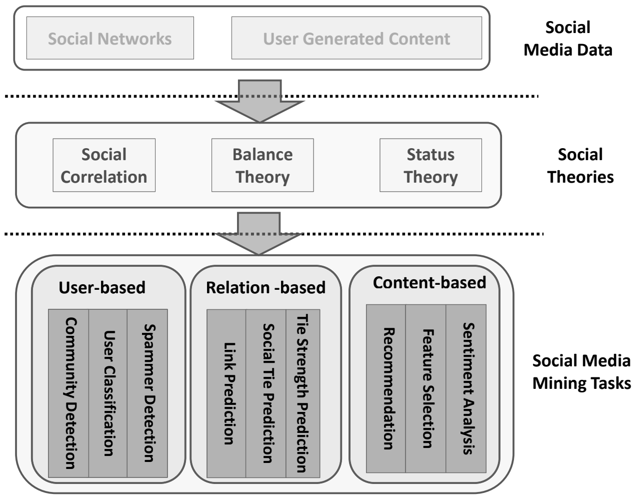


Figure 1: Social Theories in Social Media Mining.

There are many social theories developed from social sciences to explain various types of social phenomena. For example, the homophily theory [40] suggests how individuals connect to each other, while balance theory suggests that users in a social network tend to form into a balanced network structure [17]. The scale of the data social scientists employ to develop these social theories is very different from that of social media data. It is easy for social media data to include the actions and interactions of hundreds of millions of individuals in real time as well as over time. Therefore there is a fundamental question for this new type of social data - *can we apply some social theories to social media data?*. If we can apply social theories to social media data, social theories can help us understand social media data from a social perspective, and combining social theories with computational methods manifests a novel and effective perspective to mine social media data as shown in Figure 1. Social theories help bridge the gap from what we have (social media data) to what we want to understand social media data (social media mining).

Integrating social theories with computational models becomes an interesting direction in mining social media data and encourages a large body of literature in this line. The goal of this article is to provide a review of some key social theories in mining social media data. The contributions and organization of this article are summarized as below:

- The social property of social media data determines that it differs from data in traditional data mining and social sciences. In Section 2, we provide an overview of the unique properties of social media data;

- An increasing number of social theories is verified in social media data. In Section 3, we focus on three key and widely used social theories with basic concepts, verification approaches and key findings;

- The fast growing interests and intensifying need to harness social media data make social media mining grow rapidly. Integrating social theories with computational methods becomes a principled way to mine social media data. In Section 4, we review the state-of-the-art algorithms that exploit social theories in mining social media, and summary feature engineering, constraint generating and objective defining as three ways to explain social theories for computational models.

- Social theories in mining social media data is still an active area of exploration and there could be more existing social theories to be employed or new social theories to be discovered from this new type of social media data. In Section 5, we discuss some open issues and possible research directions.

2. SOCIAL MEDIA DATA IS SOCIAL

Social relations are pervasively available and the social property of social media data determines that social media data is substantially different from data in traditional data mining and social sciences. In this section, we discuss some unique properties of social media data. Before details, we first introduce some notations used in this article. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ and $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ be the set of n users and m items (or pieces of user generated content). We use $\mathbf{S} \in \mathbb{R}^{n \times n}$, $\mathbf{R} \in \mathbb{R}^{n \times m}$ and $\mathbf{C} \in \mathbb{R}^{m \times K}$ to denote user-user relation, user-content interaction and content-feature matrices where we extract a set of K features $\mathcal{F}$ to represent the content set $\mathcal{P}$.

Big: In social media, we have little data for each specific individual. However, the social property of social media data links individuals' data together, which provides a new type of big data. For example, more than 300 million tweets are sent to Twitter per day; more than 3,000 photos are uploaded to Flickr per minute, and more than 153 million blogs are posted per year.

Linked: The availability of social relations determines that social media data is inherently linked [52]. An illustration example is shown in Figure 2 where user generated content (or p_1 to p_8) are linked via social relations among users (u_1 to u_4). Linked social media data is patently not independent and identically distributed, which contradicts one of the most enduring and deeply buried assumptions of traditional data mining and machine learning methods [23; 57].

Noisy: A successful data mining exercise entails extensive data preprocessing and noisy removal as "garbage in and garbage out." However, social media data can contain a large portion of noisy data. Users in social media can be both passive content consumers and active content producers, causing the quality of user generated content to vary drastically [1]. The noisy issues of social media data are not stop here. The social networks in social media are also noisy. First some social media users work as spammers to spread malicious or unwanted messages [47]. Second, the low cost of link formation leads to acquaintances and best friends mixed together [65].

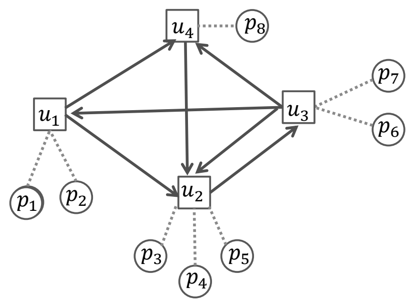


Figure 2: Linked Social Media Data.

Unstructured: User generated content in social media is often highly unstructured. Nowadays more and more users use their mobile devices to publish content such as updating statuses in Facebook, sending tweets in Twitter and commenting on posts, which results in (1) short texts and (2) typos and spacing errors occurring very frequently [25]. Free-from languages are widely adopted by social media users in the online communication such as ASCII art (e.g., :) and :( ) and abbreviations (e.g., h r u?) [46]. The short and highly unstructured social media data challenges the vast majority of existing techniques.

Incomplete: Users' attributes are predictable with their personal data [26]. To address such privacy concerns, social media services often allow their users to use their profile settings to mark their personal data such as demographic profiles, status updates, lists of friends, videos, photos, and interactions on posts, invisible to others. For example, a very small portion of Facebook users ($< 1\%$) make their personal data public available [41]. The available social media data could be incomplete and extremely sparse. For example, for social recommendation, more than 99% of entities in the user-content interaction matrix $\mathbf{R}$ are missed [51].

3. SOCIAL THEORIES

Social theories from social sciences are useful to explain various types of social phenomena. In social media, it is increasingly possible for us to observe social data from hundreds of millions of individuals. Given its large-scale size and social property, a natural question is - *can we apply social theories to social media data?*. More and more social theories have been proven to be applicable to social media data. In this section, we concentrate on three important social theories with basic concepts, ways to verify them and key findings.

3.1 Social Correlation Theory

Social correlation theory is one of the most important social theories and it suggests that there exist correlations between behaviors or attributes of adjacent users in a social network. Homophily, influence and confounding are three major social process to explain these correlations as shown in Figure 3.

- Homophily is to explain our tendency to connect to others that share certain similarity with us. For example, birds of a feather flock together.

- Influence suggests that people tend to follow the behaviors of their friends and adjacent users are likely to exhibit similar behaviors. For example, if most of one's friends switch to a mobile phone company, he could be influenced by them and switch, too

- Confounding is a correlation between users that can also be forged due to external influences from envi-

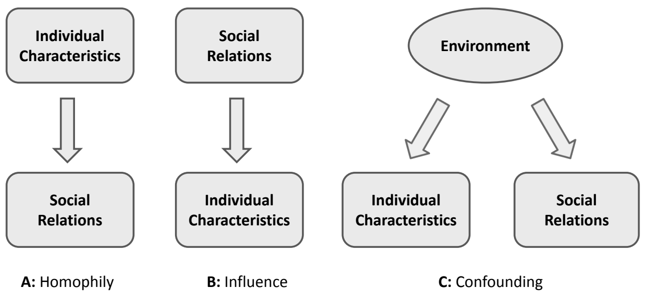


Figure 3: Major Social Forces of Social Correlation.

ronment. For example, two individuals living in the same city are more likely to become friends than two random individuals.

To help us verify the applicability of social correlation theory to social media data, essentially we need to answer the following question - *are users with social relations more similar than these without?* To answer this question, for each social relation from u_i to u_j, we calculate two similarities s_{ij} and r_{ik} where s_{ij} is the similarity between u_i and u_j, while r_{ik} is the similarity between u_i and a randomly chosen user u_k who does not connect to u_i. Let $\mathcal{S}$ be the set of s_{ij}s, which denotes the set of similarities of pairs of connected users. Let $\mathcal{R}$ be the set of r_{ik}s, which represents the set of similarities of pairs of randomly chosen users. We perform a t-test on $\mathcal{S}$ and $\mathcal{R}$. The null hypothesis is that similarities with social relations are no larger than these without, i.e., $H_0 : \mathcal{S} \leq \mathcal{R}$; the alternative hypothesis is that the similarities with social relations are larger than these without, i.e., $H_1 : \mathcal{S} > \mathcal{R}$. If there is strong evidence to reject the null hypothesis, we verify that social correlation theory is applicable to social media data.
Via above verification process, social correlation theory has been proven to be applicable to various social media sites. Twitter users with following relations are likely to share similar topics or opinions [63; 20]. Users in Epinions with trust relations are likely to rate same items with similar scores [49]. In [52], we shows that users in Digg and BlogCategory with social relations are likely to joint groups of similar interests. In location-based social networks such as Foursquare, users with social relations are likely to do check-ins in the same locations [69; 14].

3.2 Balance Theory

In general, balance theory implies the intuition that "the friend of my friend is my friend" and "the enemy of my enemy is my friend" [17]. Basically, it considers the balance of signs on a triad involving three users in a social network with positive and negative links [28; 27]. We use s_{ij} to denote the sign of the relation between u_i and u_j where $s_{ij} = 1$ (or $s_{ij} = -1$) if we observe a positive relation (or a negative relation) between u_i and u_j. Balance theory suggests that a triad $\langle u_i, u_j, u_k \rangle$ is balanced if

- $s_{ij} = 1$ and $s_{jk} = 1$, then $s_{ik} = 1$; or
- $s_{ij} = -1$ and $s_{jk} = -1$, then $s_{ik} = 1$.

For a triad $\langle u_i, u_j, u_k \rangle$, there are four possible sign combinations **A**(+,+,+), **B**(+,+,-) **C**(+,-,-) and **D**(-,-,-) as shown in Figure 4, while only **A**(+,+,+) and **C**(+,-,-) are balanced. The way to verify balance theory is straightforward. We examine all these triads $\langle u_i, u_j, u_k \rangle$ and then to check the

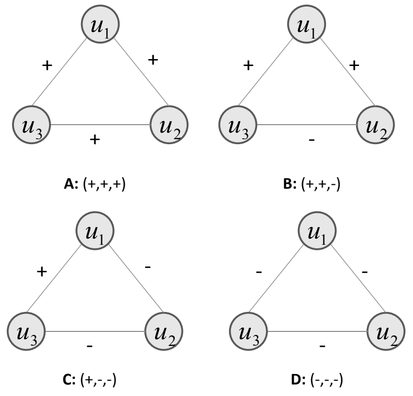


Figure 4: An Illustration of Balance Theory.

ratio of **A**(+,+,+) and **C**(+,-,-) among all four possible sign combinations. A high ratio suggests that balance theory is applicable to social media data
We check the distributions of four possible sign combinations on the three widely used social media datasets (i.e., Epinions, Slashdot and Wikipedia) with signed networks in [27]. The ratios of **A**(+,+,+) and **C**(+,-,-) among all four possible sign combinations are 0.941, 0.912, and 0.909 in Epinions, Slashdot and Wikipedia, respectively. More than 90% of triads are balanced. Similar observations on other social media datasets are reported by [68; 56]. Note that balance theory is developed for undirected social networks and we usually ignore their directions when applying balance theory to directed social networks [27].

3.3 Status Theory

Different from balance theory, status theory is developed for directed social networks [28]. Social status refers to the position or rank of a user in a social community, and represents the degree of honor or prestige attached to the position of each individual. In status theory, a positive link from u_i to u_j indicates that u_i has a higher status than u_j; while a negative link from u_i to u_j indicates that u_i has a lower status than u_j. For a triad, status theory suggests that if we take each negative relation, reverse its direction, and flip its sign to positive, then the resulting triangle (with all positive edge signs) should be acyclic.
In [28], contextualized links are introduced to verify status theory. A contextualized link is defined to be a triple $\langle u_i, u_j, u_k \rangle$ with the property that a link forms from u_i to u_j after each of u_i and u_j already has a link either to or from u_k. The link between u_k and u_i can go in either direction and have either sign yielding four possibilities, and similarly for the link between u_k and u_j; hence overall there are $4 \times 4 = 16$ different types of contextualized links. Figure 5 demonstrates 4 of 16 types of contextualized links where (**A**) and (**D**) satisfy the status theory, while (**B**) and (**C**) do not satisfy the status theory. For each of these types of contextualized links, we can count frequencies of positive versus negative links for the links from u_i to u_j and then calculate the ratio of contextualized links satisfying status theory.
In [53], it is reported that 99% of triads in the Enron email social network and the advisor-advisee social network satisfy status theory. Similar patterns are observed on Epinions and Wikipedia datasets in [28].

3.4 Discussion

The scale and properties of social media data substantially

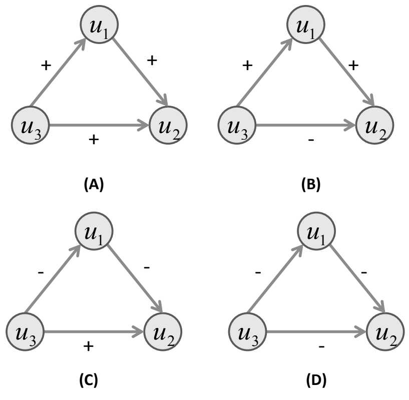


Figure 5: An Illustration of Four Out of Sixteen Types of Contextualized Links for Status Theory. Note that "+"(or "-") denotes the target node has a higher (or lower) status than the source node.

differ from these of data used by social sciences to develop social theories. Since social media data is a new type of social data, it is possible to apply some social theories to explain phenomena in social media data. The verification of social theories in social media data not only paves a way for us to understand social media data from a social perspective but also suggests that it is highly possible to facilitate social media mining tasks by integrating social theories with computational methods.

4. SOCIAL THEORIES IN SOCIAL MEDIA MINING TASKS

Social media mining is an emerging discipline under the umbrella of data mining and grows rapidly in recent years [70]. The verification of some social theories in social media data suggests that we should put "social" in social media mining and encourages a large body of literature to model and exploit social theories to advance social media mining tasks. In general, there are three types of objects in social media data - users, social relations and user generated content, which allows us to roughly classify social media mining tasks to three groups based on the mining objects - user-based tasks, relation-based tasks and content-based tasks. Next we elaborate each group with representative tasks with their definitions, challenges and the start-of-the-art algorithms to apply social theories to these specific tasks.

4.1 Social Theories in User-Related Tasks

For individuals, a better understanding of their social networks can help them share and collect reliable information more effective and efficient. For social media service providers, a better understanding of their customers can help them provide better services. User-related tasks provide necessary and effective means to understand social media users. In this subsection, we review social theories in some key user-related tasks.

4.1.1 Community Detection

Communities in social media can be explicit such as Yahoo! Groups. However, in many social media sites, communities are implicit and their members are obscure to social media users. Community detection is proposed to find these implicit communities in social media by identifying groups of users that are more densely connected to each other than to the rest of the network [55]. Detecting implicit communities can benefits many social media mining tasks such as social targeting and personalization. The major difference between clustering in data mining and community detection is that in community detection, individuals are connected to others in social networks; while in clustering, data points are not embedded in a network and they are assume to be independent and identically distributed. Formally, for a social network $G(\mathcal{U}, \mathbf{S})$, community detection is to find a set of communities $\mathcal{C}$ where users are more densely connected within a community than to the rest of users.

Homophily suggests that similar users are likely to be linked, and influence indicates that linked users will influence each other and become more similar. The suggestions from social correlation theory in creating new ties based on the similarity gives rise to macro patterns of associations, also known as communities [7]. Two users in the same community have higher similarity [44]. The modularity maximization method is to maximize the sum of the actual number of social relations between two users minus expected number of social relations between them since two users in the same community should have a higher probability to establish a relation than two randomly chosen users [43]. Wang et al. [60] find that users within the community are likely to share similar tags in social tagging systems and they take advantage of the bipartite network between users and tags in social tagging systems to discover these overlapping communities. In [66], a density-based framework is proposed with the intuition that users in the same community should interact more frequently with each other.

Recently applying balance theory to detect communities from signed networks has attracted increasing attention. In [11], a generalized balance theory is proposed where a network is k-balanced iff users can be partitioned in to k-subsets such that positive links lie within the sets and the negative links between them. Balance theory suggests that the assignment of users related by negative links should be done the opposite way of positive links, with negative links sparse within and more dense between communities therefore the potts model is extended to incorporate both positive and negative links to detect communities in signed network [58]. In [2], a two-objective approach is proposed for community detection in signed networks based on balance theory. One is that the partitioning should have dense positive intra-connections and sparse negative interconnections, and the other is that it should have as few as possible negative intra-connections and positive inter-connections.

4.1.2 User Classification

Due to privacy concerns, social media users tend to hide their profiles. For social media service providers, users' profile information is useful for them to customize their services to the users in many ways such as friend and content recommendations and personalized search. More they know about users and their preferences, better they can serve them. Given a social network and some user information (attributes, preferences or behaviors), user classification is designed to infer the information of other users in the same network [15]. In the user classification problem, users in $\mathcal{U}$ are partially labeled as $\mathcal{U} = [\mathcal{U}^L, \mathbf{U}^U]$ where $\mathcal{U}^L$ and $\mathbf{U}^U$ are the sets of labeled and unlabeled users, respectively. Formally the task of user classification is to label users from a finite set of categorical values in $\mathbf{U}^U$ with the social network $G(\mathcal{U}, \mathbf{S})$ and $\mathcal{U}^L$.

Social correlation theory suggests that the labels of linked users should be correlated, which is the major reason why researchers believe that the labels of $\mathcal{U}^L$ can be predicted with the network structure and the partially labeled users [15]. Social correlation theory is the underlying assumption of most of existing user classification methods, which design algorithms for collective classification. A typical user classification algorithm includes parts of the three components [37]:

- A local classifier - it is used for initial label assignment;
- A relational classifier - it learns a classifier from the labels of its neighbors to the label of one user suggested by social correlation theory; and
- Collective classification - it applies relational classifier to each node iteratively until the inconsistency between neighboring labels is minimized.

In [36], a weighted-vote relational neighborhood classifier wvRN is introduced for user classification. wvRN is like a lazy learner and estimates the labels of users as the weighted mean of their neighbors. In [34], the proposed framework first creates relational features of one user by aggregating the label information of its neighbors and then a relational classifier can be constructed based on labeled data. Neville and Jensen in [42] propose to use clustering algorithms to find out the cluster memberships of each user first, and then fix the latent group variables for later inference. Xiang et al. [64] propose a novel latent relational model based on copulas. It can make predictions in a discrete label space while ensuring identical marginals and at the same time incorporating some desirable properties of modeling relational dependencies in a continuous space. A community-based framework is proposed in [54]. It first extracts overlapping communities based on social network structure, then uses communities as features to represent users and finally a traditional classifier such as SVM is trained to assign labels for unlabeled users in the same network.

4.1.3 Social Spammer Detection

Social media has become an important and efficient way to disseminate information. Given its popularity and ubiquity, social spammers create many fake accounts and send out unsolicited commercial content [62]. Social spammers have become rampant and the volume of spam has increased dramatically. For example, 83% of the users of social networks have received at least one unwanted friend request or message [47]. This not only causes misuse of communication bandwidth, storage space and computational power, but also wastes users' time and violates their privacy rights. Therefore developing effective social spammer detection techniques is critically important in improving user experience and positively affecting the overall value of social media services [47]. Given a social network $G(\mathcal{U}, \mathbf{S})$, social spammer detection is to find a set of spammers $\mathcal{U}^S$ from $\mathcal{U}$ with $\mathcal{U}^S \subset \mathcal{U}$.

Based on social correlation theory, there are two observations for normal users and spammers [73]. First normal users perform similarly with their neighbors. Second, spammers perform differently from their neighbors since most of their neighbors are normal users. Therefore a social regularization term is proposed under the matrix factorization framework to model these observations where two connected normal users should be close in the latent space since they share similar interests and may perform similar social activities, while spammers should be far away from their neighbors in the latent space. In Twitter, users have directed following relations and spammers can easily follow a large number of normal users within a short time. In [19], we divide user-user following relations in Twitter into four types - [spammer, spammer], [normal, normal], [normal, spammer], and [spammer, normal]. Since the fourth relation can be intentionally faked by spammers, we only consider the first three types of relations. Specifically we introduce a graph regularization term to model social correlation theory in the directed social relations, which is integrated into the standard Lasso formulation to train a linear classification for social spammer detection. Spammers and normal users have very different social behaviors. Normal users are likely to form a group with other normal users, while spammers are likely to from spammer groups [29]. In [6], the authors incorporate community-based features of users with basic topological features to improve spammer classifiers. It first finds overlapping community structure of users and then extracts features based on these communities such as the features which express the role of a user in the community structure like a boundary node or a core node and the number of communities it belongs to.

4.2 Social Theories in Relation-Related Tasks

A social network is usually represented by a binary adjacent matrix. First the matrix is extremely sparse since there are many pairs of users with missing relations. Second, social networks in social media are more complicated. For example, strengths of relations might be heterogeneous such as acquaintances and best friends, while a social network may a composite of various types of relations such as family, classmates and colleagues. Relation-Related tasks focus on mining relations among users and aim to reveal a fine-grained and comprehensive view of social relations. Signed networks arise in social network with various ways when users can implicitly or explicitly tag their relationship with other users as positive or negative. In this section, we review social theories in some key relation-related tasks on signed and unsigned networks.

4.2.1 Link Prediction

It is critical for social media sites to provide services to encourage more user interactions with better experience such as expanding one's social network. One effective way is to automatically recommend connections since it is hard for users to figure out who is available on social media sites. Most social media sites provide friend recommendation services to their customers such as Facebook, Twitter and LinkedIn. The essential problem of friend recommendation is known as link prediction [30]. When there is no relation between u_i and u_j, $\mathbf{S}_{ij} = 0$. The task of link prediction is to predict which pairs of users u_i and u_j without relations $\mathbf{S}_{ij} = 0$ are likely to get connected given a social network $G(\mathcal{U}, \mathbf{S})$.

Unsigned Networks : Homophily in social correlation theory suggests that similar users are likely to establish social relations. In [30], various similarity measurements such as common neighbors based on the network structure are reviewed for link prediction. One challenging problem in link prediction is the sparsity problem - some users may have very few or even no links. In [49], a low-rank matrix factorization framework with homophily effect hTrust is proposed to predict trust relations. Homophily coefficients are defined to measure the strength of homophily among users. The stronger homophily between two users is, the smaller distance between them in the latent space is. Homophily regularization is then defined to model homophily effect by controlling users' distances in the latent space with the help of homophily coefficients. Through homophily regularization, trust relations can be suggested to users with few or even no relations and mitigate the sparsity problem in link prediction. The confounding effect in social correlation theory suggests that people who share high degree of overlap in their trajectories are expected to have a better likelihood of forming new links. In [59], the effect of confounding is investigated for link prediction. Specifically, it leverages mobility information to extract features which can capture some degree of closeness in physical world between two individuals. Status theory suggests new links are more likely to be attached from users with low statuses to these with high statues and the preferential attachment models are widely used to predict link prediction based on status measures such as the degree of nodes and PageRank [5].

Signed Networks : In [27], local-topology-based features (or 16 triad types) based on balance theory and status theory are extracted to improve the performance of a logistic regression classifier in signed relation prediction. In [13], the authors use a probabilistic treatment of trust combined with a modified spring-embedded layout algorithm to classify a relation based on balance theory. Instead of having all users repel, the model adds a repelling force only between users connected with a negative relation to capture balance theory. For example, one is friends with an enemy of the other; the forces will push them in different locations. In [10], the authors show how any quantitative measure of social imbalance in a network can be used to derive a link prediction algorithm and extend the approach in [27] by presenting a supervised machine learning based link prediction method that uses features derived from longer cycles in the network. The motivation to derive features from longer cycles is that higher order cycles in a signed network yield a "measure of imbalance" suggested the balance theory. In [18], it shows that the notion of weak structural balance in signed networks naturally leads to a global low-rank model for the network. Under such a model, the sign inference problem can be formulated as a low-rank matrix completion problem.

4.2.2 Social Tie Prediction

Social networks in social media can be a composite of various types of relations. For example, the relation types in Facebook could be family, colleagues, classmates and friends. However, in most online networks such as Facebook, Twitter and LinkedIn, such type information is usually unavailable [56]. Different types of relations may influence people in different ways. For example, one user's work style may be mainly influenced by her/his colleagues; while the daily life habits may be strongly affected by her/his family. It is necessary and important to reveal these different types of social relations therefore we ask whether we can automatically infer the types of social relations for social networks in social media. A novel task of social tie prediction is designed to answer the above question, which aims to predict the type of a given social relation. A non-zero value of $\mathbf{S}_{ij}$ suggests that there is a connection between u_i and u_j. Formally social tie prediction is to predict the type of a social relation between u_i and u_j with $\mathbf{S}_{ij} \neq 0$ from a finite set of categorical types such as { family, classmates, colleagues and friends}.

In [53], a framework is proposed to classify the type of social relationships by learning across heterogeneous networks. The framework incorporates social theories such as balance theory and status theory into a factor graph model, which effectively improves the accuracy of inferring the type of social relationships in a target network by borrowing knowledge from a different source network. Balance theory and status theory should be general over different types of networks. To learn knowledge from the source network to the target network, transfer features are extracted based on balance theory and status theory, which are shared by different types of networks. In particular, from social balance, the paper extracts triad based features to denote the proportion of different balanced triangles in a network; and from status theory, it defines features over triads to respectively represent the probabilities of the seven most frequent formations of triads. Different from [53], approaches are suggested by [68] to model balance theory and status theory mathematically. To model the balance theory, it introduces an one-dimensional latent factor β_i for each user u_i and defines the sign between u_i and u_j as $s_{ij} = \beta_i \beta_j$. To model status theory, it introduces a global user-independent parameter η to capture the partial ordering of users. η maps the latent user profile of u_i γ_i to a scalar quantity $\ell_i = \eta\gamma_i$, which reflects the corresponding user u_i's social status. According to status theory, it characterizes social ties from u_i to u_j by modeling the relative status difference between them as $\ell_{ij} = \ell_i - \ell_j$

4.2.3 Tie Strength Prediction

Social media users can have hundreds of social relations. However, a recent study shows that Twitter users have a very small number of friends compared to the number of followers and followees they declare [21]. The low cost of link formation in social media can lead to networks with heterogeneous relationship strengths (e.g., acquaintances and best friends mixed together) [65]. Pairs of users with strong strengths are likely to share greater similarity than those with weak strengths; therefore a better understanding of strengths of social relations can help social media sites serve their customers well such as better recommendations and more effective friend management tools, which arises the problem of tie strength prediction. In the binary relation presentation, once there is a connection between u_i and u_j, $\mathbf{S}_{ij} = 1$. The task of tie strength prediction is to predict a connection strength between 0 and 1 for u_i and u_j with $\mathbf{S}_{ij} = 1$. After tie strength prediction, the binary relation representation matrix $\mathbf{S}_{ij} \in \{0, 1\}$ will be converted into a continuous valued relation representation matrix $\mathbf{S}_{ij} \in [0, 1]$. In [24], guided by social correlation theory, four different categories of features, i.e., attribute similarity, topological connectivity, transactional connectivity, and network transactional connectivity, are extracted from sources including friendship links, profile information, wall postings, picture postings, and group memberships. Then various classifiers are trained to predict link strength from transactional information based on these extracted features. A unsupervised latent variable model is proposed to predict tie strength in online social network [65] with user profiles and interactions. One key underlying assumption of the proposed model is social correlation theory. Homophily in social correlation theory postulates that users tend to form ties with other people who have similar characteristics, and it is likely that the stronger the tie, the higher the similarity. Thus the proposed framework models the tie strength as homophily effect of nodal profile similarities. The relationship strength directly influences the nature and frequency of online interactions between a pair of users. The stronger the relationship, the higher likelihood that a certain type of interaction between the pair of users. Therefore the propose framework models the relationship strength as the hidden cause of influence among users.

4.3 Social Theories in Content-Related Tasks

Numerous techniques are developed for various content mining tasks such as classification and clustering in the last decade. User generated content in social media is usually linked, noisy, highly unstructured and incomplete, which determines that existing techniques become difficult when applying these mining tasks on user generated content in social media. Before the popularity of social media, researchers have already noticed that exploiting link information can improve content classification [72] and clustering [32]. The popularity of social media makes social relations pervasively available, which encourages the exploitation of social relations in more and more mining tasks. Social theories can help us understand social relations better and in this subsection, we review how social theories help some representative content-related tasks.

4.3.1 Social Recommendation

The pervasive use of social media generates massive data in an unprecedented rate and the information overload problem becomes increasingly serve for social media users. Recommendation has been proven to be effective in mitigating the information overload problem and presents its significance to improve the quality of user experience, and to positively impact the success of social media. Users in the physical world are likely to seek suggestions from their friends before making a purchase decision and users' friends consistently provide good recommendations [45], we have similar observations in the online worlds. For example, 66% of people on social sites have asked friends or followers to help them make a decision and 88% of links that 14-24 year olds clicked were sent to them by a friend and 78% of consumers trust peer recommendations over ads and Google SERPs[1]. These intuitions motive a new research direction of recommendation social recommendation, which aims to take advantage of social relations to improve the performance of recommendation. Formally, a social recommender system is to predict missing values in the user-content interaction matrix $\mathbf{R}$ based on information from the user-user relation matrix $\mathbf{S}$ and the observed values in $\mathbf{R}$ [51].

The major reason why people believe that social relations are helpful to improve recommendation performance is evidence from social correlation theory, which suggests that a user's preference is similar to or influenced by their directly connected friends [51]. Therefore social media users rarely make decisions independently and usually seek advice from their friends before making purchase decisions. Social relations may provide both similar and familiar evidence for users, MoleTrust uses socially connected users to replace similar users in traditional user-based collaborative filtering method for recommendation in [39]. Social correlation theory indicates that a user's preference should be similar to her/his social network. Ensemble methods predict a missing value for a given users as a linear combination of ratings from the user and her/his social network based on traditional matrix factorization CF method with the intuition that users and their social networks should have similar ratings on the same items [50]. While regularization methods add regularization terms to force the preference of a user close to that of users in her/his social network under the matrix factorization CF method. For example, SocialMF defines a regularization term to force the preference of a user to be close to the average preference of the user's social network [22], and SoReg uses social regularization to force the preferences of two connected users close [35].

4.3.2 Feature Selection

One characteristic of user generated content in social media is high-dimensional such as there are tens of thousands of terms in tweets or pixels for photos in Flickr. Traditional data mining tasks such as classification and clustering may fail due to the curse of dimensionality. Feature selection has been proven to be an effective way to handle high-dimensional data for efficient data mining [31]. As mentioned above, user generated content is linked due to the availability of social relations and poses challenges to traditional feature selection algorithms which are typically designed for IID data. The formal definition of feature selection for user generated content in social media is stated as [52] - we aim to develop a selector which selects a subset of most relevant features from $\mathcal{F}$ on the content-feature matrix $\mathbf{C}$ with its social context $\mathbf{S}$ and $\mathbf{R}$.

LinkedFS is proposed as a feature selection framework for user generated content with social context based on social correlation theory in [52]. Four types of relations, i.e., coPost, coFollowing, coFollowed and Following, are extracted from social context $\mathbf{S}$ and $\mathbf{R}$ of user generated content $\mathbf{C}$. Social correlation theory suggests that linked users are likely to share similar topics. Based on social correlation theory, LinkedFS turns these four types of relations to four corresponding hypotheses that can affect feature selection with linked data. For example, following hypothesis assumes that one user u_i follows another user u_j because u_i share u_j's interests, and their user generated content is more likely similar in terms of topics; hence LinkedFS models following relations mathematically by forcing topics of two users with following relations close to each other. LinkedFS jointly incorporates group Lasso with the regularization term to model each type of relations for feature selection.

4.3.3 Sentiment Analysis

Nowadays social media services such as Twitter and Facebook are increasingly used by online users to share and exchange opinions, providing rich resources to understand public opinions. For example, in [3], a simple model exploiting Twitter sentiment and content outperforms market-based predictors in terms of forecasting box-office revenues for movies; public mood as measured from a large-scale collection of tweets obtains an accuracy of 86.7% in predicting the daily up and down changes in the closing values of the DJIA [8]. Therefore sentiment analysis for such opinion-rich social media data has attracted increasing attention in recent years [46; 20]. Formally sentiment analysis for user generated content with social relations is to obtain a predictor from the content-feature matrix $\mathbf{C}$ with its social context $\mathbf{S}$ and $\mathbf{R}$, which can automatically label the sentiment polarity of an unseen post.

Social correlation theory indicates that sentiments of two linked users are likely to be similar. In [48], graphical models are proposed to incorporate social network information to improve user-level sentiment classification of different topics based on two observations - (1) user pairs in which at least one party links to the other are more likely to hold the same sentiment, and (2) two users with the same sentiment are more likely to have at least one link to the other than two users with different sentiment. Social correlation theory suggests that social relations are kinds of sentiment correlations. In [46], the authors propagate sentiment labels of tweets via user-user social relations $\mathbf{S}$ and user-tweet relations $\mathbf{R}$ to assign sentiment labels to unlabeled tweets. In [20], tweet-tweet correlation network are built from $\mathbf{S}$ and $\mathbf{R}$ based on social correlation theory. For example, tweets from users with following relations should be correlated as

[1]http://www.firebellymarketing.com/2009/12/social-search-statistics-fromses-chicago.html

Social Media Mining Tasks		Feature Engineering	Constraint Generating	Objective Defining
User Related	Community Detection			[42],[59],[65],[57],[2]
	User Classification			[35],[33],[41],[41],[63],[53]
	Spammer Detection	[28],[6]	[72],[18]	
Relation Related	Link Prediction	[29],[58],[26],[10]	[48]	[5],[12],[17]
	Social Tie Prediction	[52]		[67]
	Tie Strength Prediction	[23]		[64]
Content Related	Recommendation		[21],[23]	[38]
	Feature Selection		[51]	
	Sentiment Analysis		[47],[19]	[45]

Figure 6: Social Theories in Social Media Mining.

suggested by social correlation theory. Two tweets linked in the tweet-tweet correlation network are likely to share similar sentiments; hence the proposed framework SANT adds a graph regularization term in the Lasso classifier to force the sentiments of two correlated tweets close to each other.

4.4 Discussion

In reviewing state-of-the-art algorithms that exploit social theories in mining social media, we understand that they aim to find mathematical explanations of social theories for computational models. We notice that algorithms share similar ways in applying social theories such as feature engineering, constraint generating and objective defining.

- *Feature Engineering:* It uses social theories to extract features for computational models. For example, in link prediction, confounding effect in social correlation theory suggests that people who are physically close have a better likelihood of forming new links and new features from users' mobility information are extracted in [59] to improve link predilection; while triad features based on status theory are extracted as transfer features to infer social ties by transferring knowledge from the source network to the target network [53].

- *Constraint Generating:* It generates constraints from social theories for computational models. Regularization is one of the most popular ways to implement constraint generating. For example, SocialMF in social recommendation adds a social regularization term to force the performance of a user close to that of her/his social network to capture social correlation theories [22]; and hTrust adds a homophily regularization term to capture homophily effect and mitigate the sparsity problem in link prediction [49].

- *Objective Defining:* It uses social theories to define the objectives of the computational models. For example, two objectives are defined from balance theory to detect communities in signed networks [2]; and the user classification task is to make the labels of a user similar to these of her/his social network [15].

Instead of brute-force search, social theories can guide us to extract relevant features via feature engineering, to generate constraints via constraint generating, and to define objectives via objective defining for computational models. The algorithms reviewed earlier that exploit social theories in various social media mining tasks are summarized in Figure 6. We notice that for the same task, social theories can be exploited in different ways. For example, for link prediction, social theories are explained via feature engineering, constraint generating and objective defining.

5. OPEN ISSUES AND FUTURE RESEARCH DIRECTIONS

5.1 More "Social" in Mining Social Media Data

Some social theories have been proven to be applicable to social media data, which encourages us to put "social" in social media mining. Integrating some social theories with computational models advances various social media mining tasks and has attracted increasing attention. The exciting progress not only proves that the direction of integrating social theories in mining social media data is appealing but also suggests that we should put more "social" in social media mining. In this article, we review the state-of-the-art algorithms that employ social correlation theory, balance theory and status theory in various social media mining tasks. These theories are just illustrative examples and there could be more social theories to be applicable and employed such as small world theory [74] as shown in recent efforts to investigate and verify more social theories for social media data. Some of these efforts have already made initial progress such as structural hole theory [9] and weak tie theory [16]. A person is said to span a structural hole in a social network if he or she is linked to people in parts of the network that are otherwise not well connected to one another [9]. Tang et al. [56] employ structural hole theory in the problem of social tie prediction; while Lou and Tang confirm the importance of structural hole in information diffusion with social media data, and show that mining structural hole can benefit various social media mining tasks such as community detection and link prediction [33]. Weak tie theory suggests that more novel information flows to individuals through weak rather than strong ties [16]. Recently researchers find that weak ties of a user are helpful to predict the preference of the user for user classification [54] and social recommendation [71].

5.2 New Social Theories

No doubt that social media data is a new type of social data and is much more complicated than the data social sciences use to study social theories. It is highly possible that new social theories can be discovered from social media data to make meaningful progress on important problems in social media mining, however, that progress requires serious engagement of both computer scientists and social scientists [61]. Data availability is still a challenging problem for social scientists. The data required to address many problems of interest to social scientists remain difficult to assemble and it has been impossible to collect observational data on the scale of hundreds of millions, or even tens of thousands, of individuals [61]. Social media provides a virtual world for users' online activities and makes it possible for social scientists to observe social behavior and interaction data of hundreds of millions of users. However social media data is too big to be directly handled by social scientists. On the other hand, computer scientists can employ data mining and machine learning techniques to handle big social media data; but, we lack necessary theories to help us understand social media data better. For example, without a better understanding of social media data, computer scientists may waste a lot of time in feature engineering, which is the key to the success of many real-world applications [12]. Therefore engagement of both computer scientists and social scientists in social media data is truly mutually beneficial. Computer scientists can take advantage of social theories to mine social media data and provide computational tools that are of great potential benefit to social scientists; while social scientists can make use of computational tools to handle social media data and develop new social theories to help computer scientists provide better computational tools.

6. CONCLUSION

The social nature of social media data calls for new techniques and tools and cultivates a new field - social media mining. Social theories from social sciences have been proven to be applicable to mining social media. Integrating social theories with computational models is becoming an interesting way in mining social media data and makes exciting progress in various social media mining tasks. In this article, we review three key social theories, i.e., social correlation theory, balance theory and status theory, in mining social media data. In detail, we introduce basic concepts, verification methods, interesting findings and the state-of-the-art algorithms to exploit these social theories in social media mining tasks, which can be categorized to feature engineering, constraint generating and objective defining. As future directions, more existing social theories could be employed or new social theories could be discovered to advance social media mining.

7. REFERENCES

[1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, 2008.

[2] A. Amelio and C. Pizzuti. Community mining in signed networks: a multiobjective approach. In *ASONAM*, 2013.

[3] S. Asur and B. A. Huberman. Predicting the future with social media. In *WI-IAT*, 2010.

[4] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *WWW*, 2012.

[5] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 1999.

[6] S. Y. Bhat and M. Abulaish. Community-based features for identifying spammers in online social networks. In *ASONAM*, pages 100–107. ACM, 2013.

[7] H. Bisgin, N. Agarwal, and X. Xu. Investigating homophily in online social networks. In *WI-IAT*, 2010.

[8] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[9] R. S. Burt. *Structural holes: The social structure of competition*. Harvard university press, 2009.

[10] K.-Y. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon. Exploiting longer cycles for link prediction in signed networks. In *CIKM*, 2011.

[11] J. A. Davis. Clustering and structural balance in graphs. *Human relations*, 1967.

[12] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 2012.

[13] T. DuBois, J. Golbeck, and A. Srinivasan. Predicting trust and distrust in social networks. In *socialcom*, 2011.

[14] H. Gao, J. Tang, and H. Liu. Exploring social-historical ties on location-based social networks. In *ICWSM*, 2012.

[15] L. Getoor and C. P. Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 2005.

[16] M. Granovetter. The strength of weak ties. *JSTOR*, 1973.

[17] F. Heider. Attitudes and cognitive organization. *The Journal of psychology*, 1946.

[18] C.-J. Hsieh, K.-Y. Chiang, and I. S. Dhillon. Low rank modeling of signed networks. In *KDD*, 2012.

[19] X. Hu, J. Tang, Y. Zhang, and H. Liu. Social spammer detection in microblogging. In *IJCAI*, 2013.

[20] X. Hu, L. Tang, J. Tang, and H. Liu. Exploiting social relations for sentiment analysis in microblogging. In *WSDM*, 2013.

[21] B. Huberman, D. M. Romero, and F. Wu. Social networks that matter: Twitter under the microscope. *First Monday*, 2008.

[22] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Recsys*, 2010.

[23] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*, 2002.

[24] I. Kahanda and J. Neville. Using transactional information to predict link strength in online social networks. In *ICWSM*, 2009.

[25] D. Kim, D. Kim, E. Hwang, and S. Rho. Twittertrends: a spatio-temporal trend detection and related keywords recommendation scheme. *Multimedia Systems*,2014.

[26] M. Kosinski, D. Stillwell, and T. Graepel. Private traits and attributes are predictable from digital records of human behavior. *PNAS*, 2013.

[27] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, 2010.

[28] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *CHI*, 2010.

[29] F. Li and M.-H. Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *CEAS*, 2006.

[30] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *JASIST*, 2007.

[31] H. Liu and H. Motoda. *Computational methods of feature selection*. CRC Press, 2007.

[32] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML*, 2006.

[33] T. Lou and J. Tang. Mining structural hole spanners through information diffusion in social networks. In *WWW*, 2013.

[34] Q. Lu and L. Getoor. Link-based classification. In *ICML*, 2003.

[35] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, 2011.

[36] S. A. Macskassy and F. Provost. A simple relational classifier. In *MRDM*, 2003.

[37] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *JMLR*, 2007.

[38] P. Massa. A survey of trust use and modeling in real online systems. *Trust in E-services: Technologies, Practices and Challenges*, 2007.

[39] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *CoopIS, DOA, and ODBASE*, 2004.

[40] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 2001.

[41] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *WSDM*, 2010.

[42] J. Neville and D. Jensen. Leveraging relational autocorrelation with latent group models. In *MRDM*, 2005.

[43] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *PRE*, 69(2):026113, 2004.

[44] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos. Community detection in social media. *DMKD*, 2012.

[45] R. R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends. In *DELOS*, 2001.

[46] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In *ULNLP*, 2011.

[47] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *ACSAC*, 2010.

[48] C. Tan, L. Lee, J. Tang, L. Jiang, M. Zhou, and P. Li. User-level sentiment analysis incorporating social networks. In *KDD*, 2011.

[49] J. Tang, H. Gao, X. Hu, and H. Liu. Exploiting homophily effect for trust prediction. In *WSDM*, 2013.

[50] J. Tang, H. Gao, and H. Liu. mtrust: discerning multifaceted trust in a connected world. In *WSDM*, 2012.

[51] J. Tang, X. Hu, and H. Liu. Social recommendation: a review. *SNAM*, 2013.

[52] J. Tang and H. Liu. Feature selection with linked data in social media. In *SDM*, 2012.

[53] J. Tang, T. Lou, and J. Kleinberg. Inferring social ties across heterogenous networks. In *WSDM*, 2012.

[54] L. Tang and H. Liu. Relational learning via latent social dimensions. In *KDD*, 2009.

[55] L. Tang and H. Liu. Community detection and mining in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2010.

[56] W. Tang, H. Zhuang, and J. Tang. Learning to infer social ties in large networks. In *PKDD*, 2011.

[57] B. Taskar, P. Abbeel, M.-F. Wong, and D. Koller. Label and link prediction in relational data. In *SRL*, 2003.

[58] V. Traag and J. Bruggeman. Community detection in networks with positive and negative links. *PRE*, 80(3):036115, 2009.

[59] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi. Human mobility, social ties, and link prediction. In *KDD*, 2011.

[60] X. Wang, L. Tang, H. Gao, and H. Liu. Discovering overlapping groups in social media. In *ICDM*, 2010.

[61] D. J. Watts. Computational social science: Exciting progress and future directions. *Winter Issue of The Bridge on Frontiers of Engineering*, 2014.

[62] S. Webb, J. Caverlee, and C. Pu. Social honeypots: Making friends with a spammer near you. In *CEAS*, 2008.

[63] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM*, 2010.

[64] R. Xiang and J. Neville. Collective inference for network data with copula latent markov networks. In *WSDM*, pages 647–656. ACM, 2013.

[65] R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In *WWW*, 2010.

[66] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger. Scan: a structural clustering algorithm for networks. In *KDD*, 2007.

[67] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In *ICDM*, 2010.

[68] S.-H. Yang, A. J. Smola, B. Long, H. Zha, and Y. Chang. Friend or frenemy?: predicting signed ties in social networks. In *SIGIR*, 2012.

[69] M. Ye, X. Liu, and W.-C. Lee. Exploring social influence for recommendation: a generative model approach. In *SIGIR*, 2012.

[70] R. Zafarani, M. A. Abbasi, and H. Liu. *Social Media Mining: An Introduction*. Cambridge University Press, 2014.

[71] X. Zhang, J. Cheng, T. Yuan, B. Niu, and H. Lu. Toprec: domain-specific recommendation through community topic mining in social network. In *WWW*, 2013.

[72] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In *SIGIR*, 2007.

[73] Y. Zhu, X. Wang, E. Zhong, N. N. Liu, H. Li, and Q. Yang. Discovering spammers in social networks. In *AAAI*, 2012.

[74] D. Watts, and S, Steven. Collective dynamics of 'small-world'networks. In *nature*, 1998.

Brain Network Analysis: a Data Mining Perspective

Xiangnan Kong
Department of Computer Science
University of Illinois at Chicago
Chicago, USA
xkong4@uic.edu

Philip S. Yu
Department of Computer Science
University of Illinois at Chicago
Chicago, USA
psyu@cs.uic.edu

ABSTRACT

Following the recent advances in neuroimaging technology, the research on brain network analysis becomes an emerging area in data mining community. Brain network data pose many unique challenges for data mining research. For example, in brain networks, the nodes (i.e., the brain regions) and edges (i.e., relationships between brain regions) are usually not given, but should be derived from the neuroimaging data. The network structure can be very noisy and uncertain. Therefore, innovative methods are required for brain network analysis. Many research efforts have been devoted to this area. They have achieved great success in various applications, such as brain network extraction, graph mining , neuroimaging data analysis. In this paper, we review some recent data mining methods which are used in the literature for mining brain network data.

Keywords

Brain networks, Graph mining, Functional magnetic resonance imaging, Subgraph Patterns

1. INTRODUCTION

Recent advances in neuroimaging technology has unleashed a torrent of neuroscience data. The data give us an unprecedented opportunity to look into the activity and connectivity of the human brain non-invasively and in vivo. Brain tissue is generally described according to the broad classes of gray matter and white matter. This distinction has a historical basis in the appearance at dissection, reflecting the preponderance of cell bodies and dendrites (gray matter, e.g. cortex) and of myelinated axons, which have a fatty, whitish appearance (white matter, e.g. corpus callosum) [29]. The activity of the brain, however, is organized according to vast and complex patterns of connectivity involving multifocal distributed neural networks and white matter pathways that are considered critical to an understanding of higher order cognitive function and dysfunction [42].

The last several decades have witnessed explosive expansion in knowledge concerning the structure and function of the human brain. This can be attributed in part to advances in Magnetic Resonance (MR) imaging capabilities. Techniques such as diffusion tensor imaging (DTI), for example, that can be used for in vivo interrogation of the brain at levels that approximate cellular dimensions, have enabled tractography of the vast network of white matter fiber pathways, yielding fundamental insights into structural connectivity [5; 8; 27; 28; 2]. Functional magnetic resonance imaging (fMRI) has been used to identify localized patterns of brain activation on the basis of cerebral blood flow and the BOLD response [30; 31; 6]. Strategies, such as resting state fMRI (rs-fMRI) have been used to map functional connectivity – networks defined on the basis of correlated activity in low frequency oscillations between gray matter regions [6; 17; 33].

Brain networks have been studied extensively in recent years [36; 11], because of potential application to detection of a wide variety of brain diseases [43]. A detailed book on this topic may be found in [35]. Conventional research on brain networks focuses on connectivity derived from various neuroimaging modalities, such as electroencephalography (EEG), fMRI and DTI. These approaches emphasize creating a network among brain regions (or ROIs) and detecting changes in connectivity related to brain diseases. Conventional strategies often use either equally sized regions or anatomical landmarks (e.g., gyral and sulcal-based regions) as nodes of the network with unclear relationship to the underlying functional and structural organization of the brain. The links among these regions (e.g., functional connections or structural connections) are extracted based on neuroimaging data to form a network.

Data mining has already made significant impacts in many real-world applications in industry and science, such as social network analysis, web mining. However, brain network data pose many unique challenges for data mining community. For example, in conventional network analysis, the nodes (e.g., webpages) and edges (e.g., hyperlinks) are usually clearly defined. In brain networks, however, the nodes (i.e., the brain regions) and edges (i.e., relationships between brain regions) are not given, but need to be derived from the neuroimaging data. Different parcellation of the brain regions can result in significantly different network structures. The edges of brain networks are also highly uncertain due to the noise in imaging signals. Conventional data mining methods can seldom be directly applied to brain network data. A wide variety of new questions can also be asked in context of the brain network analysis.

In this paper, we focus on reviewing some recent data mining methods for (1) direct mining of neuroimaging data; (2) extracting brain networks from neuroimaging data; (3) mining subgraph patterns from brain networks.

Imaging-based Approaches: Neuroimaging data can naturally represented as tensor data [47; 14; 18], which gen-

Table 1: Properties of Data Mining Methods for Neuroimaging Data.

Publication	**Target Disease**	**Imaging Data**	**Mining Task**	**Information Sources**
KDD'08 [47]	Alzheimer's	MRI, CSF	Data Fusion	Neuroimage, CSF, Blood Markers
KDD'09 [37]	Alzheimer's	FDG-PET	Region Connectivity	Neuroimage
NIPS'09 [19]	Alzheimer's	PET	Region Connectivity	Neuroimage
KDD'11 [20]	Alzheimer's	FDG-PET	Effective Connectivity	Neuroimage
NIPS'11 [22]	Alzheimer's	PET, MRI	Region Identification	Neuroimage
SDM'13 [24]	AD, ADHD, HIV	fMRI	Subgraph Patterns Mining	Functional Brain Network
KDD'13a [14]	Alzheimer's	fMRI	Network Discovery	Neuroimage
KDD'13b [45]	Alzheimer's	MRI, PET, CSF	Multi-source Learning	Neuroimage, CSF, Proteomics
SDM'14 [18]	AD, ADHD, HIV	fMRI	Supervised Tensor Learning	Neuroimage

eralize the conventional vector/matrix data models. An MRI sample corresponds a 3D-array, or three-mode tensor. An fMRI sample can be represented as (3D $\times$ time) a 4-dimensional array, or four-mode tensor. Other imaging data, such as DTI, can be represented as tenors with even higher modes. Ideal data mining methods for neuroimaging data should be able to handle the extremely high dimensionality within the data, while preserving the tensor structure in the model [18].

Brain Network Extraction: Another important challenge in brain network analysis is that the network structure is very difficult to extract. In order to extract a meaningful brain network, the nodes and the edges of the networks should both be carefully extracted from neuroimaging data. Many research efforts are devoted to mining important brain regions [22; 14] and connection estimation [37; 19; 20].

Brain Network Analysis: Once the brain networks are constructed from the neuroimaging data, the next step is to analyze the networks. The challenges are that conventional network measures are optimally suited for binary/certain networks and are less well suited for weighted, uncertain, signed networks. Ideal data mining methods for brain network analysis should be able to take the unique properties on the edges (e.g., uncertainty, weights) into consideration. In some applications, we need to extract important connectivity patterns from brain networks. For example, in the classification task of brain networks, we need to extract discriminative subgraphs from the brain networks to figure out the differences among different groups of subjects: Alzheimer's patient (AD) and normal controls (NC). These subgraphs are expected to be most discriminative and reliable in the uncertain brain networks.

To summarize, we create a table of the different methods, and the different properties such as target disease, mining tasks or what type of information sources were used. This is provided in Table 1.

The rest of the paper is organized as follows. We review tensor-based neuroimaging analysis in Section 2. The network extraction is presented in Section 3. We discuss brain network analysis in Section 4. Finally, we conclude the paper in Section 5.

2. IMAGING-BASED APPROACHES

2.1 Tensor-based Modeling

Neuroimaging data can naturally represented as tensor data [47; 14; 18]. A *tensor* is a high order generalization of a vector (first-order tensor) and a matrix (second-order tensor), which is also known as multidimensional array. An m-th order tensor can be represented as $\mathcal{A} = (a_{i_1,i_2,\cdots,i_m}) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_m}$. I_i is the dimension of $\mathcal{A}$ along the i-th mode. In the context of neuroimaging data, each fMRI sample can be represented as (3D $\times$ time) a 4-dimensional array, or four-mode tensor. An MRI sample corresponds a 3D-array, or three-mode tensor. Other imaging data, such as DTI, can be represented as tenors with even higher modes.

Based on the above definition, inner product, tensor norm, and tensor product can be defined as follows:

Inner product: the inner product of two same-sized tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_m}$ is

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_m=1}^{I_m} a_{i_1,i_2,\ldots,i_m} b_{i_1,i_2,\ldots,i_m}.$$

Tensor norm: the norm of a tensor $\mathcal{A}$ is

$$\|\mathcal{A}\|_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle} = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_m=1}^{I_m} a^2_{i_1,i_2,\ldots,i_m}}.$$

The norm of a tensor is a generalization of the *Frobenius norm* for matrices and of the l_2 *norm* for vectors.

Tensor product: the tensor product $\mathcal{A} \otimes \mathcal{B}$ of tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{B} \in \mathbb{R}^{I'_1 \times I'_2 \times \cdots \times I'_M}$ is another tensor, where each element is

$$(\mathcal{A} \otimes \mathcal{B})_{i_1,i_2,\ldots,i_N,i'_1,i'_2,\ldots,i'_M} = a_{i_1,i_2,\cdots,i_N} b_{i'_1,i'_2,\cdots,i'_M}$$

A mth-order tensor is a rank-one tensor if it can be defined as the tensor product of m vectors: $\mathcal{A} = \mathbf{a}^{(1)} \otimes \mathbf{a}^{(2)} \otimes \cdots \otimes \mathbf{a}^{(m)}$. For rank-one tensors $\mathcal{A} = \mathbf{a}^{(1)} \otimes \mathbf{a}^{(2)} \otimes \cdots \otimes \mathbf{a}^{(m)}$ and $\mathcal{B} = \mathbf{b}^{(1)} \otimes \mathbf{b}^{(2)} \otimes \cdots \otimes \mathbf{b}^{(m)}$, we have

$$\langle \mathcal{A}, \mathcal{B} \rangle = \left\langle \mathbf{a}^{(1)}, \mathbf{b}^{(1)} \right\rangle \left\langle \mathbf{a}^{(2)}, \mathbf{b}^{(2)} \right\rangle \cdots \left\langle \mathbf{a}^{(m)}, \mathbf{b}^{(m)} \right\rangle.$$

CP factorization: given a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_m}$ and an integer R, if it can be expressed as

$$\mathcal{A} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \otimes \mathbf{a}_r^{(2)} \otimes \cdots \otimes \mathbf{a}_r^{(m)} = \sum_{r=1}^{R} \prod_{i=1}^{m} \otimes \mathbf{a}_r^{(i)}$$

We call it a CP factorization of $\mathcal{A}$.

Major Challenges: Different from conventional data in vector space, the major research challenges of mining tensor data are as follows:

High dimension: In neuroimaging tensor data, each sample is usually represented as a high-dimensional multi-mode (also known as multi-way) array. One straightforward solution to tensor data mining is to reshape the tensors into vectors. However, the number of features will be extremely

high. For example, a typical MRI image of size $256 \times 256 \times 256$ voxels contains $16,777,216$ features [49]. This makes traditional data mining methods prone to critical issues, such as overfitting, especially with a small or moderate-sized dataset.
Tensor structure: Different from conventional vector data, tensor data can preserve the structural information of the high-dimensional space, such as the spatial relationships among different voxels in a 3-D image. These structural information can be very important in neuroimaging applications. For example, in MRI data, the values of adjacent voxels are usually correlated with each other. When converting tensors into vectors, such structural information will be lost.

2.2 Supervised Tensor Learning

Suppose we have a training set of n pairs of samples $\{(\mathcal{X}_i, y_i)\}_{i=1}^n$ for binary tensor classification problem. $\mathcal{X}_i \in \mathbb{R}^{I_1 \times \cdots \times I_m}$ is the input of the neuroimaging sample, which is represented as a tensor of m-mode. $y_i \in \{-1, +1\}$ is the corresponding class labels of $\mathcal{X}_i$. For example, if the i-th subject has Alzheimer's disease, the subject is associated with a positive label, *i.e.*, $y_i = +1$. Otherwise, if the subject is in the control group, *i.e.* the normal people, the subject is associated with a negative label, *i.e.*, $y_i = -1$.
Brain image classification tasks can be defined as a supervised tensor learning problem [18]. The optimization problem of supervised tensor learning is

$$\begin{aligned} &\min_{\mathcal{W},b,\xi} \frac{1}{2} \|\mathcal{W}\|_F^2 + C \sum_{i=1}^n \xi_i \\ &\text{s.t. } y_i \left(\langle \mathcal{W}, \mathcal{X}_i \rangle + b\right) \geq 1 - \xi_i, \\ &\qquad \xi_i \geq 0, \forall i = 1, \cdots, n. \end{aligned}$$

where $\mathcal{W}$ is the weight tensor of the separating hyperplane. $b \in \mathbb{R}$ is the bias. C is the trade-off between the classification margin and misclassification error. The above optimization problem is a generalization of the standard SVM from vector data to tensor data.
Nonlinear separability: In many neuroimaging applications, the data is usually not linearly separable in the input space. Conventional supervised tensor learning methods which can preserve tensor structures are often based upon linear models. Thus these methods cannot efficiently solve nonlinear learning problems on tensor data.
In the work [18], He et. al. studied the problem of supervised tensor learning with nonlinear kernels which can preserve the structure of the tensor data.
Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_m}$, and a mapping function into a Hilbert space

$$\phi : \mathcal{X} \to \phi(\mathcal{X}) \in \mathbb{R}^{H_1 \times H_2 \times \cdots \times H_P}.$$

The optimization problem becomes

$$\begin{aligned} &\max_{\alpha_1,\alpha_2,\cdots,\alpha_n} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle \phi(\mathcal{X}_i), \phi(\mathcal{X}_j) \rangle \\ &\text{s.t.} \sum_{i=1}^n \alpha_i y_i = 0, \\ &\qquad 0 \leq \alpha_i \leq C, \forall i = 1, \cdots, n, \end{aligned}$$

where α_i are the Lagrangian multipliers and $\langle \phi(\mathcal{X}_i), \phi(\mathcal{X}_j) \rangle$ are the inner product between the mapped tensors of $\mathcal{X}_i$ and $\mathcal{X}_j$ in the Hilbert space. Based on a suitable tensor kernel function $\kappa(\mathcal{X}_i, \mathcal{X}_j)$, the decision function can be written as

$$f(\mathcal{X}) = sign\left(\sum_{i=1}^n \alpha_i y_i \kappa(\mathcal{X}_i, \mathcal{X}) + b\right).$$

Dual Tensor Kernel (DuSK): In order to preserve the tensor structure in both original space and feature space, we can use CP factorizations to define the tensor kernels. Similar to other kernel functions in vector space, the feature space of tensor data is a high-dimensional tensor space. We can factorize tensor data directly in both the feature space and the original space, which is equivalent to performing the following mapping:

$$\phi : \sum_{r=1}^R \prod_{i=1}^m \otimes \mathbf{x}_r^{(n)} \to \sum_{r=1}^R \prod_{i=1}^m \otimes \phi\left(\mathbf{x}_r^{(i)}\right).$$

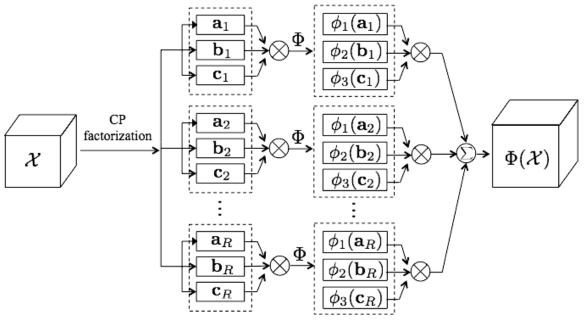


Figure 1: Dual-tensorial mapping [18]

The function can be considered as the operation of mapping the original data into tensor feature space and then conducting the CP factorization in the feature space. It is called the dual-tensorial mapping function (see Figure 1).
Suppose we have the CP factorization of $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_m}$ as $\mathcal{X} = \sum_{r=1}^R \prod_{i=1}^m \otimes \mathbf{x}_r^{(i)}$ and $\mathcal{Y} = \sum_{r=1}^R \prod_{i=1}^m \otimes \mathbf{y}_r^{(i)}$ respectively. Then the CP factorization of the data can be mapped into the tensor product feature space,

$$\begin{aligned} &\kappa\left(\sum_{r=1}^R \prod_{i=1}^m \otimes \mathbf{x}_r^{(i)}, \sum_{r=1}^R \prod_{i=1}^m \otimes \mathbf{y}_r^{(i)}\right) \\ =&\sum_{i=1}^R \sum_{j=1}^R \prod_{k=1}^m \kappa\left(\mathbf{x}_i^{(k)}, \mathbf{y}_j^{(k)}\right) \end{aligned}$$

The DuSK is an extension of the kernels in the vector space to tensor space. Thus DuSK kernel can be applied to any kernel-based learning methods to solve supervised tensor learning problems.

3. BRAIN NETWORK EXTRACTION

Brain networks are very different from conventional networks, such as social networks or Web, where the nodes and edges of the networks are predefined, e.g., the users/friendship or the webpages/hyperlinks. In brain networks, either the nodes or the edges are defined beforehand, but derived from neuroimaging data.
For the nodes of brain networks, it is essential to parcellate cerebral cortex into a set of disjoint regions and to use these brain regions as the nodes. Depending on the scale-levels, the specific brain regions represented by nodes can

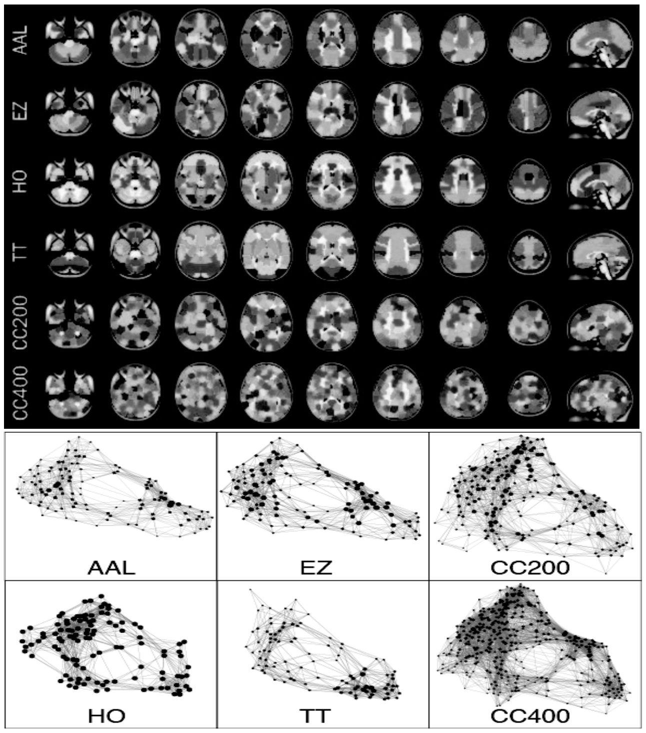


Figure 2: Different brain parcellation methods and their functional networks [13]. (a) AAL (automated anatomical labeling) [41]. (b) Harvard Oxford (HO) derived from anatomical landmarks (sulci and gyral) [15]. (c) EZ (Eickhoff-Zilles) [16]. (d) TT (Talariach Daemon) [26]. (e) CC200 and CC400 are derived from functional parcellations [12]. The functional networks were derived from all pairwise correlations between ROIs.

range from the small patches of cortex contained in individual MRI voxels to larger brain areas (e.g., dorsolateral prefrontal cortex). The structures of brain networks depend greatly on how the nodes are defined. Different parcellation methods will result in different network structures. For example, in Figure 2, we show six existing methods for brain parcellation, where the brain regions are partitioned differently according to different criteria. We can see that the structures of brain networks are also quite different when different parcellation methods are used.

For the edges of brain networks, it is essential to estimate different relationships among the brain regions [7; 35]. Examples include functional connectivity [44; 14], structural connectivity, effective connectivity [21], *etc.* Different types of connectivity will result in totally different networks, and can capture different types of relationships among brain regions.

3.1 Defining Nodes

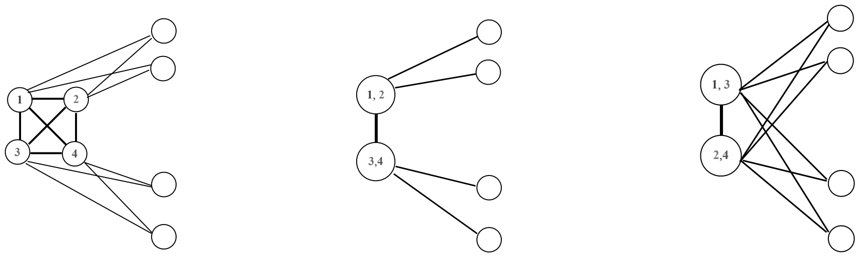
Early work in brain parcellation focused on anatomical atlases. Although much has been learned from these anatomical atlases, no functional or structural connectivity information was used to construct them. Thus an anatomically parcellated region (e.g., the anterior cingulate cortex) can contain subregions that are each characterized by dramatically different functional and structural connectivity patterns. These can significantly limit the utility of the constructed networks. The reasons are as follows: when we construct a brain network based upon a brain parcellation, we need to integrate (e.g., average) the data (e.g., functional activities) within individual regions in order to reduce the noises. We also need to integrate the connections between each pair of regions in order to reduce the uncertainty of connections. Ideally, each of the brain regions should contain subregions of homogeneous connectivity patterns, in order to preserve the utility of the network. For example, in Figure 3(a), we show the connectivity patterns of four subregions, represented as nodes ①-④. We can see that nodes ① and ② share similar connectivity patterns, which are very different from those of nodes ③ and ④. If we merge subregions with similar connectivity patterns into a larger region, as in Figure 3(b), the connectivity patterns are well preserved, because the network constructed can accurately represent the connectivity patterns of the subregions. However, if we merge subregions of different connectivity patterns into a larger region, as in Figure 3(c), the connectivity patterns can be distorted, because different patterns are integrated in the process of network construction.

Defining the nodes for brain networks, which corresponds to the community detection problem of network data, is a challenging task. The reasons are as follows:

- One key challenge comes from the noise and uncertainty of the data in the subregions. Ideally, all the subregions of a cortical region should share similar neurobiological properties, such that by aggregating these subregions into a large region, the noise and uncertainty can be reduced and the neurobiological properties of the subregions can be preserved in the large region. On the other hand, merging subregions with different properties can substantially reduce the utility of the brain network. There are trade-offs between reducing noise and preserving utility in brain parcellation.

- The second key challenge of defining nodes lies in the multiple domains of neurobiological properties, corresponding to multi-domain connectivity. These domains include spatial contiguity, functional connectivity, structural connectivity, etc. Different domains of connectivity correspond to different similarity measures among the subregions and will result in different nodes being defined during brain parcellation. There is lack of agreement on which is the best domain to define the cortical regions, because each domain can provide a unique view of the data. The domain of spatial-contiguity can improve the interpretability of the parcellated regions. The domain of functional-connectivity can help identify subregions with similar functions and connectivity patterns. The domain of structural-connectivity can help identify subregions with similar structural-connectivity patterns.

In this direction, Huang et. al. [22] studied the problem of identifying brain regions that are related to Alzheimer's disease from multi-modality neuroimaging data. Specially, MRI and PET are used to jointly identify disease related regions. MRI images can capture the structure information of the brain, while PET can capture the functional information of the brain. Both types are images can compensate each other. A sparse composite linear discriminant analysis



(a) connectivity of subregions (b) connectivity-based parcellation (c) alternative parcellation

Figure 3: An example of the network parcellation based upon connectivity. The nodes ①-④ represent four subregions of the cortex. In subfigures (b) and (c), the subregions are grouped into larger regions based upon different strategies.

model was proposed to identify brain regions from multiple types of imaging data.

Several works exist for parcellating the brain into a set of brain regions for connectivity analysis. Early efforts on brain parcellation use anatomical atlases [38] through postmortem architectonic measurements, e.g., cell morphology. Such atlases may contain subregions of heterogeneous functional or structural connectivity patterns. For resting-state functional connectivity analyses, different criteria have been used for evaluating the quality of a set of regions. (1) Functionally homogeneity: the regions should be functionally homogeneous [39]. The regionss voxels should have similar time courses [48] or produce similar functional connectivity patterns [10]. (2) Spatial contiguity: the regions should be spatially contiguous to preserve the interpretability of the parcellated regions [4; 33]. Spatial contiguity can also help identifying anatomically homogeneous regions, and hence preserve the interpretability of the connectivity results [39].

3.2 Estimating Edges

There are idiosyncratic differences among different types of edges that can be extracted from neuroimaging data.

Graph Representation: Functional brain networks are typically undirected and weighted [34]. The edge weights can be positive or negative [32]. Effective connections are directed, from source region to target region. Structural networks can be unweighted (binary tractography [1]) or weighted (in probabilistic tractography [3]) and are strictly nonnegative.

Interpretation: Structural networks can be thought of as the physical pathways along which the information flows. But functional connections cannot be interpreted in the same way, but can be thought of as the pair of regions that need to work together in order to perform a certain function. While effective connections corresponds to the causal relationships between the activities of different brain regions.

Functional connections: For functional connections, many of the research efforts focus on using sparse learning methods to derive a sparse network from functional neuroimaging data [37; 19]. In the work [37], the nodes of the brain network are given, which correspond to a set of brain regions. Then the functional activities within each region are aggregated by averaging the signals. In this way, the functional activity within the brain regions can be modeled as follows: Suppose we have n samples which are drawn from a multivariate Gaussian distribution independently, $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. The n samples correspond to the n time frames in functional imaging data, such as fMRI or PET. These n samples are assumed to be independent from each other, though this assumption may not always hold in neuroimaging data with high temporal resolutions. But for fMRI, this assumption usually hold pretty well in practice. Assume $\boldsymbol{\mu} \in \mathbb{R}^p$, where we have p different brain regions. $\Sigma \in \mathbb{R}^{p \times p}$ is the covariance matrix to be estimated. In order to induce sparsity in the inverse covariance matrix $\Theta = \Sigma^{\succ 1}$, l_1 norm regularization was used in the estimation process.

$$\max_{\Theta \quad 0} \log \det \Theta \quad \mathrm{tr}(S\Theta) \quad \lambda \|\mathrm{vec}(\Theta)\|_1$$

where S is the empirical covariance matrix, and λ is a regularization parameter. This was solved using a block coordinate descent method.

Effective connections: For effective connections, many of the research works focus on using structure learning method for Bayesian Networks to derive a directed network from functional neuroimaging data. In the work [20], effective connections among brain regions are modeled as a Bayesian Network (BN). The nodes of the BN corresponds to the brain regions, while the directed arcs between two nodes corresponds to the effective connections. The brain network extraction problem in this scenario becomes the structure learning problem for BN. Similar to functional connections, l_1 norm was also used to induce the sparsity within the network.

4. BRAIN NETWORK ANALYSIS

Once the brain networks are constructed from the neuroimaging data, the next step is to analyze the networks. The challenges are that conventional network measures are optimally suited for binary networks and are less well suited for weighted and signed networks. This often necessitates the conversion of weighted and signed networks to binary and unsigned networks. Such conversions are made by limiting the scope of studies to only positively weighted edges and defining a weight threshold to convert weighted networks into binary networks. These binarizing and simplifying manipulations are associated with great loss of information.

(1) The threshold is often arbitrarily made.

(2) Positively and negatively weighted edges are quite different in functions and closely related to each other in brain networks.

An ideal method for brain network analysis should be able to overcome the above methodological problems by generalizing the network edges to positive and negative weighted cases.

Conventional pattern mining research on brain networks can be divided into two schemes.

(1) The first scheme is usually called a bag of edges, where the graphs are treated as a collection/bag of edges. Statistical analysis is performed on each edge at a time, or a bag of independent edges through multivariate regression/classification methods. In these analyses, the connectivity structures of the networks are blinded.

(2) The second scheme is usually call graph invariants developed from graph theory. Topological measures, such as centrality and modularity, are used to capture global patterns in connectivity. But these approaches are less sensitive to local changes in connection (e.g., changes in only a few edges) as the first scheme.

In brain network analysis, the ideal patterns we want to mine from the data should combine the two schemes together. On the one hand, the pattern should be able to model the network connectivity patterns around the nodes, just like graph invariants methods. On the other hand, the pattern should be able to capture the changes in local areas, just like bag-of-edges methods. *Subgraph patterns* are more suitable for brain networks, which satisfy both of the above requirements.

4.1 Subgraph Pattern Mining on Uncertain Graphs

To determine whether a brain network functions normally or not, we can view the brain network derived from fMRI/PET data or DTI data as a graph and apply graph classification techniques which have been used in various applications, including drug discovery, i.e., predicting the effectiveness of chemical compounds on diseases [25]. Each graph object corresponds to the brain network of a subject in the study, which is associated with a label based upon certain properties of the subject. For example, if a subject has Alzheimer's disease, the graph object corresponding to the subject can be associated with a positive label. Otherwise, if the subject is in the control group, *i.e.* the normal people, the graph object is associated with a negative label.

Mining discriminative subgraph patterns for graph objects has attracted much attention in data mining community due to its important role in selecting features for graph classifications, generating graph indices, etc. [46; 23; 9; 25; 40]. Much of the past research in discriminative subgraph feature mining has focused on certain graphs, where the structure of the graph objects are certain, and the binary edges represent the "presence" of linkages between the nodes. However, in brain network data, there is inherent uncertainty about the graph linkage structure. Such uncertainty information will be lost if we directly transform uncertain graphs into certain graphs.

Specially, in the work [24], the brain networks are modeled as uncertain graphs, where the edges are assigned with an probability of existence. Suppose we are given an uncertain graph dataset $\widetilde{\mathcal{D}} = \{\widetilde{G}_1, \cdots, \widetilde{G}_n\}$ that consists of n uncertain graphs. $\mathbf{y} = [y_1, \cdots, y_n]^\top$ corresponds to their class labels, where $y_i \in \{+1, -1\}$ is the class label of $\widetilde{G}_i$.

Definition 1 (Certain Graph). *A certain graph is an undirected and deterministic graph represented as $G = (V, E)$. $V = \{v_1, \cdots, v_{n_v}\}$ is the set of vertices. $E \subseteq V \times V$ is the set of deterministic edges.*

Definition 2 (Uncertain Graph). *An uncertain graph is an undirected and nondeterministic graph represented as $\widetilde{G} = (V, E, p)$. $V = \{v_1, \cdots, v_{n_v}\}$ is the set of vertices. $E \subseteq V \times V$ is the set of nondeterministic edges. $p : E \to (0, 1]$ is a function that assigns a probability of existence to each edge in E. $p(e)$ denotes the existence probability of edge $e \in E$.*

Consider an uncertain graph $\widetilde{G}(V, E, p) \in \widetilde{\mathcal{D}}$, where each edge $e \in E$ is associated with a probability $p(e)$ of being present. As in other works [51; 50], it is assumed that the uncertainty variables of different edges in an uncertain graph are independent from each other. All uncertain graphs in a dataset $\widetilde{\mathcal{D}}$ share a given set of nodes V, which corresponds to a parcellation of the brain regions.

Each possible outcome of an uncertain graph $\widetilde{G}$ corresponds to an implied certain graph G. Here G is *implied* from uncertain graph $\widetilde{G}$ (denoted as $\widetilde{G} \Rightarrow G$), iff all edges in $E(G)$ are sampled from $E(\widetilde{G})$ according to their probabilities of existence in $p(e)$ and $E(G) \subseteq E(\widetilde{G})$. We have

$$\Pr\Big[\widetilde{G} \Rightarrow G\Big] = \prod_{e \in E(G)} \Pr_{\widetilde{G}}(e) \prod_{e \in E(\widetilde{G}) - E(G)} \big(\, 1 - \Pr_{\widetilde{G}}(e) \,\big)$$

The possible instantiations of an uncertain graph dataset $\widetilde{\mathcal{D}} = \{\widetilde{G}_1, \cdots, \widetilde{G}_n\}$ are referred to as *worlds* of $\widetilde{\mathcal{D}}$, where each world corresponds to an implied certain graph dataset $\mathcal{D} = \{G_1, \cdots, G_n\}$. A certain graph dataset $\mathcal{D}$ is called as being *implied* from uncertain graph dataset $\widetilde{\mathcal{D}}$ (denoted as $\widetilde{\mathcal{D}} \Rightarrow \mathcal{D}$), iff $|\mathcal{D}| = |\widetilde{\mathcal{D}}|$ and $\forall i \in \{1, \cdots, |\mathcal{D}|\}$, $\widetilde{G}_i \Rightarrow G_i$. There are $\prod_{i=1}^{|\widetilde{\mathcal{D}}|} 2^{|E(\widetilde{G}_i)|}$ possible worlds for uncertain graph dataset $\widetilde{\mathcal{D}}$, denoted as $\mathcal{W}(\widetilde{\mathcal{D}}) = \{\mathcal{D} \mid \widetilde{\mathcal{D}} \Rightarrow \mathcal{D}\}$. An uncertain graph dataset $\widetilde{\mathcal{D}}$ corresponds to a probability distribution over $\mathcal{W}(\widetilde{\mathcal{D}})$. The probability of each certain graph dataset $\mathcal{D} \in \mathcal{W}(\widetilde{\mathcal{D}})$ being implied by $\widetilde{\mathcal{D}}$ is $\Pr(\widetilde{\mathcal{D}} \Rightarrow \mathcal{D})$. By assuming that different uncertain graphs are independent from each other, we have

$$\Pr\Big[\widetilde{\mathcal{D}} \Rightarrow \mathcal{D}\Big] = \prod_{i=1}^{|\widetilde{\mathcal{D}}|} \Pr[\widetilde{G}_i \Rightarrow G_i]$$

The concept of *subgraph* is then defined based upon certain graphs.

Definition 3 (Subgraph). *Let $g = (V', E')$ and $G = (V, E)$ be two certain graphs. g is a subgraph of G (denoted as $g \subseteq G$) iff $V' \subseteq V$ and $E' \subseteq E$. We use $g \subseteq G$ to denote that graph g is a subgraph of G. We also say that G contains subgraph g.*

For an uncertain graph $\widetilde{G}$, the probability of $\widetilde{G}$ containing a subgraph feature g is defined as follows:

$$\begin{aligned}\Pr(g \subseteq \widetilde{G}) &= \sum_{G \in \mathcal{W}(\widetilde{G})} \Pr(\widetilde{G} \Rightarrow G) \cdot I(g \subseteq G) \\ &= \begin{cases} \prod_{e \in E(g)} p(e) & \text{if } E(g) \subseteq E(\widetilde{G}) \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$

which corresponds to the probability that a certain graph G implied by $\widetilde{G}$ contains subgraph g.

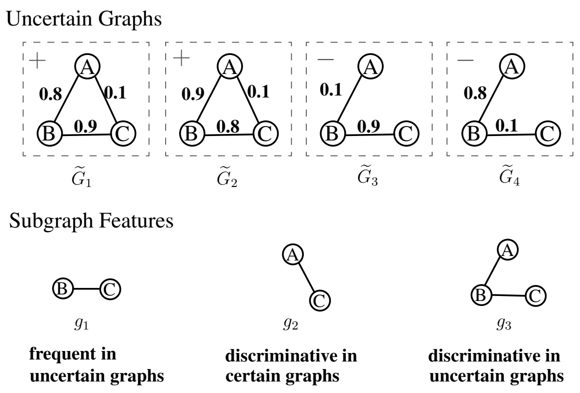


Figure 4: A toy example of uncertain brain networks, and different types of subgraph features [24].

The key issues of discriminative subgraph mining for uncertain graphs can be described as follows: The evaluation of discrimination scores for subgraph features in uncertain graphs is different from conventional subgraph mining problems. For example, in Figure 4, we show an uncertain graph dataset containing 4 uncertain graphs $\widetilde{G}_1, \cdots, \widetilde{G}_4$ with their class labels, $+$ or $-$. Subgraph g_1 is a frequent pattern among the uncertain graphs, but it may not relate to the class labels of the graphs. Subgraph g_2 is a discriminative subgraph features when we ignore the edge uncertainties. However, if such uncertainties are considered, we will find that g_2 can rarely be observed within the uncertain graph dataset, and thus will not be useful in graph classification. Accordingly, g_3 is the best subgraph feature for uncertain graph classification.

The work in [24] proposed a method based on dynamic programming to compute the probability distribution of the discrimination scores for each subgraph feature within an uncertain graph database. Then each of these probability distributions is aggregated to form a certain score based upon different statistical measures, including expectation, median, mode and φ-probability, to select discriminative subgraphs.

5. CONCLUSION

This paper provides an overview of the emerging area of brain network analysis, which has seen increasing attention in data mining communities in the recently years. Many research works on mining brain network data in the literature are not recognized as such in a formal way. This paper provides an understanding of how these works related to different data mining problems and methods. We provided different ways to categorize the data mining problems involved, such as subgraph pattern mining, supervised tensor learning and network extraction. We discussed the issue of mining brain regions that are relevant to certain diseases and the connectivities among these regions.

While brain networks are very challenging for data mining analysis, the problems are not unsurmountable. Many recent research efforts have been devoted to this area, which result in significant improvements in various dimensions. Data mining on brain networks seems to be an emerging area, which can be a fruitful research direction.

6. REFERENCES

[1] P. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi. In vivo fiber tractography using dt-mri data. *Magnetic Resonance in Medicine*, 44:625–632, 2000.

[2] P. Basser and C. Pierpaoli. Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor mri. *J of Magnetic Resonance, Series B*, 111(3):209–219, 1996.

[3] T. Behrens, H. Berg, S. Jbabdi, M. Rushworth, and M. Woolrich. Probabilistic diffusion tractography with multiple fiber orientations: What can we gain? *NeuroImage*, 34(1):144–155, 2006.

[4] P. Bellec, V. Perlbarg, S. Jbabdi, M. Pelegrini-Issac, J. Anton, J. Doyon, and H. Benali. Identification of large-scale networks in the brain using fmri. *Neuroimage*, 29:1231–1243, 2006.

[5] D. L. Bihan, E. Breton, D. Lallemand, P. Grenier, E. Cabanis, and M. Laval-Jeantet. Brain, mind, and the evolution of connectivity. *Radiology*, 161(2):401–407, 1986.

[6] B. Biswal, F. Yetkin, V. Haughton, and J. Hyde. Functional connectivity in the motor cortex of resting human brain using echo-planar mri. *Magnetic Resonance in Medicine*, 34(4):537–541, 1995.

[7] E. Bullmore and O. Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.

[8] T. Chenevert, J. Brunberg, and J. Pipe. Anisotropic diffusion in human white matter: demonstration with mr techniques in vivo. *Radiology*, 177(2):401–405, 1990.

[9] H. Cheng, D. Lo, Y. Zhou, X. Wang, and X. Yan. Identifying bug signatures using discrimative graph mining. In *ISSTA*, pages 141–152, 2009.

[10] A. Cohen, D. Fair, N. Dosenbach, F. Miezin, D. Dierker, D. V. Essen, B. Schlaggar, and S. Petersen. Defining functional areas in individual human brains using resting functional connectivity mri. *Neuroimage*, 41:45–57, 2008.

[11] R. Craddock, G. James, P. Holtzheimer, X. Hu, and H. Mayberg. A whole brain fmri atlas generated via spatially constrained spectral clustering. *Human Brain Mapping*, 2012.

[12] R. Craddock, G. James, P. Holtzheimer, X. Hu, and H. Mayberg. A whole brain fmri atlas generated via spatially constrained spectral clustering. *Human Brain Mapping*, 2013.

[13] R. Craddock, S. Jbabdi, C. Yan, J. Vogelstein, F. Castellanos, A. Martino, C. Kelly, K. Heberlein, S. Colcombe, and M. Milham. Imaging human connectomes at the macroscale. *Nature Methods*, 10:524–539, 2013.

[14] I. Davidson, S. Gilpin, O. Carmichael, and P. Walker. Network discovery via constrained tensor analysis of fmri data. In *KDD*, pages 194–202, 2013.

[15] R. Desikan, F. Segonne, B. Fischl, B. Quinn, B. Dickerson, D. Blacker, R. Buckner, A. Dale, R. Maguire, B. Hyman, M. Albert, and R. Killiany. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *NeuroImage*, 31:968–980, 2006.

[16] S. Eickhoff, K. Stephan, H. Mohlberg, C. Grefkes, G. Fink, K. Amunts, and K. Zilles. A new spm toolbox for combining probabilistic cytoarchitectonic maps and functional imaging data. *NeuroImage*, 25:1325–1335, 2005.

[17] M. Fox and M. Raichle. Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nature Reviews Neuroscience*, 8(9):700–711, 2007.

[18] L. He, X. Kong, P. Yu, A. Ragin, and Z. Hao. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In *SDM*, 2014.

[19] S. Huang, J. Li, L. Sun, J. Ye, K. Chen, and T. Wu. Learning brain connectivity of azheimer's disease from neuroimaging data. In *NIPS*, pages 808–816, 2009.

[20] S. Huang, J. Li, J. Ye, A. Fleisher, K. Chen, T. Wu, and E. Reiman. Brain effective connectivity modeling for alzheimer's disease by sparse gaussian bayesian network. In *KDD*, pages 931–939, 2011.

[21] S. Huang, J. Li, J. Ye, A. Fleisher, K. Chen, T. Wu, and E. Reiman. Brain effective connectivity modeling for alzheimer's disease study by sparse bayesian network. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(6):1328–1342, 2013.

[22] S. Huang, J. Li, J. Ye, T. Wu, K. Chen, A. Fleisher, and E. Reiman. Identifying alzheimers disease-related brain regions from multi-modality neuroimaging data using sparse composite linear discrimination analysis. In *NIPS*, pages 1431–1439, 2011.

[23] N. Jin, C. Young, and W. Wang. GAIA: graph classification using evolutionary computation. In *SIGMOD*, pages 879–890, 2010.

[24] X. Kong, P. Yu, X. Wang, and A. Ragin. Discriminative feature selection for uncertain graph classification. In *SDM*, 2013.

[25] X. Kong and P. S. Yu. Semi-supervised feature selection for graph classification. In *KDD*, pages 793–802, 2010.

[26] J. Lancaster, M. Woldorff, L. Parsons, M. Liotti, C. Freitas, L. Rainey, P. Kochunov, D. Nickerson, S. Mikiten, and P. Fox. Automated talairach atlas labels for functional brain mapping. *Human Brain Mapping*, 10(3):120–131, 2000.

[27] M. McKeown, S. Makeig, G. Brown, T. Jung, S. Kindermann, A. Bell, and T. Sejnowski. Analysis of fmri data by blind separation into independent spatial components. *Human Brain Mapping*, 6:160–188, 1998.

[28] M. Moseley, Y. Cohen, J. Kucharczyk, J. Mintorovitch, H. Asgari, M. Wendland, J. Tsuruda, and D. Norman. Diffusion-weighted mr imaging of anisotropic water diffusion in cat central nervous system. *Radiology*, 176(2):439–445, 1990.

[29] J. Nolte. *The human brain: an introduction to its functional anatomy.* Mosby-Elsevier, 2009.

[30] S. Ogawa, T. Lee, A. Kay, and D. Tank. Brain magnetic resonance imaging with contrast dependent on blood oxygenation. *Proc. of the National Academy of Sciences*, 87(24):9868–9872, 1990.

[31] S. Ogawa, T. Lee, A. Nayak, and P. Glynn. Oxygenation-sensitive contrast in magnetic resonance image of rodent brain at high magnetic fields. *Magnetic Resonance in Medicine*, 14(1):68–78, 1990.

[32] M. Rubinov and O. Sporns. Weight-conserving characterization of complex functional brain networks. *NeuroImage*, 56(4):2068–2079, 2011.

[33] S. Smith, P. Fox, K. Miller, D. Glahn, P. Fox, C. Mackay, N. Filippini, K. Watkins, R. Toro, A. Laird, and C. Beckmann. Correspondence of the brains functional architecture during activation and rest. *Proc. of the National Academy of Sciences*, 106:13040–13045, 2009.

[34] S. Smith, D. Vidaurre, C. Beckmann, M. Glasser, M. Jenkinson, K. Miller, T. Nichols, E. Robinson, G. Salimi-Khorshidi, M. Woolrich, D. Barch, K. Ugurbil, and D. V. Essen. Functional connectomics from resting-state fmri. *Trends in Cognitive Sciences*, 17(12):666–682, 2013.

[35] O. Sporns. *Networks of the Brain.* MIT Press, 2010.

[36] O. Sporns, G. Tononi, and R. Kotter. The human connectome: A structural description of the human brain. *PLoS Computational Biology*, 2005.

[37] L. Sun, R. Patel, J. Liu, K. Chen, T. Wu, J. Li, E. Reiman, and J. Ye. Mining brain region connectivity for alzheimer's disease study via sparse inverse covariance estimation. In *KDD*, pages 1335–1344, 2009.

[38] J. Talairach and P. Tournoux. *Co-planar stereotaxic atlas of the human brain: 3-Dimensional proportional system-An approach to cerebral imaging.* Thieme, 1988.

[39] B. Thirion, G. Flandin, P. Pinel, A. Roche, P. Ciuciu, and J. Poline. Dealing with the shortcomings of spatial normalization: Multi-subject parcellation of fmri datasets. *Human Brain Mapping*, 27:678–693, 2006.

[40] M. Thoma, H. Cheng, A. Gretton, J. Han, H. Kriegel, A. Smola, L. Song, P. Yu, X. Yan, and K. Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *SDM*, pages 1075–1086, 2009.

[41] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *NeuroImage*, 15:273–289, 2002.

[42] X. Wang, P. Foryt, R. Ochs, J. Chung, Y. Wu, T. Parrish, and A. Ragin. Brain, mind, and the evolution of connectivity. *Brain and Cognition*, 42(1), 2000.

[43] X. Wang, P. Foryt, R. Ochs, J. Chung, Y. Wu, T. Parrish, and A. Ragin. Abnormalities in resting-state functional connectivity in early human immunodeficiency virus infection. *Brain Connectivity*, 1(3):207–217, 2011.

[44] X. Wang, P. Foryt, R. Ochs, J. Chung, Y. Wu, T. Parrish, and A. Ragin. Abnormalities in resting-state functional connectivity in early human immunodeficiency virus infection. *Brain Connectivity*, 1(3):207, 2011.

[45] S. Xiang, L. Yuan, W. Fan, Y. Wang, P. Thompson, and J. Ye. Multi-source learning with block-wise missing data for alzheimers disease prediction. In *KDD*, pages 85–193, 2013.

[46] X. Yan, H. Cheng, J. Han, and P. Yu. Mining significant graph patterns by leap search. In *SIGMOD*, pages 433–444, 2008.

[47] J. Ye, K. Chen, T. Wu, J. Li, Z. Zhao, R. Patel, M. Bae, R. Janardan, H. Liu, G. Alexander, and E. Reiman. Heterogeneous data fusion for alzheimer's disease study. In *KDD*, pages 1025–1033, 2008.

[48] Y. Zang, T. Jiang, Y. Lu, Y. He, and L. Tian. Regional homogeneity approach to fmri data analysis. *NeuroImage*, 22:394–400, 2004.

[49] H. Zhou, L. . Li, and H. Zhu. Tensor regression with applications in neuroimaging data analysis. *American Statistical Association*, 2012.

[50] Z. Zou, H. Gao, and J. Li. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *KDD*, pages 633–642, 2010.

[51] Z. Zou, J. Li, H. Gao, and S. Zhang. Frequent subgraph pattern mining on uncertain graph data. In *CIKM*, pages 583–592, 2009.

Predictive Analysis of Engine Health for Decision Support

Shubhabrata Mukherjee[1], Aparna S. Varde[2], Giti Javidi[3], Ehsan Sheybani[4]
1. Department of Medicine, University of Washington, Seattle, WA
2. Department of Computer Science, Montclair State University, Montclair, NJ
3. Department of Math and Computer Science, Virginia State University, Petersburg, VA
4. Department of Engineering and Technology, Virginia State University, Petersburg, VA
smukherj@u.washington.edu | vardea@montclair.edu | gjavidi@vsu.edu | esheyban@vsu.edu

ABSTRACT

Data mining, the discovery of knowledge from data, bridges several disciplines such as database management, artificial intelligence, statistics, visualization and the domain of the data, e.g., biology or engineering. Knowledge discovered by mining the data can be used for various purposes such as developing decision support systems and intelligent tutors. In this paper we present such a data mining problem in the mechanical engineering domain where knowledge discovery from the data is performed using statistical approaches, to conduct predictive analysis for decision support. More specifically, we focus on the *engine health problem* which consists of using existing data on the behavior of an engine in order to predict whether the engine is capable of functioning well (i.e., it is healthy) and to offer suggestions on preventive maintenance. The data we use for this predictive analysis consists of graphs that plot process parameters such as the vibration and temperature of the engine with respect to time. In this paper we define the problem in detail, propose a solution based on statistical inference techniques, summarize our experimental evaluation and discuss the applications of this work in various fields from a decision support angle.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-data mining, scientific databases

General Terms

Design, Experimentation, Human Factors

Keywords

Decision-making, Estimation, Statistical Techniques

1. INTRODUCTION

1.1 Background

In various scientific domains data on process parameters obtained from laboratory experiments is often used to discover knowledge for assisting decision-making in corresponding real industrial processes. In this paper, we deal with such data from the domain of Mechanical Engineering pertaining to the behavior of an engine. The term **engine** refers to a machine for converting any of various forms of energy into mechanical force and motion that can be used within various devices such as cars, trains and airplanes [1]. When an engine is in use, there are process parameters associated with it that control its behavior such as vibration, noise, carbon dioxide level and temperature. These are briefly explained below.

- *Vibration:* The mechanical oscillation of an engine during its operation is referred to as vibration. This parameter serves as a fundamental measure of the functioning of an engine [1].
- *Noise:* This refers to the internal combustion engine noise produced by a rapid rate of pressure rise which excites resonance in the gas inside a combustion chamber cavity [24].
- *Carbon dioxide level:* Carbon dioxide (CO_2) is a desirable byproduct that is produced when the carbon from the fuel is fully oxidized during the combustion process [21].
- *Temperature:* This is a measure of the thermal environment during the combustion of the engine. It is the approximate temperature of the combustible gases prior to spark ignition [18].

The behavior of the engine is typically depicted by plotting these process parameters with respect to time. The result of plotting each such parameter is a time-series graph, i.e., the x-axis represents time and the y-axis represents one of the parameters. An example of such a graph is shown in Figure 1. Here, the vibration of the engine is observed over a given time interval. This enables scientists to visualize engine behavior with respect to how frequently it vibrates. Likewise, graphs of other process parameters provide visualization of the respective phenomena pertaining to the engine.

1.2 Problem Definition

The engine data on process parameters serves an excellent source of knowledge that can be discovered to predict whether the engine is good enough to be used in the real world or whether it needs to be repaired to avoid damage. Accordingly, the term **engine health** characterizes the reliability or goodness of an engine for use in various applications. The process of repairing an engine for avoiding future damage is known as **preventive maintenance**.

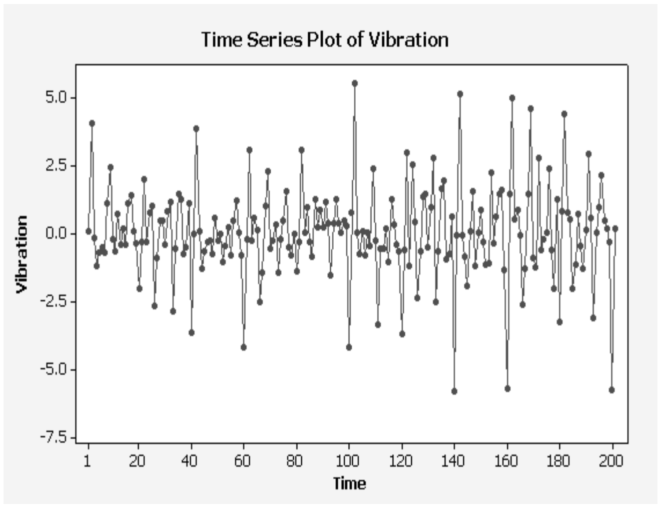


Figure 1: Time Series Graph from Engine Data

Scientists are interested in knowledge discovery from engine data in order to answer questions posed by users such as:

1. Is a given engine healthy?
2. What can be stated about future engine health?
3. Does an engine depict anomalous behavior?
4. What could be the causes of potential faults?
5. What preventive maintenance is needed?

This is precisely the problem addressed in this paper. Given the data on process parameters for a particular engine, in the form of time series graphs, the goal is to predict whether the engine is healthy, answer related questions and make recommendations accordingly. It is called the **engine health problem** and falls within the general realm of **predictive analysis**, i.e., the process of predicting or estimating tendencies in advance based on analyzing existing data. This can assist in making decisions about the engine use in corresponding applications.

1.3 Solution Approach and Tasks

We propose an approach to solve the engine health problem deploying classical techniques in the area of statistical inference that are found to be useful in estimation-related problems. More specifically, we consider concepts such as tolerance limits and survival probability, which will be elaborated in the next section. Furthermore, we develop suitable heuristics as needed for the analysis based on domain knowledge, statistical concepts and visual inspection. The significant tasks in this work include:

- *Analyzing individual parameter impact:* The different process parameters may correlate. However, in order to identify the causes of faults, it is important to extract each parameter independently of the others, which is a significant issue since the data may be mixed.
- *Determining thresholds for engine health:* It is not known apriori what constitutes a healthy engine. Hence, it is imperative to define such thresholds taking into account various factors which constitutes another important task.
- *Identifying the possibility of false alarms:* There can be situations where the behavior of an engine seems abnormal from its process parameter graphs but that is due to an external factor, e.g., a car going through a bump in a road. Thus, the engine could still be healthy. Such situations, namely, false alarms pose an additional issue to deal with.

1.4 Evaluation and Applications

We evaluate our proposed solution using data from the domain of Mechanical Engineering. The evaluation presented in this paper is conducted with data obtained from diesel engines in particular. We consider snapshots of the time-series data consisting of process parameter graphs after extracting individual parameters. The results of our analysis are summarized in this paper and are corroborated by domain experts.
The applications of this work include automobile manufacturing, locomotive design, aircraft monitoring and advance warning systems. In all these applications it is useful to answer questions pertaining to engine health and preventive maintenance in order to assist in making crucial decisions about engine use. The predictive analysis plays an important role in decision support in such applications as elaborated later in this paper.

1.5 Layout of Paper

The rest of this paper is organized as follows. Section 2 describes our proposed solution approach for the engine health problem. Section 3 summarizes the experimental evaluation. Section 4 discusses the targeted applications of this work considering a decision support perspective. Section 5 outlines the related work in the area. Section 6 gives the conclusions.

2. PROPOSED APPROACH

The solution we propose for the engine health problem is built heuristically based on techniques in the area of **statistical inference**. The aim of statistical inference is to use the information contained in the sample data to increase our knowledge about the sampled population. Inferential statistics can be subdivided into two major regions, namely, estimation (point and interval) and hypothesis testing. In this paper we have deployed some estimation techniques such as tolerance and prediction limits for predictive analysis of engine health.
We describe our predictive analysis approach with reference to the data on the vibration parameter. This is considered to be the most important of all the parameters that affect engine health as gathered from domain knowledge. Similar ideas for predictive analysis can be applied to the other parameters. We use the statistical inference techniques of tolerance limits, prediction limits and survival probability to conduct the predictive analysis as elaborated below.

2.1 Tolerance Limits

The concept of tolerance limits in statistical inference can be explained as follows. Tolerance intervals cover a fixed proportion of the population with a stated confidence. Let

β denote the *content* and γ denote the *confidence level* of a tolerance interval. Then a (β, γ) tolerance interval in constructed in such a way that the interval is to contain a proportion of at least β of the population with $100\gamma\%$ confidence. More details on this especially in the context of the normal distribution can be found in [9].

$$UTL = \bar{X} + kS_x, \quad \text{with} \quad k = \frac{1}{\sqrt{n}} t_{n-1,\gamma}(z_\beta \sqrt{n}), \qquad (1)$$

is a (β, γ) upper tolerance limit based on $\bar{X}$ and S_x^2, where

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \quad \text{and} \quad S_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

and where z_β is the βth quantile of the standard normal distribution, and $t_{m,\alpha}(\delta)$ denotes the αth quantile of a noncentral t distribution with df $= m$ and noncentrality parameter δ. The quantity k is referred to as a tolerance factor. Similarly, a lower tolerance limit (LTL) can be obtained by replacing the $+$ sign by the $-$ sign. Note that a (β, γ) upper tolerance limit also provides a $100\gamma\%$ upper confidence limit for the βth quantile and in a similar manner a (β, γ) lower tolerance limit also provides a $100\gamma\%$ lower confidence limit for the $(1 - \beta)$th quantile.

An exact two-sided tolerance interval for the normal distribution is given by $\bar{X} \pm kS_x$, where k is the tolerance factor. Odeh (1978) computed the exact tolerance factor k for $n = 2(1)98, 100$, $\beta = .75, .90, .95, .975, .99, .995, .999$ and $\gamma = .5, .75, .90, .95, .975, .99, .995$. A FORTRAN program to compute the values of k is given in [2].

2.2 Prediction Limits

Prediction limits serve to make estimations about the future based on current data. We define upper and lower prediction limits as follows. The $1 - \alpha$ Upper Prediction Limit (UPL) for a future measurement is given by

$$UPL = \bar{X} + t_{n-1,1-\alpha} S_x \sqrt{1 + \frac{1}{n}}, \qquad (2)$$

where $t_{m,c}$ denotes the cth quantile of Student's t distribution [19] with df $= m$.

The Lower Prediction Limit (LPL) can be obtained by replacing the $+$ sign in the above equation with the $-$ sign. Thus, a $1-\alpha$ lower prediction limit for a future measurement is given by

$$LPL = \bar{X} - t_{n-1,1-\alpha} S_x \sqrt{1 + \frac{1}{n}}, \qquad (3)$$

where $t_{m,c}$ denotes the cth quantile of Student's t distribution with df $= m$ [19].

2.3 Survival Probability

Survival probability helps to determine how reliable a particular measure is. Hence it is also referred to as reliability. It is explained below [12]. Suppose we want to estimate the survival probability at time t based on a sample $X_1, ..., X_n$ from a normal distribution. The survival probability is given by $S = P(X > t)$. The lower confidence limit for S can be found out using the concept of one-sided tolerance limits mentioned above. For example, if a (β, γ) lower tolerance limit for the Normal(μ, σ) distribution is greater than t, then we can conclude that S is at least β with confidence γ. Therefore, an approximate one-sided $100\gamma\%$ lower confidence limit for S is given by

$$\max\left\{p : \bar{X} - \frac{1}{\sqrt{n}} t_{n-1,\gamma}(z_\beta \sqrt{n}) S_x > t\right\}. \qquad (4)$$

So, a lower limit for survival probability denoted as LSP can be obtained as the solution (with respect to β) of the equation

$$LSP = t_{n-1,\gamma}(z_\beta \sqrt{n}) = \frac{\bar{X} - t}{S_x/\sqrt{n}}. \qquad (5)$$

2.4 Heuristics for Prediction

The concepts of tolerance interval, prediction interval and survival probability applied to the time-series graphs for engine data serve as the basis for heuristically predicting the engine health and making recommendations for preventive maintenance accordingly. (Note that a heuristic by definition is a rule-of-thumb likely to lead to the right answer but not essentially proven theoretically).

The tolerance interval calculates the range which should contain a certain proportion of each individual measurement in the population. There are two different proportions associated with the tolerance interval, namely, a coverage percentage and a degree of confidence. Based on these, we propose the following heuristic.

Tolerance Limit Heuristic: *If the* $(0.99, 0.95)$ *tolerance interval for the vibration data from a given engine falls within the* $(-3.5, 3.5)$ *range, then that given engine is likely to be healthy.*

This means that, if we measure 100 data points for vibration from the time-series graphs 100 times, 95 of those times we expect to catch at least 99 of those data points inside the limits $(-3.5, 3.5)$. This should give us an idea about what vibration values to expect for a healthy engine. If we notice that during a certain time the vibration values are exceeding these two limits, we can check the engine for faults.

Unlike the tolerance interval, the prediction interval provides estimates about the *future* values based upon past or present background samples. For example, for a healthy engine, the prediction limits enable us to estimate the future vibration values with a certain degree of confidence without actually carrying out the experiment. We propose the prediction limit heuristic here as follows.

Prediction Limit Heuristic: *If the 95% lower prediction limit for a sample signal from a given engine exceeds* 2.5, *then it can be used to estimate future engine health.*

Furthermore, reliability or survival probability gives us an idea about the *consistency* of one's measures. In our problem of engine maintenance, we can estimate the chance of occurrence of an anomalous vibration data point with respect to a sample of vibration data obtained from a healthy engine. We thus propose the survival probability heuristic as follows.

Survival Probability Heuristic: *If the lower limit for the survival probability of a given sample of engine data matches that of a known healthy engine at least 90%, with confidence 95%, then the given sample is not anomalous.*

These heuristics can be used in conjunction with each other for predicting the health of current and future engines based on the analysis of existing data. All the heuristics have been determined by using visual inspection, domain knowledge and statistical concepts. Although not proven in theory, they are found to yield good results in practice, as evident from our evaluation. It is to be noted that these heuristics may need to be adapted with modifications for other domains based on the nature of the data. However, the fundamental logic used in the predictive analysis, i.e., using statistical inference guided by heuristics, is still applicable.

2.5 Significant Tasks in the Approach

2.5.1 *Analyzing individual parameter impact*

There are different parameters involved in engine health, namely, vibration, noise, carbon dioxide level and temperature. Correlations are likely to exist between these parameters which poses a challenge in individual analysis. However, in order to conduct analysis to determine the impact of each individual parameter on engine health, we assert that it is desirable to isolate each parameter. This is done in order to single out the causes of specific problems in relation to a particular parameter.

Hence, the individual parameter impact is analyzed as follows. Note that the engine is built so as to allow each piece to be tested for its signature. When the engine is assembled each piece is therefore equipped with a sensor. By collecting the relevant signature, we extract the individual graph of a given parameter with respect to time. Using this information, we execute the process of extracting the individual parameter data rather than directly using the output from the engine which could contain a mixture of parameters. This extracted data on individual process parameters forms the basis of our predictive analysis.

2.5.2 *Determining thresholds for engine health*

There is no specific formula available in the domain that allows us to make a distinction between a healthy and a faulty engine. Hence, one of the issues we deal with is to determine suitable threshold levels for engine health. Three main criteria are involved here as follows:

- *Statistical Techniques:* The concepts from statistical inference, i.e., tolerance limits, prediction limits and survival probability provide the theoretical foundation to define the thresholds.
- *Domain Knowledge:* The fundamental knowledge of the domain, i.e., the explanations of the terms healthy and faulty and the semantics of the process parameters provided the practical aspects needed to determine appropriate thresholds.
- *Visual Inspection:* The graphs of process parameters served as good visual tools to enable analysis and comparison with the human eye hence forming the empirical basis to obtain the required thresholds.

Hence, by using all the three criteria above, engine health thresholds have been proposed in the form of heuristics to guide the predictive analysis as outlined in the previous subsection. These have been used experimentally and have been found to yield good results (as presented in Section 3).

2.5.3 *Identifying the possibility of false alarms*

A false alarm is a situation where a system may give a warning of being faulty even though it is not. An example with respect to our work is the case of an automobile running over an irregular road. The engine in such a case may be normal. However, the corresponding time-series graphs may still show an abnormal trend due to external factors. We identify the possibility of such false alarms occurring in our system.

While conducting the predictive analysis we consider the effect of such situations by taking into account how often the engine data crosses the thresholds in the concerned graphs. If for example, this happens only once in a particular graph, this is attributed to be a potential false alarm, i.e., the engine could still be considered healthy. However, if abnormal behavior occurs repeatedly, it is reported that the engine is not healthy. This reasoning is based primarily on observation.

This sets the stage for proposing a solution to avoid false alarms in the future. Upon discussions with domain experts, we suggest the following method to help alleviate the false alarm problem. We recommend that an analyzer be mounted within the engine itself such that when it senses abnormal behavior it gives a direct warning. This would reduce the chances of signaling abnormal behavior due to external factors thereby helping to minimize the effect of false alarms.

2.6 Algorithms and Questions

Based on the discussion so far, we present our fundamental algorithms. Algorithm 1 is for engine health estimation and Algorithm 2 is for anomaly detection. These serve to conduct the predictive analysis setting the stage for decision support.

Given these algorithms, we address some of the anticipated questions that users could pose with respect to decision support. These have been listed earlier and can be answered as follows.

1. **Is a given engine healthy?** The answer to this question can be conveyed to users by deploying tolerance limits over the given engine data and using the tolerance limit heuristic to identify a healthy engine (See Algorithm 1).
2. **What can be stated about future engine health?** Tolerance limits deployed in conjunction with prediction limits and their respective heuristics can guide the answer to this question because the former help to distinguish a healthy engine while the latter help to estimate future behavior based on the analysis of existing data (See Algorithm 1).
3. **Does an engine depict anomalous behavior?** The users can be given the answer to this question based on survival probability. The survival probability of the engine can be calculated using the given data and applying the corresponding heuristic to estimate whether anomalous behavior is likely to occur (See Algorithm 1).
4. **What could be the causes of potential faults?** Extraction of individual parameter signatures coupled with analysis using statistical inference and heuristics can lead to this answer (See Algorithms 1 and 2). Comparative analysis of healthy and faulty engines with respect to concerned parameters can also

Algorithm 1 Engine Health Estimation

```
Require: Given engine signal data $X$ (For each parameter: Vibration, Noise, $CO_2$ level, Temperature)
Require: $\alpha$: for $(1-\alpha)$ prediction, $\beta$: content, $\gamma$: confidence
  $OUTPUT1$: Estimation for given engine
  $OUTPUT2$: Estimation for future engines
  $H_g$: Given engine healthy (0/1)
  $H_f$: Future engine healthy (0/1)
  $UTL$: Upper tolerance limit
  $LTL$: Lower tolerance limit
  $UPL$: Upper prediction limit
  $LPL$: Lower prediction limit
  while $(\beta \geq 0.99) \wedge (\gamma \geq 0.95)$ do
    Calculate $UTL$, $LTL$ for $X$ (Equation 1)
    if $(UTL < 3.5) \wedge (LTL > -3.5)$ then
      $H_g \leftarrow 1$
    else
      $H_g \leftarrow 0$
    end if
  end while
  while $\alpha \leq 5$ do
    Calculate $LPL$ for $X$ (Equation 3)
    if $LPL > 2.5$ then
      $H_f \leftarrow H_g$
    end if
  end while
  if $H_g = 1$ then
    $OUTPUT1 \leftarrow$ Given engine is healthy
  else
    $OUTPUT1 \leftarrow$ Given engine needs preventive maintenance
  end if
  if $H_f = 1$ then
    $OUTPUT2 \leftarrow$ Future such engines would be healthy
  else
    $OUTPUT2 \leftarrow$ Future such engines would need preventive maintenance
  end if
  PRINT $OUTPUT1$, $OUTPUT2$
```

Algorithm 2 Anomaly Detection

```
Require: Given engine signal data $X$ (For each parameter)
Require: Healthy engine signal data $H$ (From Algorithm 1)
Require: $\beta$: content, $\gamma$: confidence
  $OUTPUT$: Anomaly detection for engine
  $LSP$: Lower limit for survival probability
  $A_g$: Anomaly in given engine (0/1)
  while $(\beta \geq .90) \wedge (\gamma \geq .95)$ do
    Calculate $LSP(X)$ and $LSP(H)$ for $X$ and $H$ (Equation 5)
    if $LSP(X) = LSP(H)$ then
      $A_g \leftarrow 0$
    else
      $A_g \leftarrow 1$
    end if
  end while
  if $A_g = 0$ then
    $OUTPUT \leftarrow$ No anomaly
  else
    $OUTPUT \leftarrow$ Anomaly occurs
  end if
  PRINT OUTPUT
```

be helpful here. For example, if the vibration data of a given engine indicates that it is healthy after applying the concepts and heuristics above but the data on carbon dioxide level of that engine is outside the acceptable tolerance limits as per the heuristic, then it can be indicated to users that: *Potential faults could occur due to low CO_2 reading.*

5. **What preventive maintenance is needed?** The prediction heuristics deployed (see Algorithm 1) along with the knowledge of the domain serve to answer this question. For example, if the engine is detected as not being healthy, and the cause of potential faults is attributed to the carbon dioxide level, a preventive maintenance suggestion offered to users could be: *Increase oxidation of fuel during combustion to give higher CO_2 reading.*

While answering these and other questions for prospective users, it is important to note that there is a range between engine health and engine failure. It is analogous to the difference between a healthy person and someone on the verge of death. A healthy engine is highly suitable for use in corresponding applications while a failed engine is not at all usable. The range in between is for engines that can be used but are not very healthy and thus could cause problems. Our predictive analysis approach targets the safer end of this range, geared towards keeping the engine as healthy as possible. Hence, the approach is useful for decision support in targeted applications where it is helpful to make decisions in advance considering various possible scenarios.

3. EXPERIMENTAL EVALUATION

We conducted experiments on our predictive model using engine data from the domain of Mechanical Engineering. This data consisted of graphs that plot vibration versus time. Thus, the impact of the vibration parameter on engine health was evaluated. The data used for evaluation, i.e.,

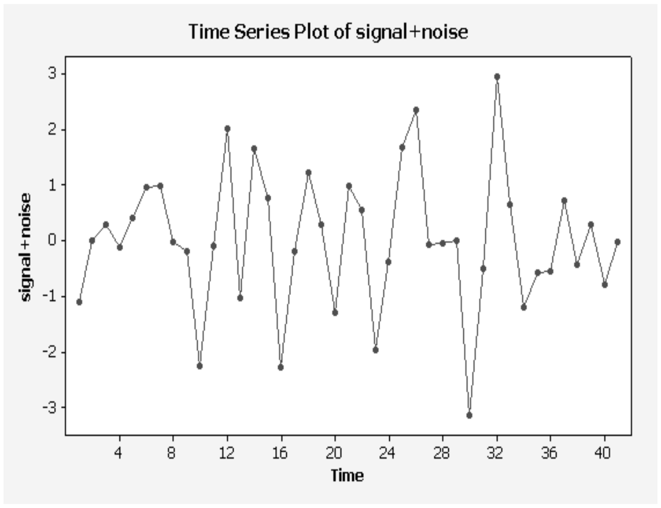


Figure 2: Graph of Vibration (Signal with Gaussian noise) versus Time

test data was distinct from the data used for building the model, i.e., training data. A summary of our experimental evaluation is presented below.

3.1 Data Description

Figure 2 shows an example of the vibration graph used in our experimental evaluation. The term signal in this graph refers to the vibration signal which is time-series data. The vibration data that was used in the experiments contained Gaussian noise [1]. (This is however different from the *Noise* parameter in the engine data which refers to the combustion of the engine). We define this experimental noise data as X_i, $i = 1, ..., 41$. A quantile-quantile plot showed an excellent fit of these data to a normal distribution. The mean is given by $\bar{X} = 0.00964$ and the standard deviation by $S_x = 1.24181$.

3.2 Observations from Experiments

We obtain the following plots for the vibration data shown in Figure 2. The summary plot depicted in Figure 3 summarizes the distribution of the data indicating where most of the data is concentrated. The normal probability plot that appears in Figure 4 checks how closely the data fits normal distribution.

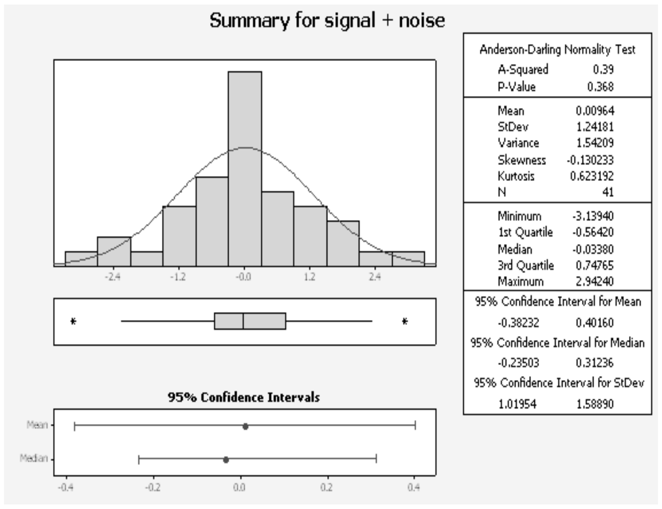
Tolerance Limits: In Table 1, we present 95% one-sided upper tolerance limits and the two-sided tolerance interval along with the corresponding tolerance factors.

Table 1. Tolerance Limits for Vibration Data

(β, γ)	Factor for one-sided	Lower limit	Upper limit	Factor for two-sided	Two-sided tolerance interval
(.9,.95)	1.69106	-2.0903	2.1096	2.04852	(-2.5342, 2.5535)
(.95,.95)	2.11831	-2.6209	2.6402	2.44036	(-3.0208, 3.0401)
(.99,.95)	2.93160	-3.6308	3.6501	3.20548	(-3.9709, 3.9902)

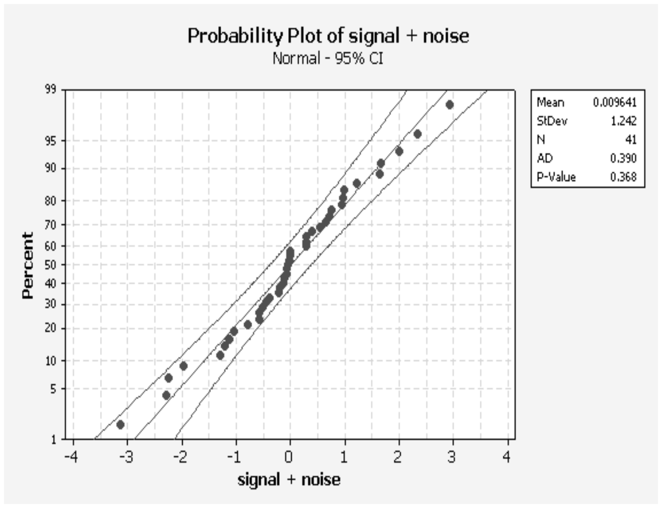
Prediction Limits: Using the formula (3), we computed the 90% prediction limit as 1.647 and 95% prediction limit as 2.126.

[1]In communications, a random interference generated by the movement of electricity in the line is called Gaussian noise. It is similar to white noise, but confined to a narrower range of frequencies. You can actually see and hear Gaussian noise you tune your TV to a channel that is not operating.



Anderson-Darling Normality Test	
A-Squared	0.39
P-Value	0.368
Mean	0.00964
StDev	1.24181
Variance	1.54209
Skewness	-0.130233
Kurtosis	0.623192
N	41
Minimum	-3.13940
1st Quartile	-0.56420
Median	-0.03380
3rd Quartile	0.74765
Maximum	2.94240
95% Confidence Interval for Mean	
-0.38232	0.40160
95% Confidence Interval for Median	
-0.23503	0.31236
95% Confidence Interval for StDev	
1.01954	1.58890

Figure 3: Summary Plot for Vibration Data



Mean	0.009641
StDev	1.242
N	41
AD	0.390
P-Value	0.368

Figure 4: Normal Probability Plot for Vibration Data

Probability of Exceeding a Threshold Value: Suppose we want to find a 95% lower limit for the probability that a sample signal exceeds 2.5, that is, $P(X > 2.5)$. Using (5), we get

$$t_{40,0.95}(z_\beta\sqrt{41}) = \frac{0.00964 - 2.5}{1.24181/\sqrt{41}} = -12.841.$$

Solving for the noncentrality parameter, we get $z_\beta\sqrt{41} = -15.6637$. This implies that $z_\beta = -2.4463$ or $\beta = .0072$. Thus, the probability that the signal exceeds 2.5 in a sample is at least 0.0072 with confidence 95%.

3.3 Interpretation of Observations

The $(0.95, 0.95)$ two-sided tolerance interval for the engine vibration data is given by $(-3.02, 3.04)$ which can be interpreted to mean that the engine is healthy using the tolerance limit heuristic (See Algorithm 1).

We can expect 95% of the values for engine vibration to fall under the lower (-3.02) and upper (3.04) limits with 95% confidence. Figure 5 provides the time-series plot of engine vibration data with the two bands representing the lower and upper limits of the $(0.95, 0.95)$ tolerance limits.

Initially, the vibrations are under the tolerance limits which signifies that the engine is healthy but after a certain amount of time an aberrant behavior is noticed since the vibrations are increasingly exceeding the tolerance limits. This should tell the experimenter that a change in the process has occurred which means that the engine is behaving abnormally. Hence, preventive measures should be taken, i.e., the engine should be fixed in order to prevent it from going to a faulty stage.

The experimental observations can be discussed in a real world context as follows. Suppose, we are interested in studying the anomalous behavior of the diesel engine. We let the engine run for a long time and collect the data for signal vibration against time. The time series plot of the vibration data along with the (.95, .95) two-sided tolerance limits is presented in Figure 5. The term signal + noise in this plot denotes the vibration signal with Gaussian noise while the two horizontal lines denote the tolerance limits.

From the figure, we can notice a change in the process given by the fact that vibrations are exceeding the limits on an average during the latter stages.

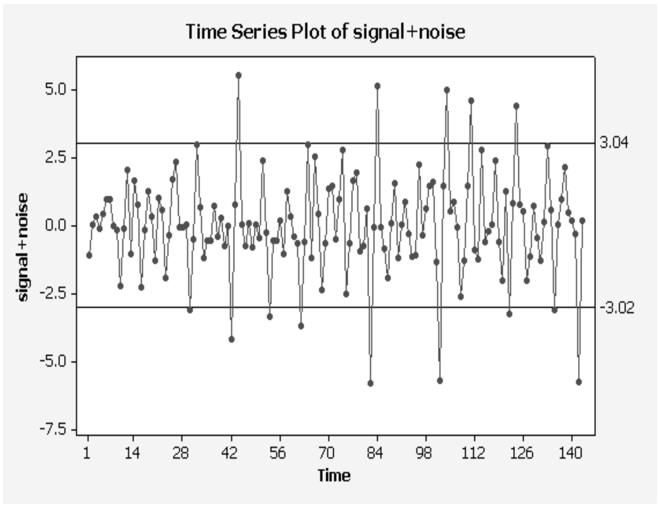


Figure 5: Vibration Data with Tolerance Limits (The two horizontal lines are the upper and lower limits of the two-sided tolerance interval)

4. DECISION SUPPORT APPLICATIONS

This work can be useful in decision support in the areas of manufacturing, engineering design and aerospace technology. For example, our predictive analysis approach can be used in automobile manufacturing, locomotive design, aircraft monitoring and advance warning systems. We discuss these targeted applications below.

4.1 Automobile Manufacturing

Automakers face the challenge of making appropriate decisions to create the best-in-class vehicles and maintain a corporate reputation for performance and value [4]. A technique often used in the making of cars is predictive process engineering that develops measurements and standards for accurately predicting the outcome of production processes so that parts are made right the first time. For example, organizations such as NIST (National Institute of Standards and Technology, USA) provide inputs to the automobile industry in order to enhance manufacturing processes using predictive process engineering.

Consider this in the context of engine vibration. The vibration of a moving part represents the balance of the part. This relates to wear and tear and hence to the life of the part. Interpreting the time-related changes in vibration amplitude and frequency provides information about the state of elements in a gas turbine [4]. Therefore, our approach which includes predictive analysis of vibration data from an engine can be useful to interpret such information and give suggestions to support decisions about the manufacturing of the automobile in order to yield a better product.

4.2 Locomotive Design

In the design of locomotives, crucial decisions need to be made about certain factors that can be controlled in order to improve its operation. One such factor is crashworthiness. A study on locomotive crashworthiness has been conducted as presented in [22]. This deals with a range of collision scenarios. It supports the efforts of the Locomotive Crashworthiness Working Group of the Federal Railroad Administration of USA. Crashworthiness itself can be related to other factors such as engine health.

Predictive analysis can be of help here. Computer automated modal testing of a rotating component can be employed. Also, since engines control systems and components, the vibrational environment of this equipment can be measured by a scanning vibrometer. This can be used to predict factors such as crashworthiness with suitable techniques. Our approach can be used here to assist in decision-making about the development of other models such as those for crashworthiness estimation during locomotive design.

4.3 Aircraft Monitoring

In recent years there have been developments in airborne and ground monitoring of aircraft gas turbine engines [5]. A device called the piezoelectric accelerometer along with enhanced electronic processing technology leads to the universal employment of an aircraft system that deploys vibration analysis. The storage of vibration data at several in-flight engine conditions aims to enable ground crew to balance the fan without a ground engine run.

Our approach for predictive analysis of engine health thus finds a potential application here. Flight simulators could be developed that predict in advance the capability of an aircraft in performing its required operation thereby providing decision support in the ground and airborne monitoring of the engines. This could for example be used to train pilots and ground crew in aircraft monitoring.

4.4 Advance Warning Systems

In the engines used in vehicles today, there are chips programmed to signal engine failures [10]. These chips typically raise an alarm when the engine is close to breakdown. Several cars, for example, would flag a low battery signal when the engine appears as though will cease to function soon. This happens in other vehicles too such as aircrafts and locomotives. As stated earlier, there is obviously a range between engine health and engine failure and our approach useful in decision support heads towards safer end of this range.

It could therefore be potentially useful to signal an alarm when the engine reaches a level such that it is no longer considered very healthy. This does not mean that it is on the verge of breakdown. However, it require preventive maintenance to avoid future problems. Thus our approach can potentially be used to build advance warning systems in vehicles to notify of possible engine failure well in advance rather than waiting for the stage when the engine is close to breakdown. Such systems would be complementary to existing warning systems in current vehicles [17].

5. RELATED WORK

Data mining techniques in conjunction with those in related fields allow a search for valuable information in large volumes of data. The explosive growth in databases has created a need to develop technologies that use information and knowledge intelligently. Thus, data mining approaches present increasingly important research areas in various fields. The last decade saw a very strong growth in this arena and novel methods have been developed in neural networks, algorithm architecture, dynamic prediction-based, analysis of systems architecture, intelligence agent systems, modeling, knowledge-based systems, system optimization and information systems, with their applications in research and practical domains [13].

Statistical techniques have often been used in data mining. In [6] an overview of the common issues between data mining and statistics is presented. It can be thought of automated exploratory data analysis of large complex data sets. These include decision trees, rule induction, nearest neighbors, feature extraction and visualization. These methods have a major impact on variety of fields such as biology, healthcare and business as big data analytics is on the forefront. Our work falls in this general category, focusing specifically on the use of statistical inference in data mining.

In [3] the topic of visual data mining is discussed with reference to current research areas of interest. Issues addressed are high-dimensional visualization, pixel-oriented techniques, clustering and classification, visual frameworks, combining visualization and data mining and domain-specific applications. We consider one more paradigm, namely, statistical techniques in visual data mining. This is particularly in the context of scientific analysis.

Computational statistics is used in areas such as exploratory graphics [23]. While presentation graphics usually summarize information to be displayed, exploratory graphics are used to look for results. For example, exploratory graphics may be drawn to support the data investigations of an analyst. They are typically fast and informative rather than slow and precise. The problem discussed in our paper is also of this nature. We explore engine data to conduct predictive analysis. Thus, our goals also involve providing estimated answers with efficiency.

In [15] they propose a time-series analysis approach which combines wavelet analysis with visualization tools such as SiZer and SiNos. Although wavelets can be used for comparison when the data being analyzed involves time series, this process is computationally intensive. Our proposed approach provides efficient analysis. Moreover, our goals are catered towards predictive analysis which goes a step beyond just visually comparing the data.

Microarray data analysis for gene expressions using statistical inference is presented in [14]. They consider variation of ratios between measured fluorescent intensities at different spots in a microarray. We can draw an analogy here since we use methods based on statistical inference in our analysis. However, the nature of our data is different. We analyze graphical plots of engine parameters dealing with issues such as parameter correlations, tolerance limits and false alarms. Hence, the specific inference-based methods need to be adapted accordingly.

Developments in IT and high performance computing make huge data sets available in scientific domains, e.g., a simulation can produce TBs of data in a few hours which humans could take weeks to analyze. Other examples are data from medical imaging, bioinformatics, and remote sensing. Thus, there is tremendous interest in scientific communities to explore emerging data mining approaches in conjunction with related domains. The work of [8] provides supercomputing professionals an insight into several scientific and engineering domains, including astrophysics, medical imaging, computational fluid dynamics, structural mechanics, and ecology. They present discovery-driven (instead of traditional hypothesis-driven) approaches to automatically extract patterns from data, especially useful in scientific domains for large data sets. Our work fits in this realm as we consider statistical techniques along with domain-driven approaches in scientific fields for knowledge discovery with respect to specific problems, in our case, predictive analysis of engine health.

Data mining approaches are suitably deployed in scientific domains for decision support problems. Pawlish et al. [16] address the issue of decision support in environmental management. They aim to make data centers more sustainable with greater efficiency, cost-effectiveness and greenness while maintaining productivity. They mine real data on scientific parameters, e.g., temperature, humidity, CPU utilization and server sprawl and analyze the results of their experiments to propose strategies for optimizing performance. For example, they propose migrating some data center operations to the cloud, by conducting analysis with case-based reasoning and decision trees incorporating various scenarios.

They head towards a hybrid model, considering privacy, security, availability etc. Some carefully selected operations are executed using cloud services while other major ones reside on host servers. Their proposed strategies are useful in decision support for data centers, helping to green the planet. We can draw a parallel here, with respect to decision support for engine health which helps in preventive maintenance. This is analogous to maintaining a green environment for a healthy planet. These scientific issues head towards prevention which is better than cure, where domain-driven data mining is are useful.

Prognostic health monitoring (PHM) technologies have found an ever expanding array of applications in the industrial environments. In particular, robust detection of mechanical damage in an engine has been demonstrated via processing of high-speed, one-dimensional accelerometer (vibration due to damaged parts) data. Such data collected with and without the damaged parts shows distinctive signatures that are quantitatively and qualitatively separable. Pattern Recognition techniques drive the signature generation and abnormality detection process through the use of data-driven techniques that estimate deviation of the engine from normal behavior [7].

The ability to measure these signatures in real-time has many applications in auto industry product and function optimization. Due to the immense effect of rough environment and corruptive parameters of engine, it is often impossible to collect and evaluate the parameters required for health monitoring. However, there are algorithms that are capable of performing this task analytically and quantitatively at a low computational cost (real-time) and high efficiency. An example of such algorithms [20] works based on a heuristic analysis of similar parameters to assess an image and quantify the quality of the image by characterizing important aspects of human visual quality.

Our work in this paper is orthogonal to such approaches. In the context of the problems described here, the issue of engine health is important. It caters to one end of the spectrum, of maintaining healthy behavior as opposed to waiting for the other end, where failure occurs. Data mining approaches overlapping with statistical methods and domain-specific analysis in engineering are very helpful here in drawing conclusions for decision support and related applications. These also present the potential for future work in these areas in conjunction with each other.

Future work obviously comes with further challenges. Han et al. have presented a list of challenges that provide the potential for further research in data mining for science and engineering domains in their work [11]. These include spatiotemporal and multimedia data mining; knowledge discovery from moving object data, RFID data and sensor network data; multidimensional online analytical mining oriented towards data cubes; mining from data streams; knowledge discovery from text, Web and other unstructured data; and analyzing information networks.

We agree and emphasize that these challenges can be viewed in the light of conducting research in data mining along with its related fields encompassing database management, artificial intelligence, statistics and visualization in addition to a detailed study of the respective science and engineering domains, with the involvement of domain experts. This multidisciplinary research would continue to present interesting and exciting opportunities to the data mining community.

6. CONCLUSIONS

In this paper, we deal with the problem of engine health. Given time-series graphs on process parameters depicting the behavior of an engine, we conduct predictive analysis to find out whether an engine is healthy and to make suggestions on preventive maintenance useful in decision support. We deploy concepts from statistical inference that we perceive as being pertinent to domain-driven data mining. These concepts are tolerance limits, prediction limits and survival probability and we them to heuristically estimate engine health. Upon running experiments with diesel engine data from the domain of Mechanical Engineering, it is found that our approach is feasible in the context of this domain and its targeted applications. This work is useful in decision support for automobile manufacturing, locomotive design, aircraft monitoring and advance warning systems. Future work involves the correlated analysis of multiple parameters together. We hope this will yield even more interesting results.

7. ACKNOWLEDGMENTS

This work started while all the authors were at Virginia State University. It has been partly supported by the sources of each author. We acknowledge the startup funds at Montclair State University for Aparna Varde and the respective support to the other authors. We particularly thank the NASA NSTI grant for the financial and NASA GSFC for the scientific support of this research.

8. REFERENCES

[1] Brotherton, T., Chadderdon, G. and Grabill, P. "Automated Rule Extraction for Engine Vibration Analysis". *Proceedings of the IEEE Aerospace Conference*, Aspen, March 1999, 1-12.

[2] Eberhardt, K., Mee, R. and Reeve, C. "Computing Factors for Exact Two-Sided Tolerance Limits for a Normal Distribution". *Communications in Statistics: Simulation and Computation*, 1989, 18, 397-413.

[3] Eick, S. and Keim, D. "Visual Data Mining: KDD Workshop Report". *ACM SIGKDD Explorations*, 2001, 3 (2), 70.

[4] Fine, C., St. Clair, R., Lafrance, J. and Hillebrand, D. *The U.S. Automobile Manufacturing Industry.* Technical Report, U.S. Department of Commerce, Office of Technology Policy, December 1996.

[5] Ford, T. "Engine Vibration Analysis". *Aircraft Engineering and Aerospace Technology*, 1997, 69(2), 126-128.

[6] Friedman, J. "Data Mining and Statistics: What is The Connection". *Proceedings of the 29th Symposium on the Interface: Computing Science and Statistics*, November 1997.

[7] Getman, A., Cooper, C.D., Key, G., Zhou, H., Frankle, N., "Detection of Mobile Machine Damage using Accelerometer Data and Prognostic Health Monitoring Techniques". *Proceedings of IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, Nashville, TN, March-April 2009, 101-104.

[8] Grossman, R., Kamath, C., Kumar, V. *Data Mining for Scientific and Engineering Applications*. Tutorial at SC2001, November 2001.

[9] Guttman, I. *Statistical Tolerance Regions: Classical and Bayesian*. Griffin Publishers, London, England, 1970.

[10] Katsumi, A. *Failure Warning System of Electric Power Unit in Vehicle*. United States Patent 5646599, Mitsubishi Denki Kabushiki Kaisha, July 1997.

[11] Han, J., Gao, J. "Research Challenges for Data Mining in Science and Engineering". Book Chapter in *Next Generation of Data Mining*, Chapman and Hall, 2009.

[12] Krishnamoorthy, K. Mathew, T., Mukherjee, S. "Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability". Technometrics, 2008, 50(1):69-78.

[13] Liao, S.H., Chu, P.H., Hsiao, P.Y., "Data mining techniques and applications . A Decade Review from 2000 to 2011. Expert Systems with Applications", 2012, 39(12): 11303-11311.

[14] Newton, M., Kendziorski, C., Richmond, C., Blattner. F., Tsui, K. "On Differential Variability of Expression Ratios: Improving Statistical Inference about Gene Expression Changes from Microarray Data". *Journal of Computational Biology*, 2001, 8(1), 37-52.

[15] Park, C., Godtliebsen, F., Taqqu, M., Stoev, S. and Marron J. "Visualization and Inference based on Wavelet Coefficients, SiZer and SiNos". *Computational Statistics and Data Analysis*, 2007, 51(12), 5994-6012.

[16] Pawlish, M., Varde A., Robila, S., "Decision Support in Data Centers for Sustainability". *IEEE ICDM-W*, Dallas, Texas, 2013, 613-620.

[17] Rahman, A. *Motor Vehicle Early Warning System*. United States Patent 6121896, Hodes Pessin and Katz, September 2000.

[18] Sandia Fluid Mechanics Group. *Engine Combustion Network*. Technical Report, Sandia National Group, California, USA, 2007.

[19] Student (Gosset, W. S.). "The Probable Error of a Mean". *Biometrika*, 1908, 6(1), 1-25.

[20] Sheybani, E., Garcia-Otero, S., Adnani, F., Javidi, G., Deshpande, M., "An Algorithm for Real-Time Blind Image Quality Comparison and Assessment"., IJECE, 2(1) February 2012, 120-129.

[21] Toyota Motor Sales. *Emission Analysis*. Technical Report Emission #2, Toyota Inc., USA.

[22] Tyrell, D., Severson, K., Marquis, B., Martinez, E., Mayville, R., Rancatore, R., Stringfellow, R., Hammond, R., Perlman, B. "Locomotive Crashworthiness Design Modifications Study". *Proceedings of IEEE / ASME Joint Railroad Conference*, April 1999.

[23] Unwin, A., Chen, C. and Hardle, W. *Handbook of Computational Statistics*, Volume III (Data Visualization). Springer-Verlag, Heidelberg, Germany, 2007.

[24] Zheng, G. and Leung, A. "Internal Combustion Engine Noise Analysis with Time-Frequency Distribution". *ASME Journal of Engineering for Gas Turbines and Power*, 2002, 124, 645-649.

About the Authors:

Dr. Shubhabrata Mukherjee is a Research Assistant Professor of Medicine at University of Washington. He obtained his Ph.D. and M.S. in Statistics from University of Louisiana, Lafayette, M.S. in Mathematics from IIT Kharagpur and B.S. in Mathematics from University of Calcutta. His papers have been published in reputed journals such as the Technometrics, PLoS One. Dr. Mukherjee has authored a chapter in an edited book "Biomathematics: Modelling and Simulation" published by World Scientific. His current research areas are psychometrics and statistical genetics.

Dr. Aparna Varde is an Associate Professor of Computer Science at Montclair State Univ, NJ, with her PhD and MS (Comp. Sci.) from WPI, Massachusetts, and BE (Comp. Eng.) from Univ of Bombay, India. She was an Assistant Professor (Comp. Sci.) at Virginia State Univ, Visiting Senior Researcher at Max Planck Institute for Informatics, Germany, and Software Engineer at Lucent and Citicorp. Dr. Varde has around 60 publications (IEEE, ACM, AAAI etc.), 2 software trademarks, research grants from NSF and PSEG, and is a research advisor / committee member for PhD, MS and BS students. She has been a panelist in NSF's IIS, reviewer for journals (TKDE, TKDD, DMKD, VLDBJ, DKE, AIEDAM etc.), co-chair of IEEE and ACM PhD Workshops, and PC Member at conferences, e.g., ICDM, EDBT, SDM, DEXA. Her teaching and research spans Data Mining, Databases and Artificial Intelligence.

Dr. Giti Javidi is an Associate Professor in Computer Science at Virginia State University. She obtained her Ph.D. in IT and M.S. in Computer Science from University of South Florida and a B.S. in Computer Science from University of Central Oklahoma. Dr. Javidi has software trademarks, journal articles and conference papers (IEEE, ACM, and ASEE). In addition to teaching, her main research interests are in the field of new information technologies applied to collaborative learning, virtual learning, information visualization, usability engineering and human-computer interaction. Dr. Javidi is the recipient of NSF/ITEST grant for a three years (2007-2010) Digispired project.

Dr. Ehsan Sheybani is a Full Professor in Computer Engineering at Virginia State University, VA. He obtained his Ph.D., MS, and BS in Electrical Engineering from University of South Florida, Florida State University, and University of Florida, respectively. Dr. Sheybani has journal articles and conference papers (IEEE, ACM, ASEE, SPIE). His professional activities include serving as a PI/Co-PI for NSF, NIH, NASA, DoD, and DEd grant proposals, reviewer for several IEEE transactions, track-chair of IEEE/ACM WTS 2006 and 2007 symposia, and reviewer for NIH and DoD. His teaching and research areas are Digital Signal Processing and Wireless Sensor Networks.

OpenML: networked science in machine learning

Joaquin Vanschoren[1], Jan N. van Rijn[2], Bernd Bischl[3], and Luis Torgo[4]
[1] Eindhoven University of Technology - j.vanschoren@tue.nl
[2] Leiden University - j.n.van.rijn@liacs.leidenuniv.nl
[3] Technische Universität Dortmund - bischl@statistik.tu-dortmund.de
[4] INESC Tec / University of Porto - ltorgo@dcc.fc.up.pt

ABSTRACT

Many sciences have made significant breakthroughs by adopting online tools that help organize, structure and mine information that is too detailed to be printed in journals. In this paper, we introduce OpenML, a place for machine learning researchers to share and organize data in fine detail, so that they can work more effectively, be more visible, and collaborate with others to tackle harder problems. We discuss how OpenML relates to other examples of networked science and what benefits it brings for machine learning research, individual scientists, as well as students and practitioners.

1. INTRODUCTION

When Galileo Galilei discovered the rings of Saturn, he did not write a scientific paper. Instead, he wrote his discovery down, jumbled the letters into an anagram, and sent it to his fellow astronomers. This was common practice among respected scientists of the age, including Leonardo, Huygens and Hooke.

The reason was not technological. The printing press was well in use those days and the first scientific journals already existed. Rather, there was little personal gain in letting your rivals know what you were doing. The anagrams ensured that the original discoverer alone could build on his ideas, at least until someone else made the same discovery and the solution to the anagram had to be published in order to claim priority.

This behavior changed gradually in the late 17th century. Members of the Royal Society realized that this secrecy was holding them back, and that if they all agreed to publish their findings openly, they would all do better [19]. Under the motto "take nobody's word for it", they established that scientists could only claim a discovery if they published it first, if they detailed their experimental methods so that results could be verified, and if they explicitly gave credit to all prior work they built upon.

Moreover, wealthy patrons and governments increasingly funded science as a profession, and required that findings be published in journals, thus maximally benefiting the public, as well as the public image of the patrons. This effectively created an economy based on *reputation* [19; 28]. By publishing their findings, scientists were seen as trustworthy by their peers and patrons, which in turn led to better collaboration, research funding, and scientific jobs. This new culture continues to this day and has created a body of shared knowledge that is the basis for much of human progress.

Today, however, the ubiquity of the internet is allowing new, more scalable forms of scientific collaboration. We can now share detailed observations and methods (data and code) far beyond what can be printed in journals, and interact in real time with many people at once, all over the world.

As a result, many sciences have turned to online tools to share, structure and analyse scientific data on a global scale. Such *networked science* is dramatically speeding up discovery because scientists are now capable to build directly on each other's observations and techniques, reuse them in unforeseen ways, mine all collected data for patterns, and scale up collaborations to tackle much harder problems. Whereas the journal system still serves as our collective long-term memory, the internet increasingly serves as our collective *short-term working memory* [29], collecting data and code far too extensive and detailed to be comprehended by a single person, but instead (re)used by many to drive much of modern science.

Many challenges remain, however. In the spirit of the journal system, these online tools must also ensure that shared data is trustworthy so that others can build on it, and that it is in individual scientists' best interest to share their data and ideas.

In this paper, we discuss how other sciences have succeeded in building successful networked science tools that led to important discoveries, and build on these examples to introduce OpenML, a collaboration platform through which scientists can automatically share, organize and discuss machine learning experiments, data, and algorithms.

First, we explore how to design networked science tools in Section 2. Next, we discuss why networked science would be particularly useful in machine learning in Section 3, and describe OpenML in Section 4. In Section 5, we describe how OpenML benefits individual scientists, students, and machine learning research as a whole, before discussing future work in Section 6. Section 7 concludes.

2. NETWORKED SCIENCE

Networked science tools are changing the way we make discoveries in several ways. They allow hundreds of scientists to discuss complex ideas online, they structure information from many scientists into a coherent whole, and allow anyone to reuse all collected data in new and unexpected ways. In this section we discuss how to design such online tools, and how several sciences have used them to make important breakthroughs.

2.1 Designing networked science

Nielsen [29] reviews many examples of networked science, and explains their successes by the fact that, through the interaction of many minds, there is a good chance that someone has just the right expertise to contribute at just the right time:

Designed serendipity Because many scientists have complementary expertise, any shared idea, question, observation, or tool may be noticed by someone who has just the right (micro)expertise to spark new ideas, answer questions, reinterpret observations, or reuse data and tools in unexpected new ways. By scaling up collaborations, such 'happy accidents' become ever more likely and frequent.

Dynamic division of labor Because each scientist is especially adept at certain research tasks, such as generating ideas, collecting data, mining data, or interpreting results, any seemingly hard task may be routine for someone with just the right skills, or the necessary time or resources to do so. This dramatically speeds up progress.

Designed serendipity and a dynamic division of labor occur naturally when ideas, questions, data, or tools are broadcast to a large group of people in a way that allows everyone in the collaboration to discover what interests them, and react to it easily and creatively. As such, for online collaborations to scale, online tools must make it practical for anybody to join and contribute any amount at any time. This can be expressed in the following 'design patterns' [29]:

- Encourage small contributions, allowing scientists to contribute in (quasi) real time. This allows many scientists to contribute, increasing the cognitive diversity and range of available expertise.
- Split up complex tasks into many small subtasks that can be attacked (nearly) independently. This allows many scientists to contribute individually and according to their expertise.
- Construct a rich and structured information commons, so that people can efficiently build on prior knowledge. It should be easy to find previous work, and easy to contribute new work to the existing body of knowledge.
- Human attention doesn't scale infinitely. Scientists only have a limited amount of attention to devote to the collaboration, and should thus be able to focus on their interests and filter out irrelevant contributions.
- Establish accepted methods for participants to interact and resolve disputes. This can be an 'honor code' that encourages respectable and respectful behavior, deters academic dishonesty, and protects the contributions of individual scientists.

Still, even if scientists have the right expertise or skill to contribute at the right time, they typically also need the right incentive to do so.

As discussed, scientists actually solved this problem centuries ago by establishing a reputation system implemented using the best medium for sharing information of the day, the journal. Today, the internet and networked science tools provide a much more powerful medium, but they also need to make sure that sharing data, code and ideas online is in scientists' best interest.

The key to do this seems to lie in extending the reputation system [29]. Online tools should allow everyone to see exactly who contributed what, and link valuable contributions to increased esteem amongst the users of the tools and the scientific community at large. The traditional approach to do this is to link useful online contributions to authorship in ensuing papers, or to link the reuse of shared data to citation of associated papers or DOI's.[1]

Moreover, beyond bibliographic measures, online tools can define new measures to demonstrate the scientific (and societal) impact of contributions. These are sometimes called *altmetrics* [35] or article-level metrics[2]. An interesting example is ArXiv[3], an online archive of preprints (unpublished manuscripts) with its own reference tracking system (SPIRES). In physics, preprints that are referenced many times have a high status among physicists. They are added to resumes and used to evaluate candidates for scientific jobs. This illustrates that what gets measured, gets rewarded, and what gets rewarded, gets done [29; 30]. If scholarly tools define useful new measures and track them accurately, scientists will use them to assess their peers.

2.2 Massively collaborative science

Online tools can scale up scientific collaborations to any number of participants. In mathematics, Fields medalist Tim Gowers proposed[4] to solve several problems that have eluded mathematicians for decades by uniting many minds in an online discussion. Each of these *Polymath projects* state a specific, unsolved math problem, is hosted on a blog[5] or wiki[6], and invites anybody who has anything to say about the problem to chip in by posting new ideas and partial progress.

Designed serendipity plays an important role here. Each idea, even if just a hunch, may spark daughter ideas with those who happen to have just the right background. Indeed, several polymaths "found themselves having thoughts they would not have had without some chance remark of another contributor".[7] There is also a clear dynamic division of labor, with many mathematicians throwing out ideas, criticizing them, synthesizing, coordinating, and reformulating the problem to different subfields of mathematics.

Blogs and wikis are ideally suited as tools, because they are designed to scale up conversations. They ensure that each contribution is clearly visible, stored and indexed, so that anybody can always see exactly what and how much you contributed.[8] Moreover, everyone can make quick, small contributions by posting comments, all ideas are organized into threads or pages, and new threads or pages can be opened to focus on subproblems. In addition, anybody can quickly scan or search the whole discussion for topics of interest.

[1]Digital Object Identifiers can be cited in papers. See, for instance, DataCite (http://www.datacite.org).

[2]http://article-level-metrics.plos.org/alm-info/

[3]http://arxiv.org

[4]http://gowers.wordpress.com/2009/01/27/is-massively-collaborative-mathematics-possible

[5]http://polymathprojects.org

[6]http://michaelnielsen.org/polymath1

[7]http://gowers.wordpress.com/2009/03/10/polymath1-and-open-collaborative-mathematics/

[8]Similarly, open source software development tools also allow anyone to see who contributed what to a project.

Protected by a set of ground rules, individual scientists also receive rewards for sharing their ideas:

Authorship Each successful polymath project resulted in several papers, linked to the original discussion. Via a self-reporting process, participants put forward their own names for authorship if they made important scientific contributions, or to be mentioned in the acknowledgements for smaller contributions.

Visibility Making many useful contributions may earn you the respect of notable peers, which is valuable in future collaborations or grant and job applications.

Productivity A scientist's time and attention is limited. It is profoundly enjoyable to contribute to many projects where you have a special insight or advantage, while the collaboration dynamically picks up other tasks.

Learning Online discussions are very engaging. Nascent ideas are quickly developed, or discarded, often leading to new knowledge or insight into the thought patterns of others.You are also encouraged to share an idea before someone else gets the same idea.

2.3 Open data

Online tools also collect and organize massive amounts of scientific data which can be mined for interesting patterns. For instance, the Sloan Digital Sky Survey (SDSS) is a collaboration of astronomers operating a telescope that systematically maps the sky, producing a stream of photographs and spectra that currently covers more than a quarter of the sky and more than 930,000 galaxies.[9] Although for a limited time, the data is only available to members of the collaboration, the SDSS decided to share it afterwards with the entire worldwide community of astronomers through an online interface [39].[10] Since then, thousands of new and important discoveries have been made by analysing the data in many new ways [7].

These discoveries are again driven by designed serendipity. Whereas the Polymath projects broadcast a question hoping that many minds may find a solution, the SDSS broadcasts data in the belief that many minds will ask unanticipated questions that lead to new discoveries. Indeed, because the telescope collects more data than a single person can comprehend, it becomes more of a question of asking the right questions than making a single 'correct' interpretation of the data. Moreover, there is also a clear dynamic division of labor: the astronomers who ask interesting questions, the SDSS scientists who collect high quality observations, and the astroinformaticians who mine the data all work together doing what they know best.

Moreover, making the data publicly available is rewarding for the SDSS scientists are well.

Citation Publishing data openly leads to more citation because other scientists can more easily build on them. In this case, other astronomers will use it to answer new questions, and credit the SDSS scientists. In fact, each data release easily collects thousands of citations.

[9]http://skyserver.sdss3.org

[10]The data is also used in Microsoft's WorldWide Telescope (http://www.worldwidetelescope.org) and Google Sky (http://www.google.com/sky).

Funding Sharing the data increases the value of the program to the community as a whole, thus making it easier to secure continued funding. Indeed, if the data was not shared, reviewers may deem that the money is better spent elsewhere [29]. In fact, journals and grant agencies are increasingly expecting all data from publicly funded science to be published after publication.

The evolution towards more open data is not at all limited to astronomy. We are 'mapping and mining' just about every complex phenomenon in nature, including the brain [21; 15], the ocean [18], gene sequences, genetic variants in humans [12], and gene functions [31]. In many of these projects, the data is produced piece by piece by many different scientists, and gathered in a central database which they all can access.

2.4 Citizen science

Online tools are also enhancing the relationship between science and society. In *citizen science* [36], the public is actively involved in scientific research. One example is Galaxy Zoo [24], where citizen scientists are asked to classify the galaxies from the SDSS and other sources such as the Hubble Space Telescope. Within a year, Galaxy Zoo received over 50 million classifications contributed by more than 150,000 people. These classifications led to many new discoveries, and the public data releases are cited hundreds of times.

Once again, designed serendipity occurs naturally. Unexpected observations are reported and discussed on an online forum, and have already resulted in the serendipitous discovery of the previously unknown 'green pea' galaxies [9], 'passive red spirals' [26], and other objects such as 'Hanny's Object' [23], named after the volunteer who discovered it. Moreover, in a dynamic division of labor, citizen scientists take over tasks that are too time-consuming for professional astronomers. In fact, the overall classifications proved more accurate than the classifications made by a single astronomer, and obtained much faster. More engaged volunteers also participate in online discussions, or hunt for specific kinds of objects.

Galaxy Zoo and similar tools are also designed for scalability. The overall task is split up in many small, easy to learn tasks, each volunteer classifies as many galaxies as she wants, and classifications from different users are combined and organized into a coherent whole.

Finally, there are many different reasons for citizen scientists to dedicate their free time [36]. Many are excited to contribute to scientific research. This can be out of a sense of discovery, e.g., being the first to see a particular galaxy, or because they believe in the goal of the project, such as fighting cancer. Many others view it as a game, and find it fun to classify many images. Some citizen science projects explicitly include a gamification component [11], providing leaderboards and immediate feedback to volunteers. Finally, many volunteers simply enjoy learning more about a specific subject, as well as meeting new people with similar interests.

Citizen science is being employed in many more scientific endeavors[11], including protein folding [11], planet hunting [37], classifying plankton[12], and fighting cancer[13]. Many of them are collecting large amounts of valid scientific data, and have yielded important discoveries.

[11]https://www.zooniverse.org/

[12]http://www.planktonportal.org/

[13]http://www.cellslider.net

3. MACHINE LEARNING

Machine learning is a field where a more networked approach would be particularly valuable. Machine learning studies typically involve large data sets, complex code, large-scale evaluations and complex models, none of which can be adequately represented in papers. Still, most work is only published in papers, in highly summarized forms such as tables, graphs and pseudo-code. Oddly enough, while machine learning has proven so crucial in analysing large amounts of observations collected by other scientists, such as the SDSS data discussed above, the outputs of machine learning research are typically not collected and organized in any way that allows others to reuse, reinterpret, or mine these results to learn new things, e.g., which techniques are most useful in a given application.

3.1 Reusability and reproducibility

This makes us duplicate a lot of effort, and ultimately slows down the whole field of machine learning [43; 14]. Indeed, without prior experiments to build on, each study has to start from scratch and has to rerun many experiments. This limits the depth of studies and the interpretability and generalizability of their results [1; 14]. It has been shown that studies regularly contradict each other because they are biased toward different datasets [20], or because they don't take into account the effects of dataset size, parameter optimization and feature selection [33; 17]. This makes it very hard, especially for other researchers, to correctly interpret the results. Moreover, it is often not even possible to rerun experiments because code and data are missing, or because space restrictions imposed on publications make it practically infeasible to publish many details of the experiment setup. This lack of reproducibility has been warned against repeatedly [20; 38; 32], and has been highlighted as one of the most important challenges in data mining research [16].

3.2 Prior work

Many machine learning researchers are well aware of these issues, and have worked to alleviate them. To improve reproducibility, there exist repositories to publicly share benchmarking datasets, such as UCI [2], LDC[14] and mldata[15]. Moreover, software can be shared on the MLOSS website[16]. There also exists an open source software track in the Journal for Machine Learning Research (JMLR) where short descriptions of useful machine learning software can be submitted. Also, some major conferences have started checking submissions for reproducibility [25], or issue open science awards for submissions that are reproducible.[17]

Moreover, there also exist experiment repositories. First, meta-learning projects such as StatLog [27] and MetaL [8], and benchmarking services such as MLcomp[18] run many algorithms on many datasets on their servers. This makes benchmarks comparable, and even allows the building of meta-models, but it does require that code be rewritten to run on their servers. Moreover, the results are not organized to be easily queried and reused.

Second, data mining challenge platforms such as Kaggle [10] and TunedIT [44] collect results obtained by different competitors. While they do scale and offer monetary incentives, they are adversarial rather than collaborative. For instance, code is typically not shared during a competition.

Finally, we previously introduced the *experiment database for machine learning* [6; 43], which organizes results from different users and makes them queryable through an online interface. Unfortunately, it doesn't allow collaborations to scale easily. It requires researchers to transcribe their experiments into XML, and only covers classification experiments.

While all these tools are very valuable in their own right, and we will build on them in this paper, they fail many of the requirements for scalable collaboration discussed above. It can be quite hard for scientists to contribute, there is often no online discussion, and they are heavily focused on benchmarking, not on sharing other results such as models.

4. OPENML

OpenML[19] is a place where machine learning researchers can automatically share data in fine detail and organize it to work more effectively and collaborate on a global scale.

It allows anyone to challenge the community with new data to analyze, and everyone able to mine that data to share their code and results (e.g., models, predictions, and evaluations).[20] OpenML makes sure that each (sub)task is clearly defined, and that all shared results are stored and organized online for easy access, reuse and discussion.

Moreover, OpenML links to data available anywhere online, and is being integrated [41] in popular data mining platforms such as Weka [13], R [5; 40], MOA [4], RapidMiner [42] and KNIME [3]. This means that anyone can easily import the data into these tools, pick any algorithm or workflow to run, and automatically share all obtained results. The OpenML website provides easy access to all collected data and code, compares all results obtained on the same data or algorithms, builds data visualizations, and supports online discussions.

Finally, it is an open source project, inviting scientists to extend it in ways most useful to them.

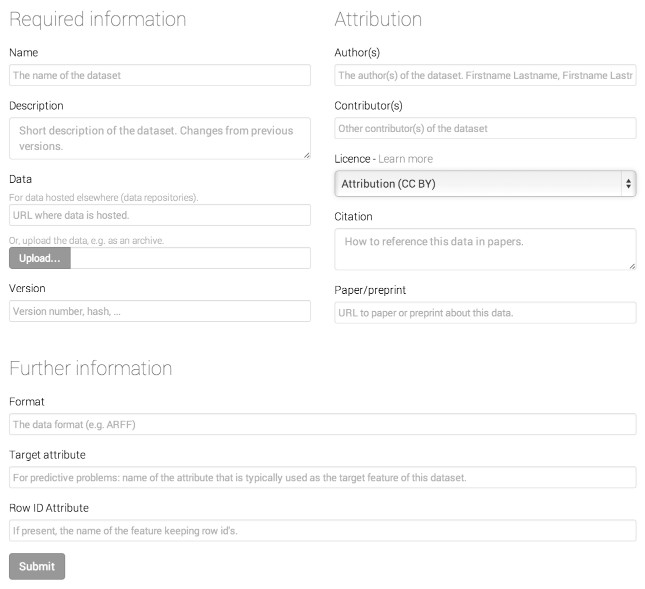
4.1 How OpenML works

OpenML offers various services to share and find data sets, to download or create scientific *tasks*, to share and find implementations (called *flows*), and to share and organize results. These services are available through the OpenML website, as well as through a REST API for integration with software tools.[21]

4.1.1 Data sets

Anyone can challenge the community with new data sets to analyze. Figure 1 shows how this is done through the website. To be able to analyse the data, OpenML accepts a limited number of formats. For instance, currently it requires the ARFF[22] format for tabular data, although more formats will be added over time.

[14]http://www.ldc.upenn.edu
[15]http://mldata.org
[16]http://mloss.org
[17]http://www.ecmlpkdd2013.org/open-science-award/
[18]http://www.mlcomp.org
[19]OpenML is available on http://www.openml.org
[20]In this sense, OpenML is similar to data mining challenge platforms, except that it allows users to work collaboratively, building on each other's work.
[21]In this paper, we only discuss the web interfaces. API details can be found on http://www.openml.org/api
[22]http://www.cs.waikato.ac.nz/ml/weka/arff.html



Required information

Name

The name of the dataset

Description

Short description of the dataset. Changes from previous versions.

Data

For data hosted elsewhere (data repositories).

URL where data is hosted.

Or, upload the data, e.g. as an archive.

Upload...

Version

Version number, hash, ...

Attribution

Author(s)

The author(s) of the dataset. Firstname Lastname, Firstname Lastr

Contributor(s)

Other contributor(s) of the dataset

Licence - Learn more

Attribution (CC BY)

Citation

How to reference this data in papers.

Paper/preprint

URL to paper or preprint about this data.

Further information

Format

The data format (e.g. ARFF)

Target attribute

For predictive problems: name of the attribute that is typically used as the target feature of this dataset.

Row ID Attribute

If present, the name of the feature keeping row id's.

Submit

Figure 1: Uploading data to OpenML.

The data can either be uploaded or referenced by a URL. This URL may be a landing page with further information or terms of use, or it may be an API call to large repositories of scientific data such as the SDSS.[23] OpenML will automatically version each newly added data set. Optionally, a user-defined version name can be added for reference. Next, authors can state how the data should be attributed, and which (creative commons) licence they wish to attach to it. Authors can also add a reference for citation, and a link to a paper. Finally, extra information can be added, such as the (default) target attribute(s) in labeled data, or the row-id attribute for data where instances are named.

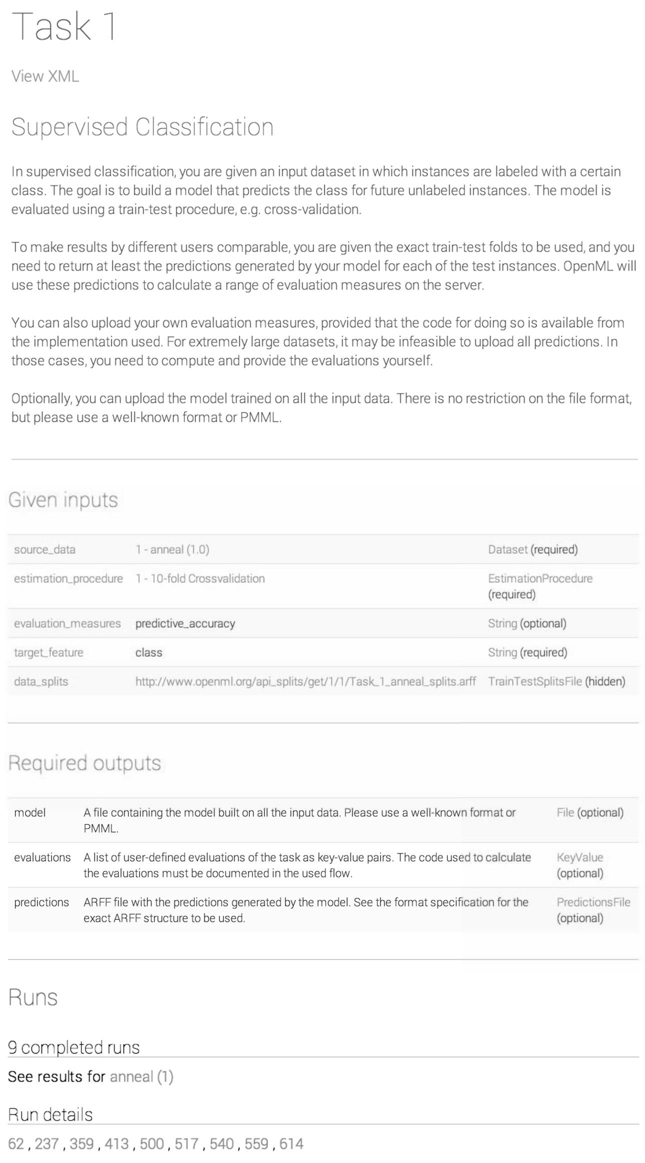
For known data formats, OpenML will then compute an array of data characteristics. For tabular data, OpenML currently computes more than 70 characteristics[24], including simple measures (e.g., the number of features, instances, classes, missing values), statistical and information-theoretic measures (e.g., skewness, mutual information) and landmarkers [34]. Some characteristics are specific to subtypes of data, such as data streams. These characteristics are useful to link the performance of algorithms to data characteristics, or for meta-learning [43] and algorithm selection [22].

OpenML indexes all data sets and allows them to be searched through a standard keyword search and search filters. Each data set has its own page with all known information.[25] This includes the general description, attribution information, and data characteristics, but also statistics of the data distribution and, for each *task* defined on this data (see below), all results obtained for that task. As will be discussed below, this allows you to quickly see which algorithms (and parameters) are best suited for analysing the data, and who achieved these results. It also includes a discussion section where the data set and results can be discussed.

[23]In some cases, such as Twitter feeds, data may be dynamic, which means that results won't be repeatable. However, in such tasks, repeatability is not expected.

[24]A full list can be found on http://www.openml.org/a

[25]See, for instance, http://www.openml.org/d/1



Task 1

View XML

Supervised Classification

In supervised classification, you are given an input dataset in which instances are labeled with a certain class. The goal is to build a model that predicts the class for future unlabeled instances. The model is evaluated using a train-test procedure, e.g. cross-validation.

To make results by different users comparable, you are given the exact train-test folds to be used, and you need to return at least the predictions generated by your model for each of the test instances. OpenML will use these predictions to calculate a range of evaluation measures on the server.

You can also upload your own evaluation measures, provided that the code for doing so is available from the implementation used. For extremely large datasets, it may be infeasible to upload all predictions. In those cases, you need to compute and provide the evaluations yourself.

Optionally, you can upload the model trained on all the input data. There is no restriction on the file format, but please use a well-known format or PMML.

Given inputs

source_data	1 - anneal (1.0)	Dataset (required)
estimation_procedure	1 - 10-fold Crossvalidation	EstimationProcedure (required)
evaluation_measures	predictive_accuracy	String (optional)
target_feature	class	String (required)
data_splits	http://www.openml.org/api_splits/get/1/1/Task_1_anneal_splits.arff	TrainTestSplitsFile (hidden)

Required outputs

model	A file containing the model built on all the input data. Please use a well-known format or PMML.	File (optional)
evaluations	A list of user-defined evaluations of the task as key-value pairs. The code used to calculate the evaluations must be documented in the used flow.	KeyValue (optional)
predictions	ARFF file with the predictions generated by the model. See the format specification for the exact ARFF structure to be used.	PredictionsFile (optional)

Runs

9 completed runs

See results for anneal (1)

Run details

62 , 237 , 359 , 413 , 500 , 517 , 540 , 559 , 614

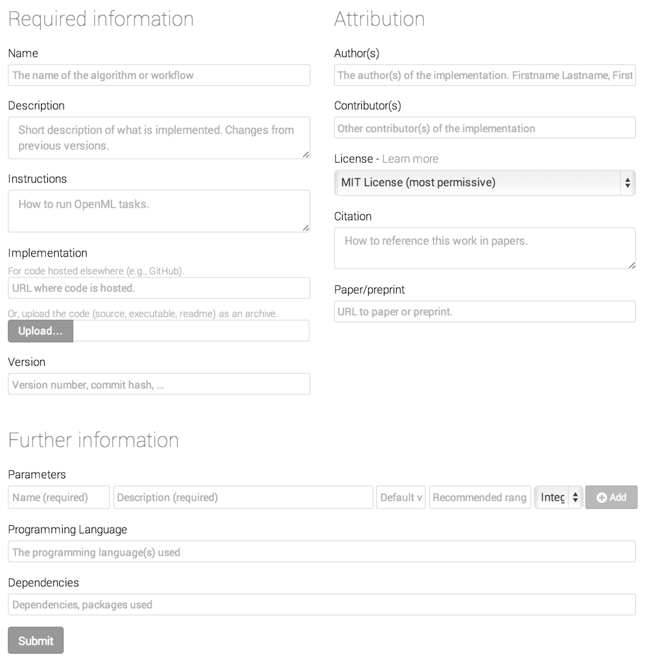
Figure 2: An OpenML task (of task type classification).

4.1.2 *Task types*

Obviously, a data set alone does not constitute a scientific challenge. We must first agree on what types of results are expected to be shared. This is expressed in *task types*: they define what types of inputs are given, which types of output are expected to be returned, and what scientific protocols should be used. For instance, classification tasks should include well-defined cross-validation procedures, labeled input data, and require predictions as outputs.[26]

OpenML currently covers classification, regression, learning curve analysis and data stream classification. Task types are created by machine learning (sub)communities through the website, and express what they think should ideally be shared. In some cases, additional support may be required, such as running server-side evaluations. Such support will be provided upon request.

[26]Complete description: http://www.openml.org/t/type/1



Required information

Name

The name of the algorithm or workflow

Description

Short description of what is implemented. Changes from previous versions.

Instructions

How to run OpenML tasks.

Implementation

For code hosted elsewhere (e.g., GitHub).

URL where code is hosted.

Or, upload the code (source, executable, readme) as an archive.

Upload...

Version

Version number, commit hash, ...

Attribution

Author(s)

The author(s) of the implementation. Firstname Lastname, First

Contributor(s)

Other contributor(s) of the implementation

License - Learn more

MIT License (most permissive)

Citation

How to reference this work in papers.

Paper/preprint

URL to paper or preprint.

Further information

Parameters

Name (required) | Description (required) | Default v | Recommended rang | Integ | Add

Programming Language

The programming language(s) used

Dependencies

Dependencies, packages used

Submit

Figure 3: Uploading flows to OpenML.

4.1.3 *Tasks*

If scientists want to perform, for instance, classification on a given data set, they can create a new machine learning *task* online. Tasks are instantiations of task types with specific inputs (e.g., data sets). Tasks are created once, and then downloaded and solved by anyone.

Such a task is shown in Figure 2. In this case, it is a classification task defined on data set 'anneal' version 1. Next to the data set, the task includes the target attribute, the evaluation procedure (here: 10-fold cross-validation) and a file with the data splits for cross-validation. The latter ensures that results from different researchers can be objectively compared. For researchers doing an (internal) hyperparameter optimization, it also states the evaluation measure to optimize for. The required outputs for this task are the predictions for all test instances, and optionally, the models built and evaluations calculated by the user. However, OpenML will also compute a large range of evaluation measures on the server to ensure objective comparison.[27]

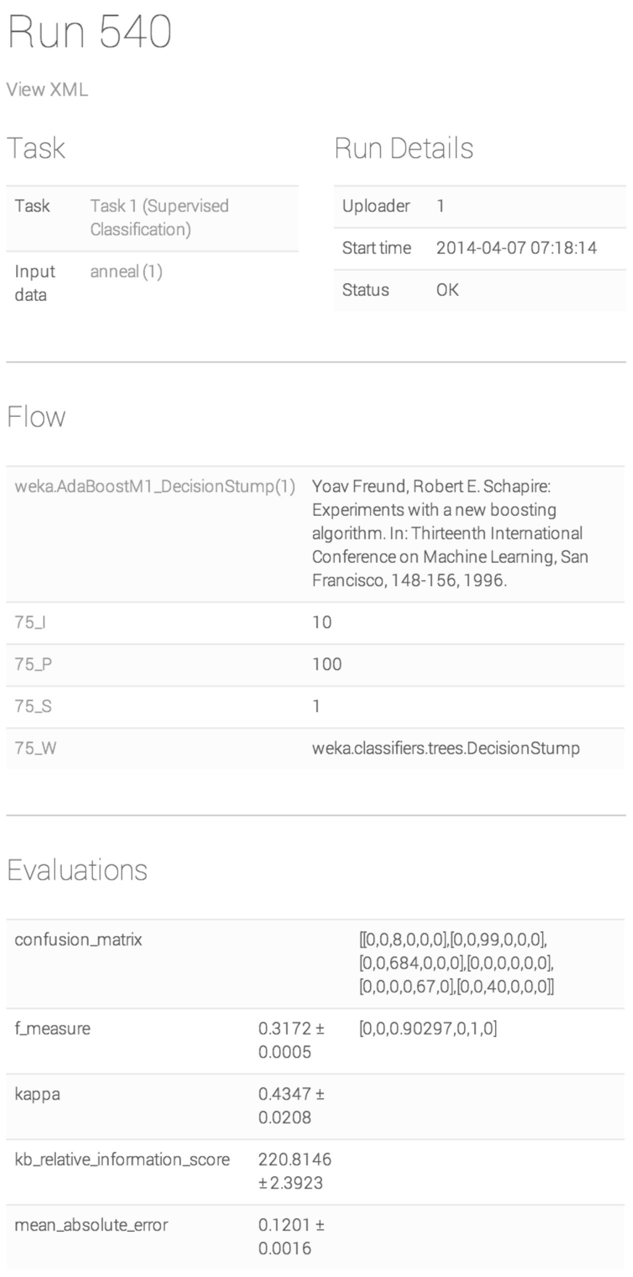
Finally, each task has its own numeric id, a machine-readable XML description, as well as its own web page including all runs uploaded for that task, see Figure 2.

4.1.4 *Flows*

Flows are implementations of single algorithms, workflows, or scripts designed to solve a given task. They are uploaded to OpenML as shown in Figure 3. Again, one can upload the actual code, or reference it by URL. The latter is especially useful if the code is hosted on an open source platform such as GitHub or CRAN. Flows can be updated as often as needed. OpenML will again version each uploaded flow, while users can provide their own version name for reference. Ideally, what is uploaded is software that takes a task id as input and then produces the required outputs. This can be a wrapper around a more general implementation. If not, the description should include instructions detailing how users can run an OpenML task (e.g., to verify submitted results). Attribution information is similar to that provided for data sets, although with a different set of licences. Finally, it is encouraged to add descriptions for the (hyper)parameters of the flow, and a range of recommended values.

It is also possible to annotate flows with characteristics, such as whether it can handle missing attributes, (non)numeric features and (non)numeric targets. As with data sets, each flow has its own page which combines all known information and all results obtained by running the flow on OpenML tasks, as well as a discussion section.

[27]The evaluation measures and the exact code can be found on http://www.openml.org/a.



Run 540

View XML

Task

Task	Task 1 (Supervised Classification)
Input data	anneal (1)

Run Details

Uploader	1
Start time	2014-04-07 07:18:14
Status	OK

Flow

weka.AdaBoostM1_DecisionStump(1)	Yoav Freund, Robert E. Schapire: Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning, San Francisco, 148-156, 1996.
75_I	10
75_P	100
75_S	1
75_W	weka.classifiers.trees.DecisionStump

Evaluations

confusion_matrix		[[0,0,8,0,0,0],[0,0,99,0,0,0],[0,0,684,0,0,0],[0,0,0,0,0,0],[0,0,0,0,67,0],[0,0,40,0,0,0]]
f_measure	0.3172 ± 0.0005	[0,0,0.90297,0,1,0]
kappa	0.4347 ± 0.0208	
kb_relative_information_score	220.8146 ± 2.3923	
mean_absolute_error	0.1201 ± 0.0016	

Figure 4: An OpenML run.

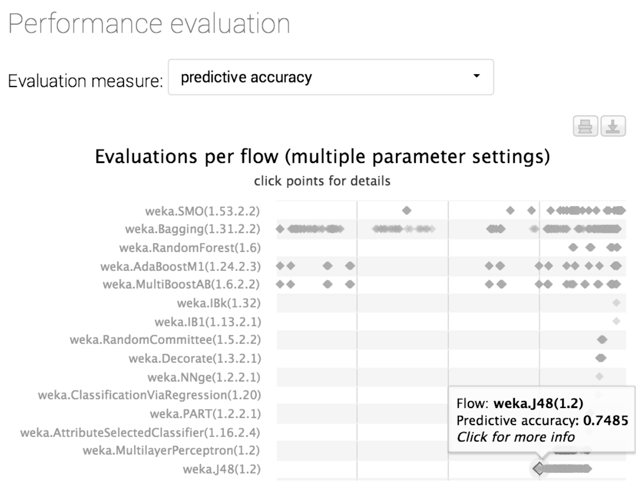


Figure 5: Portion of the page for data set 'anneal'. It compares, for a classification task, the results obtained by different flows on that data set, for multiple parameter settings.

4.1.5 *Runs*

Runs are applications of flows on a specific task. They are submitted by uploading the required outputs (e.g. predictions) together with the task id, the flow id, and any parameter settings. There is also a flag that indicates whether these parameter settings are default values, part of a parameter sweep, or optimized internally. Each run also has its own page with all details and results, shown partially in Figure 4. In this case, it is a classification run. Note that OpenML stores the distribution of evaluations per fold (shown here as standard deviations), and details such as the complete confusion matrix and per-class results. We plan to soon add graphical measures such as ROC curves. Runtimes and details on hardware are provided by the user.

Moreover, because each run is linked to a specific task, flow, and author, OpenML will aggregate and visualize results accordingly.

For instance, Figure 5 shows a comparison of results obtained on a specific classification task. Each row represents a flow and each dot represents the performance obtained in a specific run (for different parameter settings). Hovering over a dot reveals more information, while clicking on it will pull up all information about the run. Users can also switch between different performance metrics.

Conversely, Figure 6 shows a comparison of results obtained by a specific flow on all tasks it has run on. Each row represents a task (and data set), and each dot the obtained performance. Additionally, it is possible to color-code the results with parameter values. Here, it shows the number of trees used in a random forest classifier from small (blue, left) to large (red, right). Again, clicking each dot brings up all run details. As such, it is easy to find out when, how, and by whom a certain result was obtained.

OpenML also provides several other task-specific visualizations such as learning curves. Moreover, it provides an SQL endpoint so that users can (re)organize results as they wish by writing their own queries. All results can be downloaded from the website for further study, and all visualizations can be exported. The database can also be queried programmatically through the API.

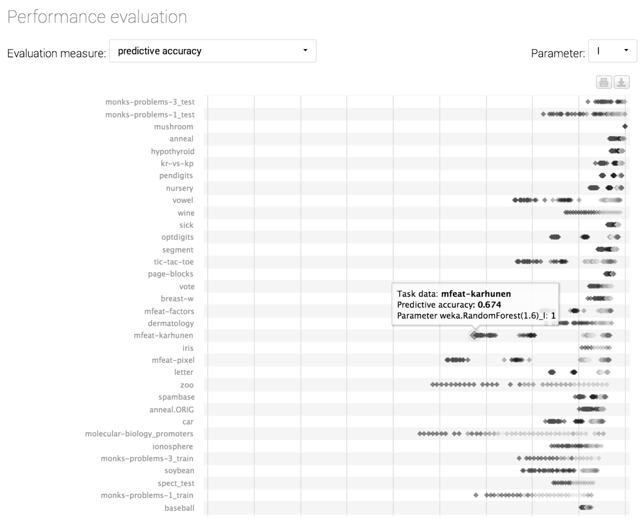


Figure 6: Portion of the page for flow 'weka.RandomForest'. It compares, for classification tasks on different data sets, the results obtained by this flow for different parameter settings. Here, colored by the number of trees in the forest.

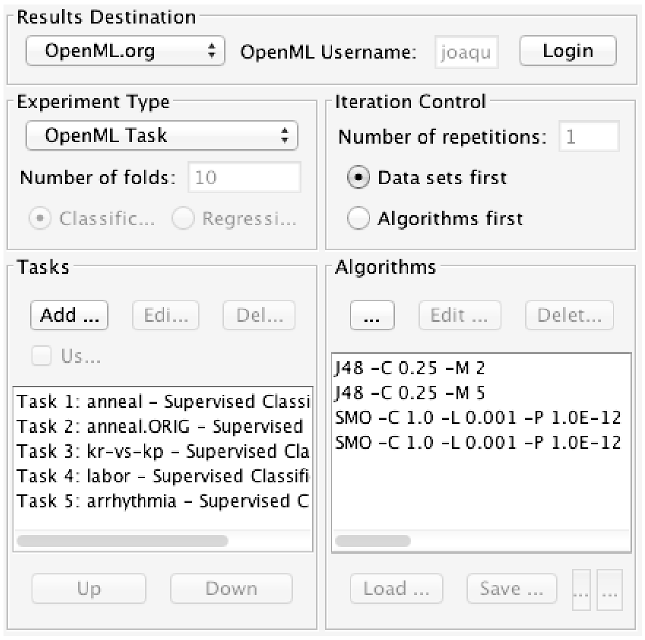
4.2 OpenML plugins

As stated above, OpenML features an API so that scientific software tools can connect to it to download data or upload new results. However, existing tools can also connect to OpenML by simply adding a few lines of code, and without any knowledge of the API. Indeed, OpenML provides language-specific libraries that take care of all server communication. For instance, software written in Java would use the OpenML Java interface to download tasks and upload results. More precisely, a method *getTask(id)* will return a Java object containing all data to run the task, and a method *submitRun(outputs)* will take the obtained results and submit them to OpenML. We also aim to provide command-line tools for connecting to OpenML.

On top of that, OpenML is being integrated in several popular machine learning environments, so that it can be used out of the box. These *plugins* can be downloaded from the website. Figure 7 shows how OpenML is integrated in WEKA's Experimenter [13]. After selecting OpenML as the result destination and providing your login credentials, you can add a number of tasks through a dialogue (or simply provide a list of task id's), and add a number of WEKA algorithms to run. Behind the scenes, the plugin will download all data, run every algorithm on every task (if possible) and automatically upload the results to OpenML. The results will also be locally available in WEKA for further analysis.

For data stream mining, one can use the MOA [4] plugin as shown in Figure 8. Similarly to WEKA, users can select OpenML tasks, and then run any algorithm on them while uploading all runs to OpenML in the background.

Finally, researchers that use R can use the *openml* package as shown in Figure 9. One first downloads a task given a task id, then runs the task by providing a learner, and finally uploads the run by providing the task, learner, results and user authentication. While we do plan to integrate OpenML into other environments as well, at the time of writing these are still in development.



Results Destination
OpenML.org OpenML Username: joaqu Login

Experiment Type
OpenML Task
Number of folds: 10
[✓ Classific...] Regressi...

Iteration Control
Number of repetitions: 1
[✓ Data sets first]
Algorithms first

Tasks
Add ... Edi... Del...
Us...

```
Task 1: anneal - Supervised Classi
Task 2: anneal.ORIG - Supervised
Task 3: kr-vs-kp - Supervised Cla
Task 4: labor - Supervised Classifi
Task 5: arrhythmia - Supervised C
```

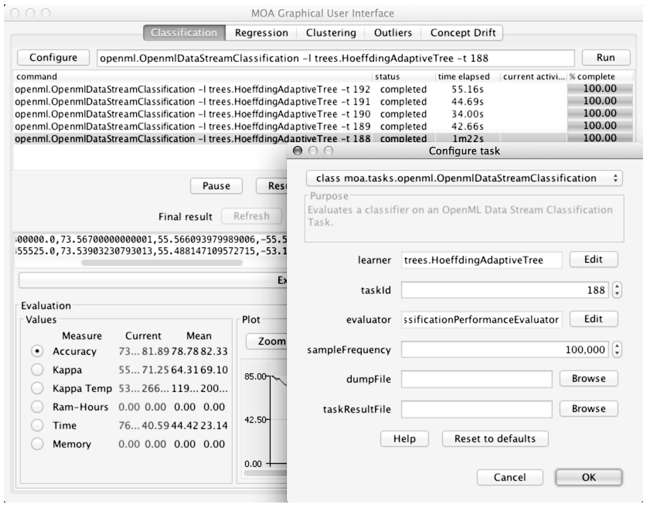
Up Down

Algorithms
... Edit ... Delet...

```
J48 -C 0.25 -M 2
J48 -C 0.25 -M 5
SMO -C 1.0 -L 0.001 -P 1.0E-12
SMO -C 1.0 -L 0.001 -P 1.0E-12
```

Load ... Save ... ...

Figure 7: WEKA integration of OpenML.



MOA Graphical User Interface
Classification | Regression | Clustering | Outliers | Concept Drift

Configure: openml.OpenmlDataStreamClassification -l trees.HoeffdingAdaptiveTree -t 188 Run

```
command                                                             status     time elapsed  current activi... % complete
openml.OpenmlDataStreamClassification -l trees.HoeffdingAdaptiveTree -t 192  completed  55.16s   100.00
openml.OpenmlDataStreamClassification -l trees.HoeffdingAdaptiveTree -t 191  completed  44.69s   100.00
openml.OpenmlDataStreamClassification -l trees.HoeffdingAdaptiveTree -t 190  completed  34.00s   100.00
openml.OpenmlDataStreamClassification -l trees.HoeffdingAdaptiveTree -t 189  completed  42.66s   100.00
openml.OpenmlDataStreamClassification -l trees.HoeffdingAdaptiveTree -t 188  completed  1m22s    100.00
```

Pause Res

Final result Refresh

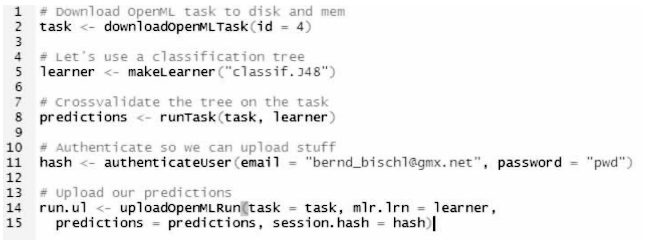
```
00000.0,73.56700000000001,55.566093979989006,-55.5
55525.0,73.53903230793013,55.488147109572715,-53.1
```

Evaluation
Values

```
  Measure      Current       Mean
● Accuracy     73... 81.89  78.78 82.33
○ Kappa        55... 71.25  64.31 69.10
○ Kappa Temp   53... 266... 119... 200...
○ Ram-Hours    0.00  0.00   0.00  0.00
○ Time         76... 40.59  44.42 23.14
○ Memory       0.00  0.00   0.00  0.00
```

Configure task
class moa.tasks.openml.OpenmlDataStreamClassification
Purpose: Evaluates a classifier on an OpenML Data Stream Classification Task.
learner: trees.HoeffdingAdaptiveTree Edit
taskId: 188
evaluator: ssificationPerformanceEvaluator Edit
sampleFrequency: 100,000
dumpFile: Browse
taskResultFile: Browse
Help Reset to defaults
Cancel OK

Figure 8: MOA integration of OpenML.



```r
# Download OpenML task to disk and mem
task <- downloadOpenMLTask(id = 4)

# Let's use a classification tree
learner <- makeLearner("classif.J48")

# Crossvalidate the tree on the task
predictions <- runTask(task, learner)

# Authenticate so we can upload stuff
hash <- authenticateUser(email = "bernd_bischl@gmx.net", password = "pwd")

# Upload our predictions
run.ul <- uploadOpenMLRun(task = task, mlr.lrn = learner,
  predictions = predictions, session.hash = hash)
```

Figure 9: R integration of OpenML.

5. NETWORKED MACHINE LEARNING

Through OpenML, we can initiate a fully networked approach to machine learning. In this section we compare OpenML to the networked science tools described before, and describe how it helps scientists make new discoveries, how it allows collaborations to scale, and how it benefits individual scientists, students and a more general audience.

5.1 OpenML and networked science

By sharing and organizing machine learning data sets, code and experimental results at scale, we can stimulate designed serendipity and a dynamic division of labor.

5.1.1 *Designed serendipity*

Similar to the SDSS, by organizing and 'broadcasting' all data, code and experiments, many minds may reuse them in novel, unforeseen ways.

First, new discoveries could by made simply by *querying* all combined experiments to answer interesting questions. These question may have been nearly impossible to answer before, but are easily answered if a lot of data is already available. In addition, with readily available data, it becomes a routine part of research to answer questions such as "What is the effect of data set size on runtime?" or "How important is it to tune hyperparameter P?" With OpenML, we can answer these questions in minutes, instead of having to spend days setting up and running new experiments [43]. This means that more such questions will be asked, possibly leading to more discoveries.

Second, we can *mine* all collected results and data characteristics for patterns in algorithm performance. Such *meta-learning* studies could yield insight into which techniques are most suited for certain applications, or to better understand and improve machine learning techniques [43].

Third, anyone could run into unexpected results by browsing through all collected data. An example of this is shown in Figure 10, which is a continuation of the results shown in Figure 6: while the performance of a random forest classifier should increase (or stagnate) when more trees are added to the forest (red dots), it sometimes happens that it decreases. As in Galaxy Zoo, such serendipitous discoveries can be discussed online, combining many minds to explore several possible explanations.

Finally, beyond experiments, data sets and flows can also be reused in novel ways. For instance, an existing technique may prove extremely useful for analysing a new data set, bringing about new applications.

5.1.2 *Dynamic division of labor*

OpenML also enables a dynamic division of labor: large-scale studies could be undertaken as a team, or hard questions could be tackled collaboratively, with many scientists contributing according to their specific skills, time or resources.

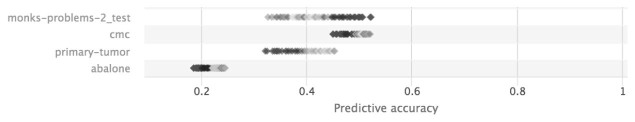


Figure 10: An unexpected result: the performance of a random forest descreases as more trees are added to the forest.

Scientists, possibly from other domains, can focus the attention of the community on an important problem. This can be done by adding new data sets (and tasks) to OpenML and collaborating with the machine learning community to analyse it. Some may suggest techniques that would otherwise not be considered, while others are especially skilled at designing custom-built workflows, running large-scale experiments, or improving code. Conversely, the scientists that contributed the data can provide direct feedback on the practical utility of suggested approaches, interpret the generated models, and otherwise guide the collaboration to the desired outcome. Such collaborations can scale to any number of scientists. OpenML also helps to coordinate the effort, e.g., by organizing all results per task (see Figure 6), so that everybody can track each other's progress, and discuss ideas and results online.

Another case is that of benchmark studies. While it is important that a new algorithm be compared against the state of the art, it is often time-consuming to hunt down their implementations and to figure out how to run them. On OpenML, each researcher that invents a new algorithm can focus on experimenting with that algorithm alone, knowing best how to apply it on different tasks. Next, she can instantly reuse the results from all other shared algorithms, ran by their original authors. As such, a very complete overview of the state of the art emerges spontaneously.

Finally, students and citizen scientists can also contribute to research simply by using the OpenML plugins to experiment while they learn about machine learning techniques.

5.2 Scaling up collaboration

As discussed in section 2.1, these benefits emerge faster if online collaborations are allowed to scale.

First of all, it is *easy to make small contributions* to OpenML. When using any of the OpenML plugins, you can easily import a task, run any algorithm or workflow, and automatically export the results to OpenML. You can just perform a single run, a few, or thousands without much effort. Moreover, scientists who know of new interesting data sets or algorithms can easily add them through the website, and watch how others start experimenting with them. It is even easier to browse through the discussions running on OpenML, and leave a comment or suggestion. Alternatively, one can browse the results shared on the website, and draw attention to unexpected results that are worth investigating. Even contributing a single run, data set or comment can be valuable to the community, and may stimulate more work in that direction. More committed scientists can contribute in many other ways, such as creating new tasks and task types, adding new data characterizations or evaluation measures, and integrating OpenML in new tools and environments.

Moreover, OpenML tasks naturally *split up complex studies* into tasks which can be run independently by many scientsists according to their skills, as discussed in Section 5.1. Tasks also split the machine learning community into smaller subcommunities (e.g., clustering) which focus on a single task type, or subgroups focusing on a single task (e.g. galaxy clustering). Designed serendipity and dynamic division of labor also occur in small but active subcommunities. They are not held back if other communities are less active.

Next, OpenML constructs a *rich and structured information commons*, building a database of all data sets, flows, tasks, runs, results, scientists, and discussions. OpenML also aggregates results in different ways, e.g., visualizing results per task and flow (see Figures 5 and 6). Keyword searches and filters make it easy to find resources, and more complex questions can be answered through the SQL interface, or by downloading data and analysing it using other tools.

As a result, all information is also *open but easily filtered*. The website organizes all results per task, data set and flow, so that researchers can focus on what interests them most, without being distracted by the activity of other scientists. In future work, we also aim to filter results by their authors.

Finally, OpenML establishes, and in some cases enforces, a scientific approach to sharing results. Indeed, OpenML tasks set a certain standard of scientific quality and trustworthiness by defining how experiments must be run and what must be reported. Because the code is shared when uploading runs, it is possible for others to verify results, and the server-side evaluations makes results objectively comparable. OpenML also makes clear who contributed what (and when), and how it is licenced. Every shared algorithm, flow, run or comment can be attributed to a specific person, and this information is always shown when someone views them online.

5.3 Benefits for scientists

How do you, as an individual scientist, benefit from sharing your experiments, data and code on OpenML?

5.3.1 More time

First, you gain more time. OpenML assists in most of the routine and tedious duties in running experiments: finding data sets, finding implementations, setting up experiments, and organizing all experiments for further analysis. Moreover, when running benchmark experiments on OpenML, you can directly compare them with the state of the art, reusing other, comparable results. In addition, you can answer routine research question in minutes by tapping into all shared data, instead of losing days setting up new experiments. Finally, having your experiments stored and organized online means they are available any place, any time, through any browser (including mobile devices), so you can access them when it is convenient.

5.3.2 More knowledge

Second, you gain more knowledge. Linking your results to everybody else's has a large potential for new discoveries. This was discussed in Section 5.1: you can answer previously impossible questions, mine all combined data, and run into unexpected results. It also makes it easy to check whether certain observations in your data are echoed in the observations of others. Next, with OpenML you can interact with other minds on a global scale. Not only can you start discussions to answer your own questions, you can also help others, and in doing so, learn about other interesting studies, and forge new collaborations. Finally, by reusing prior results, you can launch larger, more generalizable studies that are practically impossible to run on your own.

5.3.3 More reputation

Third, OpenML helps you build reputation by making your work more visible to a wider group of people, by bringing you closer to new collaborators, and by making sure that others know how to credit you if they build on any of your work.

Citation OpenML makes sure that all your contributions, every data set, flow and run, are clearly attributed to you. If others wish to reuse your results, OpenML will tell them how you wish to be credited (e.g., through citation). Moreover, OpenML makes your shared resources easy to find, thus making frequent citations more likely.

Altmetrics OpenML will also automatically track how often your data or code is reused in experiments (runs), and how often your experiments are reused in studies (see below). These are clear measures of the impact of your work.

Productivity OpenML allows you to contribute efficiently to many studies. This increases your scientific productivity, which translates to more publications.

Visibility You can increase your visibility by contributing to many studies, thus earning the respect of new peers. Also, if you design flows that outperform many others, OpenML will show these at the top of each data set page, as shown in Figure 5. You can also post links to your online results in blogs or tweets.

Funding Open data sharing is increasingly becoming a requirement in grant proposals, and uploading your research to OpenML is a practical and convincing way to share your data with others.

No publication bias Most journals have a publication bias: even if the findings are valuable, it is hard to publish them if the outcome is not positive. Through OpenML, you can still share such results and receive credit for them.

5.4 Benefits for students

OpenML can also substantially help students in gaining a better understanding of machine learning. Browsing through organized results online is much more accessible than browsing through hundreds of papers. It provides a clear overview of the state of the art, interesting new techniques and open problems. As discussed before, students can contribute in small or big ways to ongoing research, and in doing so, learn more about how to become a machine learning researcher. Online discussions may point to new ideas or point out mistakes, so they can learn to do it better next time. In short, it gives students and young scientists a large playground to learn more quickly about machine learning and discover where they can make important contributions.

6. FUTURE WORK

In this section, we briefly discuss some of the key suggestions that have been offered to improve OpenML, and we aim to implement these changes as soon as possible. In fact, as OpenML is an open source project, everyone is welcome to help extend it, or post new suggestions through the website.

6.1 OpenML studies

One typically runs experiments as part of a study, which ideally leads to a publication. Scientists should therefore be able to create *online studies* on OpenML, that combine all relevant info on one page. Such studies reference all runs of interest, either generated for this study or imported from other OpenML studies, and all data sets and flows underlying these runs. Additionally, textual descriptions can be added to explain what the study is about, and any supplementary materials, such as figures, papers or additional data, can be uploaded and attached to it.

If the study is published, a link to this online study can be added in the paper, so that people can find the original experiments, data sets and flows, and build on them. As such, it becomes the online counterpart of a published paper, and you could tell people to cite the published paper if they reuse any of the data. An additional benefit of online studies is that they can be extended after publication.

Moreover, based on the underlying runs, OpenML can automatically generate a list of references (citations) for all underlying data sets, flows and other studies. This helps authors to properly credit data that they reused from other OpenML scientists. Similar to arXiv, OpenML can also automatically keep track of this so that authors can instantly view in which studies their contributions are being reused.

Finally, similar to the polymath projects, such studies could be massively collaborative studies, aimed at solving a hard problem, and driven by many people providing ideas, experiments, data or flows. As such, each study should have a discussion section where questions can be asked, suggestions can be made and progress can be discussed. Similar to the polymath studies, it also makes sense if studies link to other studies that tackle specific subproblems, and if the study was linked to a wiki page for collaborative writing.

To keep focus on the results of a given study, it can act as a filter, hiding all other results from the website so that only the contents of that study are visible.

6.2 Visibility and social sharing

There may be cases where you want all of OpenML's benefits, but do not want to make your data public before publication. On the other hand, you may want to share that data with trusted colleagues for collaboration or feedback, or allow friends to edit your studies, e.g., to add new runs to it. It therefore makes sense if, for a limited amount of time, studies, data sets or flows can be flagged as private or 'friends only'. Still, scientists should agree that this is a temporary situation. When you wish to be attributed for your work, you must first make it publicly available.

In addition, in highly experimental settings, most results may not be interesting as such. Scientists are always able to delete those results or mark them as deprecated.

6.3 Collaborative leaderboards

When many scientists work together to design a flow for a specific task, it is useful to show which flows are currently performing best, so that others can build on and improve those flows. However, it is also important to show which contributions had the biggest impact while such a flow was constructed collaboratively. In those cases, it is useful to implement a leaderboard that does not only show the current best solution, but instead credits the authors who contributed solutions that were in, say, the top 3 at any point in time. Alternatively, this can be visualized in a graph of performance versus time, so that it is clear who caused the bigger performance 'jumps'. This is useful to later credit the people who made the most important contributions.

6.4 Broader data support

To build a well-structured information space, OpenML needs to be able to correctly interpret the uploaded data. For instance, to calculate data characteristics and build train-test splits, more information is needed about the structure of data sets. Therefore, we have initially focused on ARFF data. However, many types of data, such as graphs, can not always be adequately expressed as ARFF, and we will add support for new data types according to researchers' needs. Moreover, for some types of tasks, additional types of results (run outputs) may need to be defined so that OpenML can properly interpret them.

In the short term, we aim to add support for Graph Mining, Clustering, Recommender Systems, Survival Analysis, Multi-label Classification, Feature selection, Semi-Supervised Learning, and Text Mining. Moreover, we will extend the website to make it easy to propose and work collaboratively on support for new task types.

7. CONCLUSIONS

In many sciences, networked science tools are allowing scientists to make discoveries much faster than was ever possible before. Hundreds of scientists are collaborating to tackle hard problems, individual scientists are building directly on the observations of all others, and students and citizen scientists are effectively contributing to real science.

To bring these same benefits to machine learning researchers, we introduce OpenML, an online service to share, organize and reuse data, code and experiments. Following best practices observed in other sciences, OpenML allows collaborations to scale effortlessly and rewards scientists for sharing their data more openly.

We believe that this new, networked approach to machine learning will allow scientists to work more productively, make new discoveries faster, be more visible, forge many new collaborations, and start new types of studies that were practically impossible before.

8. ACKNOWLEDGEMENTS

This work is supported by grant 600.065.120.12N150 from the Dutch Fund for Scientific Research (NWO), and by the IST Programme of the European Community, under the Harvest Programme of the PASCAL2 Network of Excellence, IST-2007-216886.

We also wish to thank Hendrik Blockeel, Simon Fisher and Michael Berthold for their advice and comments.

9. REFERENCES

[1] D. W. Aha. Generalizing from case studies: a case study. In *Proceedings of the ninth international workshop on Machine learning*, pages 1–10, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[2] A. Asuncion and D. Newman. UCI machine learning repository. *University of California, School of Information and Computer Science*, 2007.

[3] M. Berthold, N. Cebron, F. Dill, T. Gabriel, T. Kotter, T. Meini, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. KNIME: The Konstanz information miner. *Studies in Classification, Data Analysis, and Knowledge Organization*, 5:319–326, 2008.

[4] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive Online Analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[5] B. Bischl. *mlr: Machine Learning in R.*, 2013. R package version 1.1-18.

[6] H. Blockeel and J. Vanschoren. Experiment databases: Towards an improved experimental methodology in machine learning. *Lecture Notes in Computer Science*, 4702:6–17, 2007.

[7] T. A. Boroson and T. R. Lauer. A candidate subparsec supermassive binary black hole system. *Nature*, 458(7234):53–55, 2009.

[8] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. Metalearning: Applications to data mining. *Springer*, 2009.

[9] C. Cardamone, K. Schawinski, M. Sarzi, S. P. Bamford, N. Bennert, C. Urry, C. Lintott, W. C. Keel, J. Parejko, R. C. Nichol, et al. Galaxy zoo green peas: discovery of a class of compact extremely star-forming galaxies. *Monthly Notices of the Royal Astronomical Society*, 399(3):1191–1205, 2009.

[10] J. Carpenter. May the best analyst win. *Science*, 331(6018):698–699, 2011.

[11] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[12] K. A. Frazer, D. G. Ballinger, D. R. Cox, D. A. Hinds, L. L. Stuve, R. A. Gibbs, J. W. Belmont, A. Boudreau, P. Hardenbol, S. M. Leal, et al. A second generation human haplotype map of over 3.1 million SNPs. *Nature*, 449(7164):851–861, 2007.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[14] D. Hand. Classifier technology and the illusion of progress. *Statistical Science*, 21(1):1–14, 2006.

[15] M. J. Hawrylycz, E. S. Lein, A. L. Guillozet-Bongaarts, E. H. Shen, L. Ng, J. A. Miller, L. N. van de Lagemaat, K. A. Smith, A. Ebbert, Z. L. Riley, et al. An anatomically comprehensive atlas of the adult human brain transcriptome. *Nature*, 489(7416):391–399, 2012.

[16] H. Hirsh. Data mining research: Current status and future opportunities. *Statistical Analysis and Data Mining*, 1(2):104–107, 2008.

[17] V. Hoste and W. Daelemans. Comparing learning approaches to coreference resolution. There is more to it than bias. In *Proceedings of the ICML'05 Workshop on Meta-learning*, pages 20–27, 2005.

[18] A. R. Isern. *The ocean observatories initiative: Wiring the ocean for interactive scientific discovery.* IEEE, 2006.

[19] T. Kealey. *Sex, science and profits*. Random House, 2010.

[20] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[21] E. S. Lein, M. J. Hawrylycz, N. Ao, M. Ayres, A. Bensinger, A. Bernard, A. F. Boe, M. S. Boguski, K. S. Brockway, E. J. Byrnes, et al. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, 445(7124):168–176, 2007.

[22] R. Leite, P. Brazdil, and J. Vanschoren. Selecting classification algorithms with active testing. In *Machine Learning and Data Mining in Pattern Recognition*, pages 117–131. Springer, 2012.

[23] C. J. Lintott, K. Schawinski, W. Keel, H. Van Arkel, N. Bennert, E. Edmondson, D. Thomas, D. J. Smith, P. D. Herbert, M. J. Jarvis, et al. Galaxy Zoo:Hanny's Voorwerp, a quasar light echo? *Monthly Notices of the Royal Astronomical Society*, 399(1):129–140, 2009.

[24] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, et al. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.

[25] I. Manolescu, L. Afanasiev, A. Arion, J. Dittrich, S. Manegold, N. Polyzotis, K. Schnaitter, P. Senellart, and S. Zoupanos. The repeatability experiment of SIGMOD 2008. *ACM SIGMOD Record*, 37(1):39–45, 2008.

[26] K. L. Masters, M. Mosleh, A. K. Romer, R. C. Nichol, S. P. Bamford, K. Schawinski, C. J. Lintott, D. Andreescu, H. C. Campbell, B. Crowcroft, et al. Galaxy Zoo: passive red spirals. *Monthly Notices of the Royal Astronomical Society*, 405(2):783–799, 2010.

[27] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994.

[28] M. Nielsen. The future of science: Building a better collective memory. *APS Physics*, 17(10), 2008.

[29] M. Nielsen. *Reinventing discovery: the new era of networked science*. Princeton University Press, 2012.

[30] E. Ostrom. Collective action and the evolution of social norms. *The Journal of Economic Perspectives*, pages 137–158, 2000.

[31] H. Parkinson, M. Kapushesky, M. Shojatalab, N. Abeygunawardena, R. Coulson, A. Farne, E. Holloway, N. Kolesnykov, P. Lilja, M. Lukk, et al. ArrayExpress: A public database of microarray experiments and gene expression profiles. *Nucleic acids research*, 35(suppl 1):D747–D750, 2007.

[32] T. Pedersen. Empiricism is not a matter of faith. *Computational Linguistics*, 34:465–470, 2008.

[33] C. Perlich, F. Provost, and J. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research*, 4:211–255, 2003.

[34] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. *Proceedings of the International Conference on Machine Learning (ICML)*, 951(2000):743–750, 2000.

[35] J. Priem, P. Groth, and D. Taraborelli. The Altmetrics Collection. *PLoS ONE*, 11(7):e48753, 2012.

[36] M. J. Raddick, G. Bracey, P. L. Gay, C. J. Lintott, P. Murray, K. Schawinski, A. S. Szalay, and J. Vandenberg. Galaxy zoo: Exploring the motivations of citizen science volunteers. *Astronomy Education Review*, 9(1):010103, 2010.

[37] M. E. Schwamb, C. J. Lintott, D. A. Fischer, M. J. Giguere, S. Lynn, A. M. Smith, J. M. Brewer, M. Parrish, K. Schawinski, and R. J. Simpson. Planet hunters: Assessing the kepler inventory of short-period planets. *The Astrophysical Journal*, 754(2):129, 2012.

[38] S. Sonnenburg, M. Braun, C. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K. Muller, F. Pereira, C. Rasmussen, G. Ratsch, B. Scholkopf, A. Smola, P. Vincent, J. Weston, and R. Williamson. The need for open source software in machine learning. *Journal of Machine Learning Research*, 8:2443–2466, 2007.

[39] A. S. Szalay, J. Gray, A. R. Thakar, P. Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. vandenBerg. The sdss skyserver: public access to the sloan digital sky server data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 570–581. ACM, 2002.

[40] L. Torgo. *Data Mining with R: Learning with Case Studies*. Chapman & Hall/CRC, 1st edition, 2010.

[41] J. N. van Rijn, B. Bischl, L. Torgo, B. Gao, V. Umaashankar, S. Fischer, P. Winter, B. Wiswedel, M. R. Berthold, and J. Vanschoren. OpenML: A collaborative science platform. In *Proceedings of ECML-PKDD 2013*, pages 645–649, 2013.

[42] J. N. van Rijn, V. Umaashankar, S. Fischer, B. Bischl, L. Torgo, B. Gao, P. Winter, B. Wiswedel, M. R. Berthold, and J. Vanschoren. A RapidMiner extension for open machine learning. In *RapidMiner Community Meeting and Conference 2013*, pages 59–70, 2013.

[43] J. Vanschoren, H. Blockeel, B. Pfahringer, and G. Holmes. Experiment databases. A new way to share, organize and learn from experiments. *Machine Learning*, 87(2):127–158, 2012.

[44] M. Wojnarski, S. Stawicki, and P. Wojnarowski. Tunedit.org: System for automated evaluation of algorithms in repeatable experiments. *Lecture Notes in Computer Science*, 6086:20–29, 2010.

join today!

SIGKDD & ACM

www.acm.org/sigkdd www.acm.org

The **ACM Special Interest Group on Knowledge Discovery in Data** (SIGKDD)'s primary mission is to provide the premier forum for advancement, education, and adoption of the "science" of knowledge discovery and data mining from all types of data stored in computers and networks of computers. SIGKDD promotes basic research and development in KDD, adoption of "standards" in the market in terms of terminology, evaluation, methodology and interdisciplinary education among KDD researchers, practitioners, and users. SIGKDD sponsors an annual international conference and publishes the bi-annual newsletter *SIGKDD Explorations*. Members receive free access to live SIGKDD webcasts.

The **Association for Computing Machinery** (ACM) is an educational and scientific computing society which works to advance computing as a science and a profession. Benefits include subscriptions to *Communications of the ACM*, *MemberNet*, *TechNews* and *CareerNews*, full and unlimited access to online courses and books, discounts on conferences and the option to subscribe to the ACM Digital Library.

❑ SIGKDD (ACM Member)	$ 25
❑ SIGKDD (ACM Student Member & Non-ACM Student Member)	$ 15
❑ SIGKDD (Non-ACM Member)	$ 25
❑ ACM Professional Membership ($99) & SIGKDD ($25)	$124
❑ ACM Professional Membership ($99) & SIGKDD ($25) & ACM Digital Library ($99)	$223
❑ ACM Student Membership ($19) & SIGKDD ($15)	$ 34
❑ *SIGKDD Explorations* only	$ 30
❑ Expedited Air for *SIGKDD Explorations* (outside N. America)	$ 8

payment information

Name ______________________________

ACM Member # ______________________________

Mailing Address ______________________________

____

City/State/Province ______________________________

ZIP/Postal Code/Country ______________________________

Email ______________________________

Mobile Phone ______________________________

Fax ______________________________

Credit Card Type: ❑ AMEX ❑ VISA ❑ MC

Credit Card # ______________________________

Exp. Date ______________________________

Signature ______________________________

Make check or money order payable to ACM, Inc

ACM accepts U.S. dollars or equivalent in foreign currency. Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Mailing List Restriction
ACM occasionally makes its mailing list available to computer-related organizations, educational institutions and sister societies. All email addresses remain strictly confidential. Check one of the following if you wish to restrict the use of your name:

❑ ACM announcements only
❑ ACM and other sister society announcements
❑ ACM subscription and renewal notices only

Questions? Contact:
ACM Headquarters
2 Penn Plaza, Suite 701
New York, NY 10121-0701
voice: 212-626-0500
fax: 212-944-1318
email: acmhelp@acm.org

Remit to:
ACM
General Post Office
P.O. Box 30777
New York, NY 10087-0777

SIGAPP

www.acm.org/joinsigs

Association for Computing Machinery

Advancing Computing as a Science & Profession

Association for Computing Machinery
2 Penn Plaza, Suite 701
New York, NY 10121-0701