Turk J Elec Eng & Comp Sci, Vol.20, No.Sup.1, 2012, © TÜBİTAK
doi:10.3906/elk-1102-1015

Abstract feature extraction for text classification

Göksel BİRİCİK*, **Banu DİRİ**, **Ahmet Coşkun SÖNMEZ**
Department of Computer Engineering, Yıldız Technical University,
Esenler, İstanbul-TURKEY
e-mails: {goksel,banu,acsonmez}@ce.yildiz.edu.tr

Received: 03.02.2011

Abstract

Feature selection and extraction are frequently used solutions to overcome the curse of dimensionality in text classification problems. We introduce an extraction method that summarizes the features of the document samples, where the new features aggregate information about how much evidence there is in a document, for each class. We project the high dimensional features of documents onto a new feature space having dimensions equal to the number of classes in order to form the abstract features. We test our method on 7 different text classification algorithms, with different classifier design approaches. We examine performances of the classifiers applied on standard text categorization test collections and show the enhancements achieved by applying our extraction method. We compare the classification performance results of our method with popular and well-known feature selection and feature extraction schemes. Results show that our summarizing abstract feature extraction method encouragingly enhances classification performances on most of the classifiers when compared with other methods.

Key Words: *Dimensionality reduction, feature extraction, preprocessing for classification, probabilistic abstract features*

1. Introduction

Assigning similar items into given categories is known as classification. For many years, people have been designing several classification or categorization systems for different disciplines including library sciences, biology, medical sciences, and artificial intelligence. Universal schemes covering all subjects like Dewey, Library of Congress, and Bliss are used in library classification [1]. Taxonomies such as Linnaean taxonomy perform biological classification. The ICD9-CM, ICF, and ICHI are examples of medical classifications. Statistical classification methods like K-nearest neighbors, naive Bayes, decision trees, and support vector machines are used in artificial intelligence and pattern recognition fields. Applications of classification and categorization in pattern recognition include speech and image recognition, document classification, personal identification, and many other tasks.

A sample subject of classification is represented by a set of features known as the feature vector. Depending on the type of samples and the field of application, the features might be numerical, nominal,

*Corresponding author: Department of Computer Engineering, Yıldız Technical University, Esenler, İstanbul-TURKEY or string. For instance, if we represent images, the feature vector consists of pixel values on the spatial domain. DNA or protein sequences form the feature vector for bioinformatics. Term occurrence frequencies can be used to represent textual data. If we have a dataset of time series, continuous values are the features for forecasting or regression.

Most of the research and work areas require a vast number of features to describe the data in practice. This requirement increases the cost of computation and decreases performance. One typical example is text classification, defined as the grouping of documents into a fixed number of predefined categories [2]. The information retrieval vector space is frequently used in text classification [3]. In vector space, we represent the documents with terms, which is also known as the bag-of-words model. The nature of the bag-of-words approach causes a very high dimensional and sparse feature space. As the dimensionality increases, the data become sparser. It is hard to build an efficient model for text classification in this high dimensional feature space. Due to this problem, dimension reduction has become one of the key problems of textual information processing and retrieval [4].

Dimension reduction is beneficial as it can eliminate irrelevant features or the curse of dimensionality. There are 2 approaches for reducing dimensions of the feature space. The first approach, feature selection, selects a subset of the original features as the new features, depending on a selection criterion. The second approach, feature extraction, reduces the dimension by creating new features by combining or projecting the original features. In this paper, we propose a supervised feature extraction method, which produces the extracted features by combining the effects of the input features over classes.

The paper begins with an introduction to dimension reduction and a quick review of the most widely known and used dimension reduction methods. After that, we introduce our feature extraction method, which summarizes the features of the document samples, where the new features aggregate information about how much evidence there is in a document, for each class. We test our method using standard text collections, using 7 different classification algorithms that belong to various design approaches. We examine the performances of the classifiers on the selected datasets and show the enhancements achieved by applying our extraction method, in comparison with the widely used feature selection and feature extraction methods. The paper also discusses how much evidence for classes is in the training samples by visualizing the abstract features derived from the evaluation datasets.

2. Previous work: dimensionality reduction techniques

The dimension of the data is defined as the number of variables that are measured on each observation in the statistics. We can give the same definition as the number of features that the samples of a dataset contain. Assume that we have an m-dimensional random variable $x = (x_1, ..., x_m)$. The purpose of dimension reduction is to find a representation for the variable with reduced dimensions [5], $r = (r_1, ..., r_k)$ with $k \leq m$.

We can follow 2 major ways to reduce dimensions of the feature vector. The first solution is feature selection, which derives a new subset of the original feature set. The second way to reduce dimensions is feature extraction, in which a new feature set with smaller dimensions is formed in a new feature space. Both approaches may be linear or nonlinear, depending on the linear separability of the classes.

Feature selection algorithms evaluate the input features using different techniques to output a smaller subset. Since the number of the selected features is smaller than the number of the originals, feature selection results in a lower dimensional feature space. The selection procedure is based on either the evaluation of features on a specific classifier to find the best subset [6], or the ranking of features by a metric and elimination of the ones that are below the threshold value [7]. Feature selection methods depending on the former approach are known as wrapper methods, and methods depending on the latter approach are called filter methods. Using fewer but more distinctive features reduces the cost of pattern recognition algorithms' computing power requirements and enhances the results [8].

Examples of linear feature selection methods are document frequency, chi-squared statistic, information gain, mutual information, and correlation coefficient [9]. We already know that the information gain, mutual information, and correlation coefficient methods share the same underlying entropic idea and select features via scoring. Nonlinear feature selection methods like relief and nonlinear kernel multiplicative updates are not used as much as the linear methods, because they are often complex to implement and/or computationally expensive [10].

Feature extraction algorithms map the multidimensional feature space to a lower dimensional space. This is achieved by combining terms to form a new description for the data with sufficient accuracy [11]. Since the projected features are transformed into a new space, they no longer resemble the original feature set, but extract relevant information from the input set. It is expected that the features would carry sufficient information from the input data to perform machine learning and pattern recognition tasks accurately, e.g., text classification. Mapping to a smaller space simplifies the amount of resources required to describe a large set of data [12], especially one having numerous features. Making use of feature extraction in vector space models is quite reasonable because it has a high dimensional and sparse, redundant structure, which requires a large amount of computation power.

The most widely known linear feature extraction methods are principal component analysis (PCA) and, especially for textual data, latent semantic analysis (LSA). There are many other methods discussed, including multidimensional scaling (MDS), learning vector quantization (LVQ), and linear discriminant analysis (LDA). Local linear embedding (LLE), self-organizing maps (SOM), and isometric feature mapping (ISOMAP) are examples of nonlinear feature extraction methods, as well [13].

Aside from the ones we named above, there are many types of feature selection and feature extraction methods implemented in the literature. In this section we only introduce the most commonly used and widely known methods, which we also choose to compare with our abstract feature extractor. We choose the chi-squared and correlation coefficient methods as the feature selection methods, because these methods produce better feature subsets than document frequency [14]. Information gain and mutual information are excluded since they share the same underlying entropic idea as the correlation coefficient method. We choose PCA, LSA, and LDA as the feature extraction methods, because PCA is known as the main feature extraction method and LSA is frequently used in text mining tasks. LDA is taken into account for comparison, as it is a supervised method like the proposed abstract feature extractor. The other mentioned methods are excluded, as they are used in different application fields instead of text classification.

2.1. Chi-squared feature selection

The chi-squared is a popular feature selection method that evaluates features individually by computing chi-squared statistics with respect to the classes [15]. This means that the chi-squared score for a term in a class measures the dependency between that term and that class. If the term is independent from the class, then its score is equal to 0.

A term with a higher chi-squared score is more informative. For a dataset consisting of N samples, the chi-squared score χ^2 for a variable v in a class c_i is defined in Eq. (1) [16]. We give dependency tuples in Table 1.

$$\chi^2(t, c_i) = \frac{N\left[P(t,c_i)P(\bar{t},\bar{c}_i) - P(t,\bar{c}_i)P(\bar{t},c_i)\right]^2}{P(t)P(\bar{t})P(c_i)P(\bar{c}_i)} \tag{1}$$

2.2. Correlation coefficient feature selection

The correlation coefficient is in fact a variant of chi-squared, where $cc^2 = \chi^2$. This method evaluates the worthiness of a subset of features by considering the individual predictive ability of each term along with the degree of redundancy between them [17]. The preferred subset of features is the one having high correlation within the class and low correlation between different classes.

For a dataset consisting of N samples, the correlation coefficient cc for a variable v in a class c_i is defined in Eq. (2) [16]. We give dependency tuples in Table 1.

$$cc(t, c_i) = \frac{\sqrt{N}\left[P(t,c_i)P(\bar{t},\bar{c}_i) - P(t,\bar{c}_i)P(\bar{t},c_i)\right]}{\sqrt{P(t)P(\bar{t})P(c_i)P(\bar{c}_i)}} \tag{2}$$

Table 1. Dependency tuples for the discussed feature selection methods.

	Membership in c_i	Nonmembership in c_i
Presence of t	(t, c_i)	$(t, \bar{c}_i)$
Absence of t	$(\bar{t}, c_i)$	$(\bar{t}, \bar{c}_i)$

2.3. Singular value decomposition-based methods

Before introducing PCA and LSA, we briefly describe the singular value decomposition (SVD) process as it is used in both methods.

Let A be an $m \times n$ real matrix, where $m \geq n$. We can rewrite A as the product of an $m \times n$ column-orthogonal matrix $U(U^TU = I)$, an $n\times\ n$ diagonal matrix Λ with positive or zero elements (the singular values) in descending order ($\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$), and the transpose of an $n \times n$ orthogonal matrix $V(V^TV = I)$, as in Eq. (3). This decomposition is referred to as SVD.

$$A = U\Lambda V^T \tag{3}$$

We can prove Eq. (3) by defining U, Λ, and V. If A is an $m \times n$ matrix, then A^TA is an $n \times n$ symmetrical matrix. This means that we can identify the eigenvectors and eigenvalues for A^TA as the columns of V and the squared diagonal elements of Λ (which are proven to be nonnegative as they are squared), respectively. Let λ be an eigenvalue of A^TA and x be the corresponding eigenvector. Defining Eq. (4) gives us Eq. (5).

$$\|Ax\|^2 = x^TA^TAx = \lambda x^Tx = \lambda\|x\|^2 \tag{4}$$

$$\lambda = \frac{\|Ax\|^2}{\|x\|^2} \geq 0 \tag{5}$$

If we order the eigenvalues of A^TA and define the matrix composed of the corresponding eigenvectors V, we can define the singular values with Eq. (6).

$$\sigma_j = \sqrt{\lambda_j}, j = 1, ..., n \tag{6}$$

If the rank of A is r, then the rank of A^TA is also r. Because A^TA is symmetrical, its rank equals the number of positive nonzero eigenvalues. This proves that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ and $\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_n = 0$. Assuming $V_1 = (v_1, v_2, \ldots, v_r)$, $V_2 = (v_{r+1}, v_{r+2}, \ldots, v_n)$, and Λ_1 as an $r \times r$ diagonal matrix, we can define Λ and A with Eqs. (7) and (8).

$$\Lambda = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{7}$$

$$\begin{gathered} I = VV^T = V_1V_1^T + V_2V_2^T \\ A = AI = AV_1V_1^T + AV_2V_2^T = AV_1V_1^T \end{gathered} \tag{8}$$

Now we will show that $AV = U\Lambda$. For the first r columns, we can write $Av_j = \sigma_j u_j$ and define $U_1 = (u_1, u_2, \ldots, u_r)$, $AV_1 = U_1\Lambda_1$. The rest of the m-r dimensional orthonormal column vectors can be defined with $U_2 = (u_{r+1}, u_{r+2}, \ldots, u_m)$. As $U = (U_1 \;\; U_2)$, we can rewrite Eq. (3) with Eq. (9). Solving Eq. (9) proves that $A = U\Lambda V^T$, as given in Eq. (10).

$$U\Lambda V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{9}$$

$$\begin{aligned} U\Lambda V^T &= U_1\Lambda_1V_1^T \\ &= AV_1V_1^T \\ &= A \end{aligned} \tag{10}$$

We state that both PCA and LSA depend on SVD. The eigen-decomposed input matrix makes the difference between these methods. Using SVD, the covariance matrix is decomposed in PCA, while the term-document matrix is decomposed in LSA. In fact, PCA and LSA are equivalent if the term-document matrix is centered.

2.3.1. Principal component analysis

PCA transforms correlated variables into a smaller number of correlated variables, which are known as the principal components. Invented by Pearson in 1901, it is generally used for exploratory data analysis [18]. PCA is used for feature extraction by retaining the characteristics of the dataset that contribute most to its variance, by keeping lower order principals, which tend to have the most important aspects of the data. This is accomplished by a projection into a new hyperplane using eigenvalues and eigenvectors. The first principal component is the linear combination of the features with the largest variance or, in other words, the eigenvector with the largest eigenvalue. The second principal component has a smaller variance and is orthogonal to the first one. There are as many eigenvectors as the number of the original features, which are sorted with the highest eigenvalue first and the lowest eigenvalue last. Usually, 95% variance coverage is used to reduce dimensions while keeping the most important characteristics of the dataset.

Finding the principal components depends on the SVD of the covariance matrix Σ. We can write the covariance matrix Σ as in Eq. (11), where Λ is the diagonal matrix of the ordered eigenvalues and U is a $p \times p$ orthogonal matrix of the eigenvectors. The principal components obtained by SVD are the p rows of the $p \times n$ matrix S, as shown in Eq. (12). The appropriate number of principal components can be selected to describe the overall variation with desired accuracy.

$$\Sigma = U\Lambda U^T \tag{11}$$

$$S = U^T X \tag{12}$$

PCA is a popular technique in pattern recognition, but its applications are not very common because it is not optimized for class separability [19]. It is widely used in image processing disciplines [8].

2.3.2. Latent semantic analysis

Rather than dimension reduction, LSA is known as a technique in natural language processing. Patented in 1988, LSA analyzes relationships between a document set and the terms they contain. LSA produces a set of concepts, which is smaller in size than the original set, related to documents and terms [20]. LSA uses SVD to find the relationships between documents. Given a term-document matrix X, the SVD breaks down X into a set of 3 smaller components, as:

$$X = U\Sigma V^T. \tag{13}$$

If we represent the correlations between terms over documents with XX^T, and the correlations between documents over terms with X^TX, we can also show these matrices with Eqs. (14) and (15).

$$XX^T = U\Sigma\Sigma^T U^T \tag{14}$$

$$X^TX = V\Sigma^T\Sigma V^T \tag{15}$$

When we select k singular values from Σ and the corresponding vectors from U and V matrices, we get the rank k approximation for X with a minimal error. This approximation can be seen as a dimension reduction. If we recombine Σ, U, and V and form$\hat{X}$, we can use it again as a lookup grid. The matrix we get back is an approximation of the original one, which we can show with Eq. (16). The features extracted with LSA lie in the orthogonal space.

$$\hat{X}_k = U_k \Sigma_k V_k^T \tag{16}$$

LSA is mostly used for page retrieval systems and document clustering purposes. It is also used for document classification or information filtering. Many algorithms utilize LSA in order to improve performance by working in a less complex hyperspace. LSA requires relatively high computational power and memory because the method utilizes complex matrix calculations using SVD, especially when working on datasets having thousands of documents. There is an algorithm for fast SVD on large matrices using low memory [21]. These improvements make the process easier and ensure extensive usage.

2.4. Linear discriminant analysis

LDA reveals a linear combination of features to model the difference between the classes for separation. The resulting combination can be used for dimension reduction. LDA tries to compute a transformation that maximizes the ratio of the between-class variance to the within-class variance. The class separation in direction $w$can be calculated with Eq. (17) using the between-class scatter matrix Σ_B, defined in Eq. (18), and the within-class scatter matrix Σ_W, defined in Eq. (19) [22]. For Eqs. (18) and (19), μ_c is the mean of class c and μ is the mean of class means.

$$S = \frac{w^T\Sigma_B w}{w^T\Sigma_W w} \tag{17}$$

$$\Sigma_B = \sum_c (\mu_c - \mu)(\mu_c - \mu)^T \tag{18}$$

$$\Sigma_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{19}$$

The transformation computed by LDA maximizes Eq. (17). If w is an eigenvector of $\Sigma_W^{-1}\Sigma_B$, then the class separations are equal to the eigenvalues. We can give the linear transformation by a matrix U, where the columns consist of the eigenvectors of $\Sigma_W^{-1}\Sigma_B$, as in Eq. (20). The eigenvectors obtained by solving Eq. (21) can be used for dimension reduction as they identify a vector subspace that contains the variability between features.

$$\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \dots \\ u_K^T \end{bmatrix} (x - \mu) = U^T(x - \mu) \tag{20}$$

$$\Sigma_B u_k = \lambda_k \Sigma_w u_k \tag{21}$$

3. Abstract feature extraction algorithm

The method we provide, the abstract feature extractor (AFE), is a supervised feature extraction algorithm that produces the extracted features by combining the effects of the input features over classes. Thus, the number of resulting features is equal to the number of classes. The AFE differs from most of the feature extraction methods as it does not use SVD on the feature vectors. Input features are projected to a suppositious feature space using the probabilistic distribution of the features over classes. We project the probabilities of the features to classes and sum up these probabilities to get the impact of each feature to each class.

Assume we have a total of I features in J samples within K classes. Let $n_{i,j}$ be the number of occurrences of feature f_i in sample s_j and let J_i be the total number of samples that contain f_i in the entire dataset. Since we focus on text classification, our samples are documents, and features are the terms in documents. When documents and terms are involved, $n_{i,j}$ is the term frequency of f_i in s_j. Here we list the steps of the AFE.

1. Calculate $nc_{i,k}$, the total number of occurrences of f_i in samples that belong to class c_k, with:

$$nc_{i,k} = \sum_j n_{i,j}, \qquad s_j \in c_k \ . \tag{22}$$

2. Calculate $w_{i,k}$[1], the weight of f_i that affects class c_k, with:

$$w_{i,k} = \log(nc_{i,k} + 1) \times \log\left(\frac{J}{J_i}\right) . \tag{23}$$

3. *Repeat for all of the samples:*

[1]This weighting is similar to term frequency-inverse document frequency; the difference is in the frequency calculations of the features. We calculate the feature frequencies not for each sample in the dataset individually, but for all of the samples in c_k that contain f_i. This can be seen as calculating in-class frequencies of the feature set. The results are the weights of the input features. These weights indicate how much a feature affects a class.

Calculate $Y_{j,k}$, the total effect of features in sample s_j over class c_k, with:

$$Y_{j,k} = \sum_i w_{i,k}, \qquad f_i \in s_j \ . \tag{24}$$

4. Normalize the reduced K features $AF_{j,k}$ of s_j with:

$$AF_{j,k} = \frac{Y_{j,k}}{\sum\limits_k Y_{j,k}}. \tag{25}$$

At the end, we have K extracted features in hand for our samples. The representation is formed in a reduced matrix with J rows (one row per sample) and K columns (number of extracted features equal to the number of classes). That is, features are projected onto a new feature space with dimensions equal to the number of classes.

It is possible to observe the mapping of the AFE on a document-word matrix of a given dataset. Assume we have J documents in K classes and a total of I words in our training set. We define the $J \times I$ document-word matrix X and the $J \times K$ document-class matrix Y weighted using $w_{i,k}$ in Eq. (23). The AFE projects features using $X^T Y$ with column normalization, which represents the word-class distribution matrix. The bag-of-words representation of the training document matrix X and each test document v could then be projected onto the new space as $XX^T Y$ and $vX^T Y$, respectively, again with column normalization. Since the overall operation is a linear mapping between finite-dimensional vector spaces, the normalization process breaks linearity as it depends on the inputs, X or v. Thus, original features cannot be linearly reconstructed from extracted abstract features.

The main difference from other popular feature extraction methods is that the AFE requires a labeled dataset to form the resulting projection space. Instead of utilizing a ranking strategy to choose the most distinguishing extracted features, the method depends on the number of classes because the main idea is to find the probabilistic distribution of input features over the classes. Once the distribution is calculated using Eqs. (22) and (23), we can easily produce extracted features for the samples in the dataset using Eqs. (24) and (25). The extracted K features AF_k for a sample s_j can be seen as the membership probabilities of s_j to K classes.

3.1. Discussion on term weighting

Assigning weights to terms is the key point in information retrieval and text classification [23]. Therefore, many weighting schemes are presented in the literature. Term weighting can be as simple as binary representation or as detailed as a blend of term and dataset existence probabilities derived from complex information theoretic underlying concepts. New approaches like term frequency-relevance frequency (TFRF) [24] show that it is better to award the terms with higher frequencies in the positive category and penalize the terms with higher frequencies in the negative category. More or less, term frequency-inverse document frequency (TFIDF) is the most widely known and used weighting method, and it is still even comparable with novel methods [24]. We use TFIDF to weight the terms in term-document matrices of our evaluation datasets. However, the notion of TFRF inspired us to weight the effects of terms on the classes as well.

In the AFE, we combine the in-class term frequencies given in Eq. (22) with inverse document frequencies and use this scheme to weight the effects of terms on the classes, as in Eq. (23). Using in-class term frequencies shares the idea of TFRF. A recent study on concise semantic analysis (CSA) [22] modeled the term vectors in a similar way to the AFE, but term and document weighting factors differed. Moreover, CSA creates features as much as concepts, which have to be determined before the process. The number of extracted features with the AFE is as much as the number of classes, which is an already known number. Even if the number of concepts would be selected equal to the number of classes, the resulting features of CSA and the AFE are different since the weightings are different and the AFE executes an additional mapping.

4. Materials and methods

In this section we introduce our evaluation datasets and dimension reduction methods that we choose to compare with the AFE. We also introduce the selected classification algorithms and their parameters.

4.1. Selected datasets as evaluation material

We test our AFE method and compare it with other methods by examining the performances of classifiers applied on standard textual data. The first dataset is Reuters-21578[2] and the second is the reduced version of the 20 Newsgroups dataset, which is known as the 20 Newsgroups Mini[3] dataset. Both selected datasets are known as the standard test collections for text categorization. We use 2 ports of the Reuters-21578 dataset, with the details described in this section.

In the first Reuters dataset port, we choose the news that contains only one topic label and body text as our samples. In order to be as fair as possible, we choose our samples from the classes that have an approximately equal number of samples. To achieve this, we apply a filter on the number of samples each class contains, we calculate the mean and standard deviation for the distribution of samples among the classes, and then we filter this distribution with a box-plot with the center μ and boundaries $(\pm 0.2 \times \sigma)$. The classes having a number of samples in this interval are chosen for evaluation. As a result, the chosen dataset of Reuters consists of 1623 samples in 21 classes. The selected classes for classification and the number of training samples within them are listed in Table 2. We choose 10-fold cross validation for this dataset for the test results.

Table 2. Distribution of the samples among the selected 21 classes of the Reuters dataset.

Classes	Number of samples	Classes	Number of samples	Classes	Number of samples
Alum	48	Gnp	115	Nat-gas	48
Bop	46	Gold	111	Oilseed	78
Cocoa	58	Ipi	45	Reserves	50
Coffee	124	Iron-steel	51	Rubber	40
Copper	57	Jobs	47	Ship	194
Cpi	75	Livestock	55	Sugar	144
Dlr	34	Money-supply	110	oil	93

The second Reuters dataset port is the standard ModApte-10 split. Instead of cross validation, we use the standard train/test splits of Reuters ModApte-10. Reuters is known as an extremely skewed dataset. This port of the Reuters dataset is chosen to prove that the AFE works well both on homogeneous and heterogeneous data.

[2]Dataset is retrieved from http://www.daviddlewis.com/resources/testcollections/reuters21578
[3]Dataset is retrieved from http://kdd.ics.uci.edu/databases/20newsgroups

The original 20 Newsgroups dataset consists of 20,000 messages taken from 20 different Usenet newsgroups. The characteristic of the dataset is known as some of the newsgroups are highly related, while some are irrelevant, generally bunched in 6 clusters. Names and clusters of the 20 Newsgroups dataset are shown in Figure 1. The original dataset contains approximately 1000 messages per class. We use the reduced version of the dataset that contains 100 messages in each class with a total of 2000 samples, which is known as 20 Newsgroups Mini, with no prior filtering process.

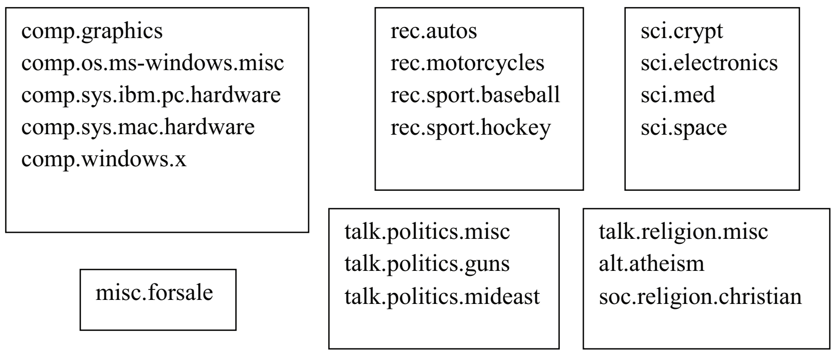

Figure 1. Distribution classes of the 20 Newsgroups dataset and clusters according to their subject relations.

We use the stemmer of Porter [25] to stem the terms of the samples for both datasets. We remove stop words, numbers, and all punctuation marks after stemming. When the preprocessing is done, the Reuters dataset has a total of 8120 terms in 1623 documents and the 20 Newsgroups dataset contains 25,204 terms in 2000 documents. This means that we represent the Reuters dataset as a term-document matrix with 1623 rows and 8120 columns. The term-document matrix of the 20 Newsgroups dataset is much larger, with 2000 rows and 25,204 columns. The ModApte-10 port of the Reuters dataset contains 16,436 terms and 9034 documents when the train and test splits are combined.

We use the popular and well-known TFIDF scheme for weighting the terms in our term-document matrices, which is calculated with Eq. (26), where $n_{i,j}$ is the number of occurrences of term t_i in document d_j, $|D|$ is the total number of documents, and $|\{d_j : t_i \in d_j\}|$ is the number of documents where term t_i appears.

$$tfidf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \times \log \frac{|D|}{|\{d_j : t_i \in d_j\}|} \tag{26}$$

4.2. Methods for comparison

We pick 5 popular and widely used dimension reduction schemes to compare with our feature extraction method. As Jensen [14] points out, the chi-squared and correlation coefficient methods produce better feature subsets than the document frequency method. Thus, we pick the correlation coefficient (as an entropy-based method) and chi-squared methods as feature selectors. We choose PCA because it is known as the main feature extraction method. The second extraction method we utilize for comparison is LSA, which is popular in text mining tasks. The last feature extraction method compared is LDA, which is a supervised method like the AFE. We apply these methods and the AFE on the chosen datasets to compare their effects on classification performances. The number of features obtained by applying the selected dimension reduction techniques is given in Table 3. We see that the number of reduced features is different for each method and dataset. These numbers are obtained by running the selected dimension reduction methods with their default settings and parameters. We also include tests by setting the number of reduced features equal to the AFE for other methods in order to see if the number of dimensions affects performance.

Table 3. Number of reduced features obtained with the selected methods.

	Reuters	20 Newsgroups	ModApte-10
No reduction	8121	25,205	16,436
AFE	22	20	10
Chi-squared	327	327	2020
Correlation coefficient	39	70	39
PCA	287	1423	1887
LSA	1146	1057	1173
LDA	21	19	9

We choose 7 classification algorithms of different design approaches to compare the effects of the dimension reduction techniques on classification performances. We list the selected algorithms here:

- Naive Bayes as a simple probabilistic classifier, which is based on applying Bayes' theorem with strong independence assumptions [26].
- C4.5 decision tree algorithm [27] as a basic tree based classifier. We choose the confidence factor as 0.25 and the minimum number of instances per leaf as 2.
- RIPPER [28] as a rule-based learner. The minimum total weight of the instances in a rule is set to 2.0. We choose 3-fold for pruning and 2 optimization runs.
- Ten-nearest neighbor algorithm to test instance-based classifiers. We use the *1/distance* distance weighting scheme. We also run one-nearest neighbor with default Euclidean distance calculation and no weighting in order to evaluate the nearest neighbor algorithm with its standard settings.
- A 10-tree random forest to construct a collection of decision trees with controlled variations [29]. We set the tree depth limit as infinite.
- Support vector machine (SVM) [30] as a kernel-based learner, which is also robust to data sparsity. We choose the linear kernel $u'*v$. We set the cost parameter to 1.0 and the termination tolerance epsilon to 0.001.
- LINEAR [31] as a linear classifier that is known to be accurate, especially on large and sparse datasets. We set the cost parameter to 1.0 and the termination tolerance epsilon to 0.01.

5. Experimental results

We evaluate the efficiency of the AFE among the other dimension reduction schemes described in Section 2 by using 7 different classification algorithms on the selected datasets, which we introduce in Section 4.2. We utilize Weka [32] as our test environment.

For independent random splitting of training and test sets, a 10-fold cross-validation method is used on the Reuters and 20 Newsgroups datasets. We quantify the results as the average precision with Eq. (27), recall with Eq. (28), and F_1 measure with Eq. (29), obtained from the 10 runs on each fold. For Eqs. (27), (28), and (29), *TP* is the number of true positives, *FP* is the number of false positives, and *FN* is the number of false negatives. For the ModApte-10 split of the Reuters dataset, we use the standard train and test splits instead of cross-validation for fair comparison.

$$precision = \frac{TP}{TP + FP} \tag{27}$$

$$recall = \frac{TP}{TP + FN} \tag{28}$$

$$F_1 = \frac{2 \times precision \times recall}{(precision + recall)} \tag{29}$$

Table 4. Performance comparisons of the dimension reduction schemes applied before classification of the Reuters dataset.

Reuters dataset		Without dimension reduction	AFE	Chi-squared	Correlation coefficient	PCA	LSA	LDA
Naïve Bayes	Precision	0.738	0.932	0.821	0.726	0.564	0.656	0.723
	Recall	0.708	0.932	0.808	0.649	0.481	0.519	0.580
	F_1 measure	0.715	0.931	0.810	0.638	0.487	0.517	0.584
C4.5	Precision	0.835	0.914	0.830	0.807	0.578	0.680	0.820
	Recall	0.835	0.913	0.829	0.807	0.567	0.680	0.814
	F_1 measure	0.834	0.912	0.828	0.806	0.570	0.679	0.813
RIPPER	Precision	0.824	0.921	0.838	0.805	0.528	0.650	0.806
	Recall	0.808	0.918	0.822	0.776	0.483	0.638	0.769
	F_1 measure	0.810	0.919	0.824	0.781	0.492	0.640	0.773
1-nearest neighbor	Precision	0.770	0.966	0.826	0.838	0.767	0.845	0.835
	Recall	0.619	0.965	0.810	0.834	0.708	0.258	0.828
	F_1 measure	0.633	0.965	0.811	0.835	0.723	0.312	0.827
10-nearest neighbor	Precision	0.774	0.969	0.870	0.861	0.779	0.350	0.847
	Recall	0.506	0.969	0.762	0.844	0.687	0.088	0.837
	F_1 measure	0.481	0.969	0.789	0.847	0.692	0.046	0.836
Random forest	Precision	0.649	0.931	0.824	0.846	0.684	0.370	0.841
	Recall	0.642	0.929	0.821	0.845	0.678	0.366	0.833
	F_1 measure	0.635	0.929	0.819	0.844	0.672	0.357	0.832
SVM	Precision	0.911	0.969	0.913	0.871	0.819	0.761	0.857
	Recall	0.900	0.969	0.909	0.855	0.781	0.610	0.839
	F_1 measure	0.901	0.969	0.910	0.856	0.783	0.598	0.837
LINEAR	Precision	0.934	0.838	0.893	0.869	0.867	0.792	0.858
	Recall	0.932	0.852	0.892	0.868	0.866	0.739	0.847
	F_1 measure	0.932	0.820	0.892	0.868	0.865	0.743	0.845
Average	Precision	0.804	0.930	0.852	0.828	0.698	0.638	0.823
	Recall	0.744	0.931	0.832	0.810	0.656	0.487	0.793
	F_1 measure	0.742	0.927	0.835	0.809	0.661	0.487	0.793

5.1. Tests using default parameters

We set up the first test using the default parameters of the selected dimension reduction methods. This results in a different number of reduced features for each method, which are given in Table 3. The classification performances obtained from the tests using the Reuters, 20 Newsgroups, and ModApte-10 datasets are consecutively listed in Tables 4, 5, and 6. We see that the AFE improves the precision, recall, and F$_1$ measure results of naive Bayes, C4.5, RIPPER, 1-nearest neighbor, 10-nearest neighbor, and random forest classifiers in comparison with the other dimension reduction schemes on all of the datasets. Prior to the SVM, the AFE gives the highest precision, recall, and F$_1$ measure values among other methods on the Reuters and 20 Newsgroups datasets, but the chi-squared method and application of no reduction show better performance than the AFE on the ModApte-10 split. Prior to the LINEAR classifier, the AFE provides the highest precision, recall, and F$_1$ measure values on both the 20 Newsgroups and ModApte-10 split datasets, while it is only better than LSA on the Reuters dataset.

Table 5. Performance comparisons of the dimension reduction schemes applied before classification of the 20 Newsgroups dataset.

20 Newsgroups dataset		Without dimension reduction	AFE	Chi-squared	Correlation coefficient	PCA	LSA	LDA
Naïve Bayes	Precision	0.559	0.899	0.612	0.481	0.514	0.577	0.419
	Recall	0.521	0.898	0.605	0.446	0.470	0.504	0.298
	F$_1$ measure	0.527	0.897	0.597	0.436	0.472	0.516	0.289
C4.5	Precision	0.501	0.869	0.506	0.446	0.438	0.453	0.407
	Recall	0.484	0.869	0.498	0.438	0.432	0.444	0.353
	F$_1$ measure	0.490	0.869	0.500	0.439	0.434	0.447	0.363
RIPPER	Precision	0.504	0.878	0.515	0.471	0.405	0.413	0.511
	Recall	0.417	0.877	0.451	0.391	0.385	0.407	0.303
	F$_1$ measure	0.433	0.877	0.467	0.391	0.391	0.408	0.343
1-nearest neighbor	Precision	0.701	0.923	0.511	0.415	0.732	0.774	0.344
	Recall	0.108	0.922	0.483	0.396	0.091	0.222	0.302
	F$_1$ measure	0.108	0.922	0.489	0.402	0.081	0.278	0.311
10-nearest neighbor	Precision	0.442	0.940	0.553	0.488	0.330	0.525	0.421
	Recall	0.082	0.939	0.449	0.431	0.066	0.065	0.356
	F$_1$ measure	0.056	0.939	0.463	0.444	0.037	0.036	0.372
Random forest	Precision	0.535	0.913	0.473	0.392	0.183	0.227	0.344
	Recall	0.459	0.912	0.466	0.378	0.173	0.209	0.306
	F$_1$ measure	0.472	0.912	0.465	0.382	0.171	0.211	0.313
SVM	Precision	0.731	0.932	0.664	0.581	0.714	0.692	0.566
	Recall	0.695	0.930	0.614	0.496	0.696	0.644	0.377
	F$_1$ measure	0.705	0.930	0.631	0.521	0.701	0.659	0.409
LINEAR	Precision	0.772	0.950	0.600	0.545	0.703	0.677	0.557
	Recall	0.749	0.949	0.597	0.523	0.703	0.671	0.394
	F$_1$ measure	0.754	0.948	0.597	0.525	0.702	0.673	0.426
Average	Precision	0.593	0.851	0.540	0.472	0.516	0.532	0.449
	Recall	0.547	0.912	0.582	0.501	0.486	0.505	0.409
	F$_1$ measure	0.505	0.864	0.537	0.456	0.438	0.468	0.376

Table 6. Performance comparisons of the dimension reduction schemes applied before classification of the Reuters ModApte-10 split dataset.

ModApte-10 split dataset		Without dimension reduction	AFE	Chi-squared	Correlation coefficient	PCA	LSA	LDA
Naïve Bayes	Precision	0.860	0.918	0.891	0.867	0.671	0.541	0.797
	Recall	0.407	0.911	0.750	0.755	0.628	0.547	0.743
	F_1 measure	0.540	0.911	0.803	0.790	0.612	0.508	0.732
C4.5	Precision	0.867	0.949	0.881	0.850	0.839	0.840	0.819
	Recall	0.870	0.949	0.882	0.851	0.841	0.840	0.808
	F_1 measure	0.868	0.948	0.881	0.849	0.840	0.840	0.805
RIPPER	Precision	0.874	0.950	0.867	0.841	0.868	0.840	0.779
	Recall	0.875	0.949	0.868	0.841	0.864	0.832	0.777
	F_1 measure	0.872	0.949	0.863	0.836	0.865	0.836	0.767
1-nearest neighbor	Precision	0.754	0.957	0.764	0.854	0.730	0.754	0.787
	Recall	0.542	0.956	0.688	0.851	0.695	0.641	0.780
	F_1 measure	0.484	0.956	0.659	0.852	0.667	0.652	0.775
10-nearest neighbor	Precision	0.741	0.962	0.728	0.875	0.679	0.824	0.821
	Recall	0.468	0.962	0.483	0.876	0.538	0.522	0.810
	F_1 measure	0.359	0.961	0.369	0.875	0.460	0.516	0.807
Random forest	Precision	0.828	0.918	0.868	0.887	0.735	0.566	0.785
	Recall	0.837	0.911	0.872	0.890	0.752	0.602	0.780
	F_1 measure	0.825	0.911	0.863	0.888	0.719	0.550	0.775
SVM	Precision	0.927	0.878	0.914	0.890	0.881	0.858	0.828
	Recall	0.929	0.905	0.917	0.893	0.886	0.864	0.810
	F_1 measure	0.927	0.882	0.914	0.891	0.883	0.857	0.808
LINEAR	Precision	0.926	0.953	0.907	0.893	0.920	0.824	0.830
	Recall	0.925	0.945	0.911	0.895	0.921	0.810	0.808
	F_1 measure	0.925	0.937	0.908	0.893	0.920	0.790	0.808
Average	Precision	0.847	0.936	0.853	0.870	0.790	0.756	0.806
	Recall	0.732	0.936	0.796	0.857	0.766	0.707	0.790
	F_1 measure	0.725	0.932	0.783	0.859	0.746	0.694	0.785

For the Reuters dataset, the best precision, recall, and F_1 measure values (both 96.9%) are achieved with the AFE applied before the 10-nearest neighbor and SVM classifiers. The following highest precision is 96.6%, and the recall and F_1 measures are 96.5%, achieved with the AFE applied before the 1-nearest neighbor. For the 20 Newsgroups dataset, the best precision is 95.0%, the best recall is 94.9%, and the best F_1 measure is 94.8%, all achieved with the AFE applied before LINEAR. The following highest precision of 94.9% and recall and F_1 measures of 93.9% are achieved again with the AFE prior to the 10-nearest neighbor classifier. For the ModApte-10 split dataset, the best precision is 96.2%, the best recall is 96.2%, and the best F_1 measure is 96.1%, all achieved by applying the AFE before the 10-nearest neighbor classifier. The following highest precision of 95.7% and recall and F_1 measures of 95.6% are achieved again with the AFE prior to the 10-nearest neighbor classifier.

If we look at the average performances of the classifiers among the dimension reduction methods on the Reuters dataset, the highest average precision of 93.0%, recall of 93.1%, and F_1 measure of 92.7% are achieved with the AFE, followed by the correlation coefficient method with 85.2% precision, 83.2% recall, and 80.9% F_1 measure scores. Focusing on the 20 Newsgroups dataset, the AFE is by far the best with 84.1% average precision, 91.2% recall, and 86.4% F$_1$ measure. The second-best average F$_1$ measure is 53.7% by the chi-squared feature selection. On the ModApte-10 split, highest average precision and recall of 93.6% and F$_1$ measure of 93.2% is achieved with the AFE, followed by the 85.9% F$_1$ measure score of the correlation coefficient method.

5.2. Tests with equal number of reduced features

We set up the second test by setting the number of reduced features equal to the number of classes in the datasets for the selected dimension reduction methods. This makes a fair comparison of the AFE with other methods and tests, whether the number of reduced features affects the classifier's performances or not. We set the number of reduced features to 21 for the Reuters, 20 for 20 Newsgroups, and 10 for ModApte-10 datasets. LDA is excluded from this test since it outputs $C-1$ number of extracted features as linear discriminants for the classes.

The classification performance results of this test are listed in Tables 7-9, each for one of the 3 datasets. We see that the AFE results in the highest performances for each classifier on all of the datasets, except for

Table 7. Performance comparisons of the dimension reduction schemes, each having 21 reduced features, applied before classification of the Reuters dataset.

Reuters dataset		AFE	Chi-squared	Correlation coefficient	PCA	LSA
Naïve Bayes	Precision	0.938	0.650	0.669	0.798	0.843
	Recall	0.931	0.603	0.572	0.771	0.816
	F$_1$ measure	0.930	0.595	0.559	0.769	0.817
C4.5	Precision	0.923	0.743	0.774	0.695	0.765
	Recall	0.913	0.743	0.767	0.670	0.743
	F$_1$ measure	0.912	0.727	0.759	0.663	0.739
RIPPER	Precision	0.926	0.713	0.747	0.716	0.755
	Recall	0.918	0.717	0.720	0.643	0.712
	F$_1$ measure	0.917	0.692	0.707	0.644	0.712
1-nearest neighbor	Precision	0.966	0.735	0.776	0.818	0.876
	Recall	0.965	0.744	0.776	0.816	0.873
	F$_1$ measure	0.965	0.734	0.775	0.816	0.874
10-nearest neighbor	Precision	0.972	0.761	0.808	0.838	0.890
	Recall	0.969	0.766	0.795	0.819	0.874
	F$_1$ measure	0.968	0.748	0.789	0.816	0.874
Random forest	Precision	0.938	0.736	0.800	0.802	0.852
	Recall	0.929	0.744	0.789	0.786	0.832
	F$_1$ measure	0.927	0.727	0.783	0.781	0.829
SVM	Precision	0.972	0.753	0.797	0.825	0.879
	Recall	0.969	0.759	0.795	0.801	0.858
	F$_1$ measure	0.968	0.737	0.781	0.793	0.854
LINEAR	Precision	0.823	0.744	0.782	0.844	0.759
	Recall	0.852	0.765	0.791	0.831	0.789
	F$_1$ measure	0.818	0.740	0.774	0.826	0.754
Average	Precision	0.932	0.729	0.769	0.792	0.827
	Recall	0.931	0.730	0.751	0.767	0.812
	F$_1$ measure	0.926	0.713	0.741	0.764	0.807

PCA applied before LINEAR, which gives better results than the AFE on the Reuters dataset. On the Reuters dataset, the best precision is 97.2%, recall is 96.9%, and F$_1$ measure is 96.8%, as scored by the AFE applied before the 10-nearest neighbor classifier. The nearest F$_1$ measure of the compared methods is 87.4%, achieved by LSA with the 10-nearest neighbor classifier. For the 20 Newsgroups dataset, the best precision is 95.3%, recall is 94.8%, and F$_1$ measure is 94.7% using the AFE with the LINEAR classifier. The chi-squared and correlation coefficient methods give the worst results on this dataset with 20 selected features; their F$_1$ measures are around 30%. This shows us that selecting too few features out of the original feature set is not suitable for the 20 Newsgroups dataset. PCA and LSA score around 53% F$_1$ measures on average by extracting 20 features, which is about 20% better than the feature selection methods but about 40% worse than the AFE. The highest F$_1$ measure of the compared methods is 59.8%, as scored by LSA with the SVM. For the ModApte-10 split dataset, the best precision and recall is 96.2% and the F$_1$ measure is 96.1% by applying the AFE before the 10-nearest neighbor classifier. The nearest F$_1$ measure of the compared methods is 91.4%, achieved by PCA with the 10-nearest neighbor classifier.

Table 8. Performance comparisons of the dimension reduction schemes, each having 20 reduced features, applied before classification of the 20 Newsgroups dataset.

20 Newsgroups dataset		AFE	Chi-squared	Correlation coefficient	PCA	LSA
Naïve Bayes	Precision	0.910	0.350	0.374	0.575	0.598
	Recall	0.898	0.301	0.295	0.556	0.587
	F$_1$ measure	0.897	0.281	0.280	0.540	0.577
C4.5	Precision	0.881	0.370	0.365	0.446	0.452
	Recall	0.869	0.304	0.321	0.438	0.439
	F$_1$ measure	0.868	0.297	0.326	0.432	0.436
RIPPER	Precision	0.898	0.334	0.447	0.547	0.540
	Recall	0.877	0.251	0.289	0.421	0.412
	F$_1$ measure	0.879	0.245	0.303	0.439	0.434
1-nearest neighbor	Precision	0.923	0.394	0.322	0.498	0.532
	Recall	0.922	0.291	0.292	0.492	0.530
	F$_1$ measure	0.922	0.302	0.302	0.494	0.530
10-nearest neighbor	Precision	0.944	0.414	0.386	0.594	0.586
	Recall	0.939	0.318	0.333	0.584	0.607
	F$_1$ measure	0.939	0.317	0.341	0.564	0.571
Random forest	Precision	0.919	0.380	0.339	0.525	0.562
	Recall	0.912	0.288	0.291	0.519	0.551
	F$_1$ measure	0.912	0.291	0.300	0.510	0.546
SVM	Precision	0.936	0.379	0.503	0.621	0.622
	Recall	0.930	0.303	0.350	0.584	0.607
	F$_1$ measure	0.930	0.296	0.371	0.580	0.598
LINEAR	Precision	0.953	0.340	0.428	0.597	0.562
	Recall	0.948	0.323	0.364	0.604	0.566
	F$_1$ measure	0.947	0.299	0.359	0.582	0.525
Average	Precision	0.921	0.370	0.396	0.550	0.557
	Recall	0.912	0.297	0.317	0.525	0.537
	F$_1$ measure	0.912	0.291	0.323	0.518	0.527

Table 9. Performance comparisons of the dimension reduction schemes, each having 10 reduced features, applied before classification of the Reuters ModApte-10 split dataset.

ModApte-10 split dataset		AFE	Chi-squared	Correlation coefficient	PCA	LSA
Naïve Bayes	Precision	0.918	0.755	0.859	0.896	0.881
	Recall	0.911	0.516	0.657	0.892	0.869
	F_1 measure	0.911	0.504	0.706	0.893	0.869
C4.5	Precision	0.949	0.788	0.798	0.886	0.886
	Recall	0.949	0.809	0.806	0.881	0.883
	F_1 measure	0.948	0.795	0.796	0.879	0.883
RIPPER	Precision	0.950	0.742	0.747	0.872	0.889
	Recall	0.949	0.769	0.765	0.869	0.890
	F_1 measure	0.949	0.748	0.740	0.866	0.887
1-nearest neighbor	Precision	0.957	0.729	0.752	0.900	0.895
	Recall	0.956	0.737	0.761	0.895	0.890
	F_1 measure	0.956	0.732	0.755	0.896	0.891
10-nearest neighbor	Precision	0.962	0.796	0.799	0.916	0.915
	Recall	0.962	0.806	0.809	0.915	0.914
	F_1 measure	0.961	0.797	0.800	0.914	0.913
Random forest	Precision	0.961	0.733	0.773	0.904	0.905
	Recall	0.960	0.742	0.782	0.900	0.907
	F_1 measure	0.960	0.736	0.776	0.899	0.904
SVM	Precision	0.961	0.804	0.796	0.907	0.892
	Recall	0.959	0.814	0.808	0.891	0.891
	F_1 measure	0.957	0.802	0.793	0.870	0.875
LINEAR	Precision	0.953	0.790	0.803	0.895	0.765
	Recall	0.945	0.808	0.813	0.894	0.792
	F_1 measure	0.937	0.785	0.804	0.880	0.736
Average	Precision	0.951	0.767	0.791	0.897	0.879
	Recall	0.949	0.750	0.775	0.892	0.880
	F_1 measure	0.947	0.737	0.771	0.887	0.870

Table 10. A simple 2-class dataset.

		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Class 1	$Sample_1$	1	1	1	0	0	0	0	0
	$Sample_2$	0	1	0	1	0	0	0	1
	$Sample_3$	0	2	0	0	1	0	0	0
Class 2	$Sample_4$	0	0	0	0	1	1	0	0
	$Sample_5$	0	0	0	0	1	0	1	1
	$Sample_6$	0	1	0	0	1	1	0	0

6. Discussion

The AFE depends on the class membership probabilities of the samples, depending on the features that they contain. We weight the features and observe their probabilistic distribution over the classes. Projecting and summing up the probabilities of features to the classes gives us the impact of each extracted abstract feature to each class. This extraction procedure reveals the evidence in the training samples about the classes. These evidences are actually hidden in the features.

In this section, we give a brief example to visualize the abstract features in a 2-class problem. We also visualize the abstract features extracted from our selected datasets in the experimental results.

6.1. Abstract features of a sample two-class problem

Assume that we have a 2-class dataset with 6 samples and 8 features. Let the values of the features be as in Table 10. Applying the AFE on this dataset gives us the extracted features listed in Table 11. If we visualize Table 11 on Figure 2, we can easily track the evidences of class memberships hidden in the samples. The extracted abstract features can be seen as the membership probabilities of samples to the classes.

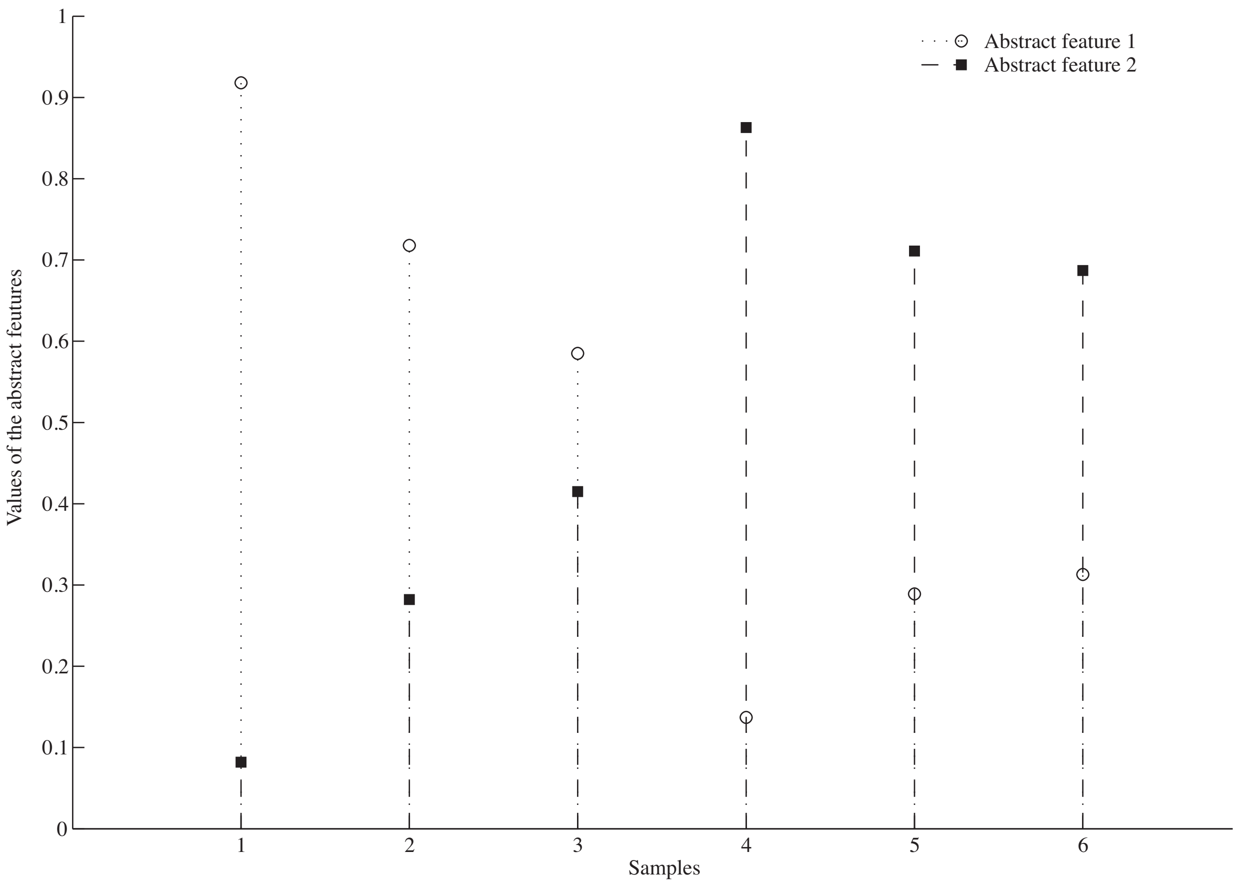

Figure 2. Visualization of the extracted abstract features for the given 2-class dataset.

Table 11. Values of the extracted features for the given 2-class dataset.

	$AbstractFeature_1$	$AbstractFeature_2$
$Sample_1$	0.918	0.082
$Sample_2$	0.718	0.282
$Sample_3$	0.585	0.415
$Sample_4$	0.137	0.863
$Sample_5$	0.289	0.711
$Sample_6$	0.313	0.687

As we have 2 classes in our example, we have 2 abstract features extracted with the AFE, whose values are given in Table 11. In order to observe the class separabilities, we apply PCA an LSA to the sample dataset and extract 2 features with each method. The extracted values of the samples in our dataset with the AFE, PCA, and LSA are compared in Figure 3. When we observe the distribution of the samples, we see that the abstract features extracted with the AFE have the most definite and distinct discriminant and, thus, the clearest separability. Features extracted with PCA can separate linearly, but its discriminant is not as clear and apart as the AFE's. Samples cannot be linearly separated using the extracted features of LSA. Therefore, the discriminant of LSA is quadratic, which reduces the performances of classifiers because of its complexity.

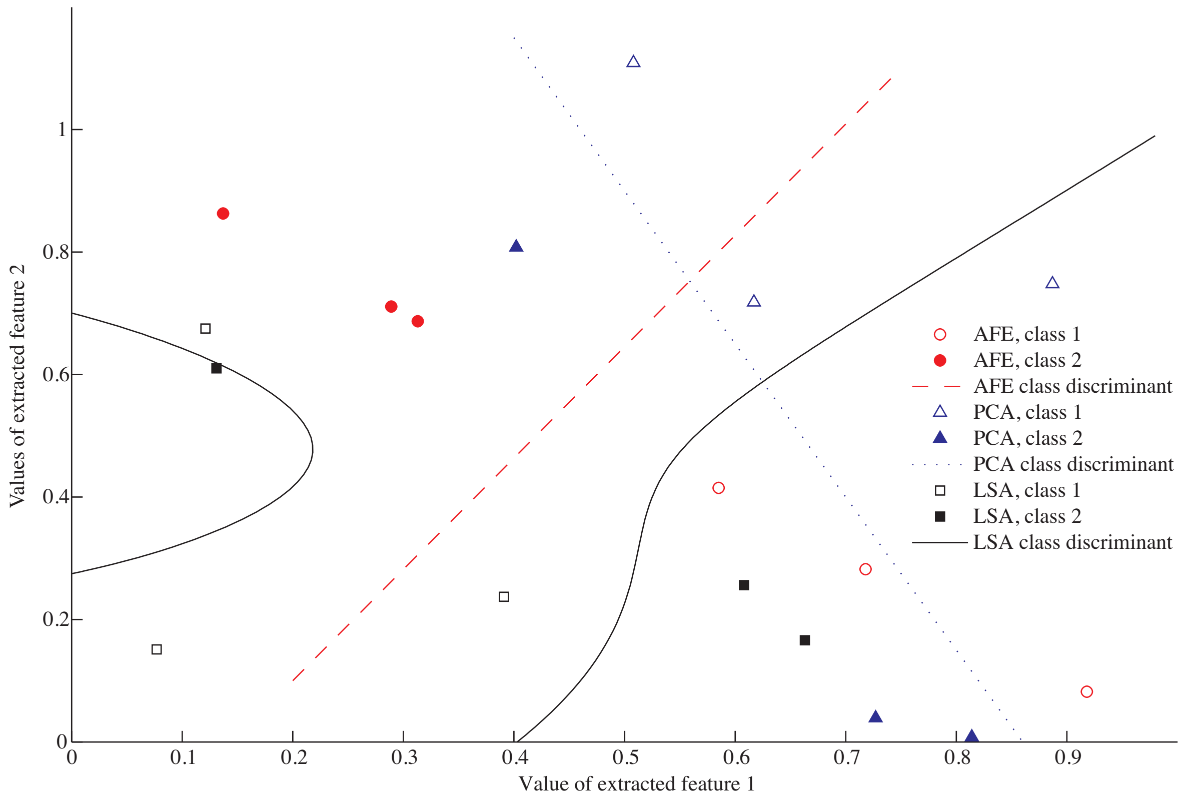

Figure 3. Class discriminants and samples with extracted features using the AFE, PCA, and LSA for the given 2-class dataset.

6.2. Abstract features extracted from the experimental results

The averages of the abstract features extracted from the Reuters, 20 Newsgroups, and ModApte-10 datasets are given in Figures 4, 5, and 6. We see that each abstract feature gets the highest score in its class in our experimental tests. The consequent scored features show the likelihood of the samples in that class to other classes.

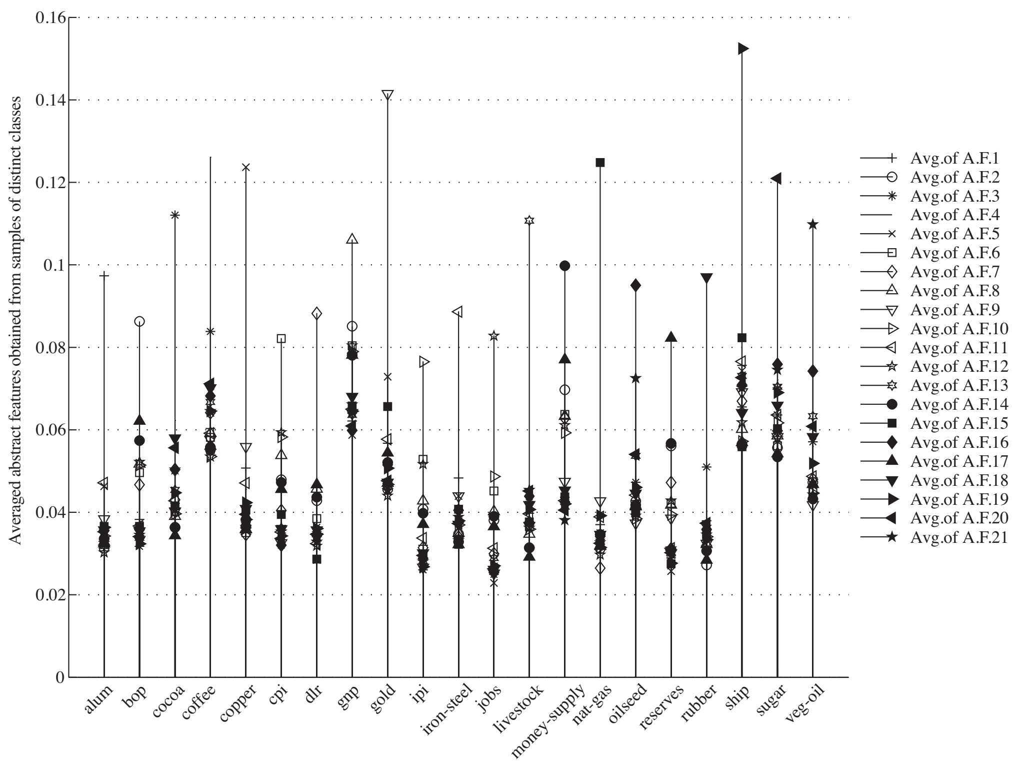

Figure 4. Averages of the abstract features extracted from the Reuters dataset, each obtained from the samples that belong to the corresponding class.

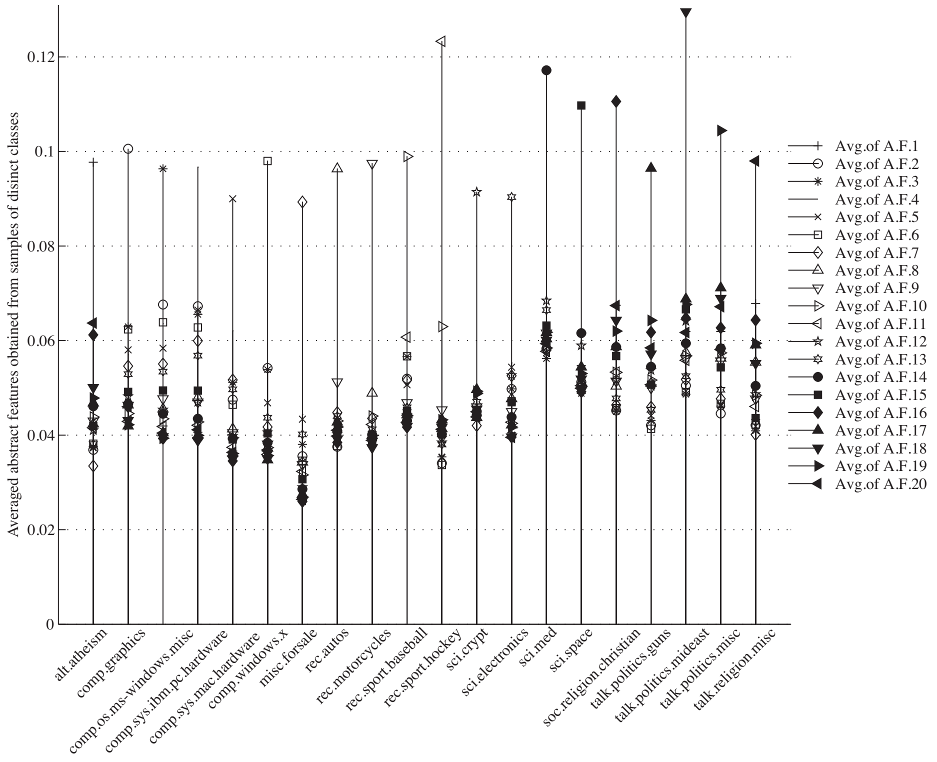

Figure 5. Averages of the abstract features extracted from the 20 Newsgroups dataset, each obtained from the samples that belong to the corresponding class.

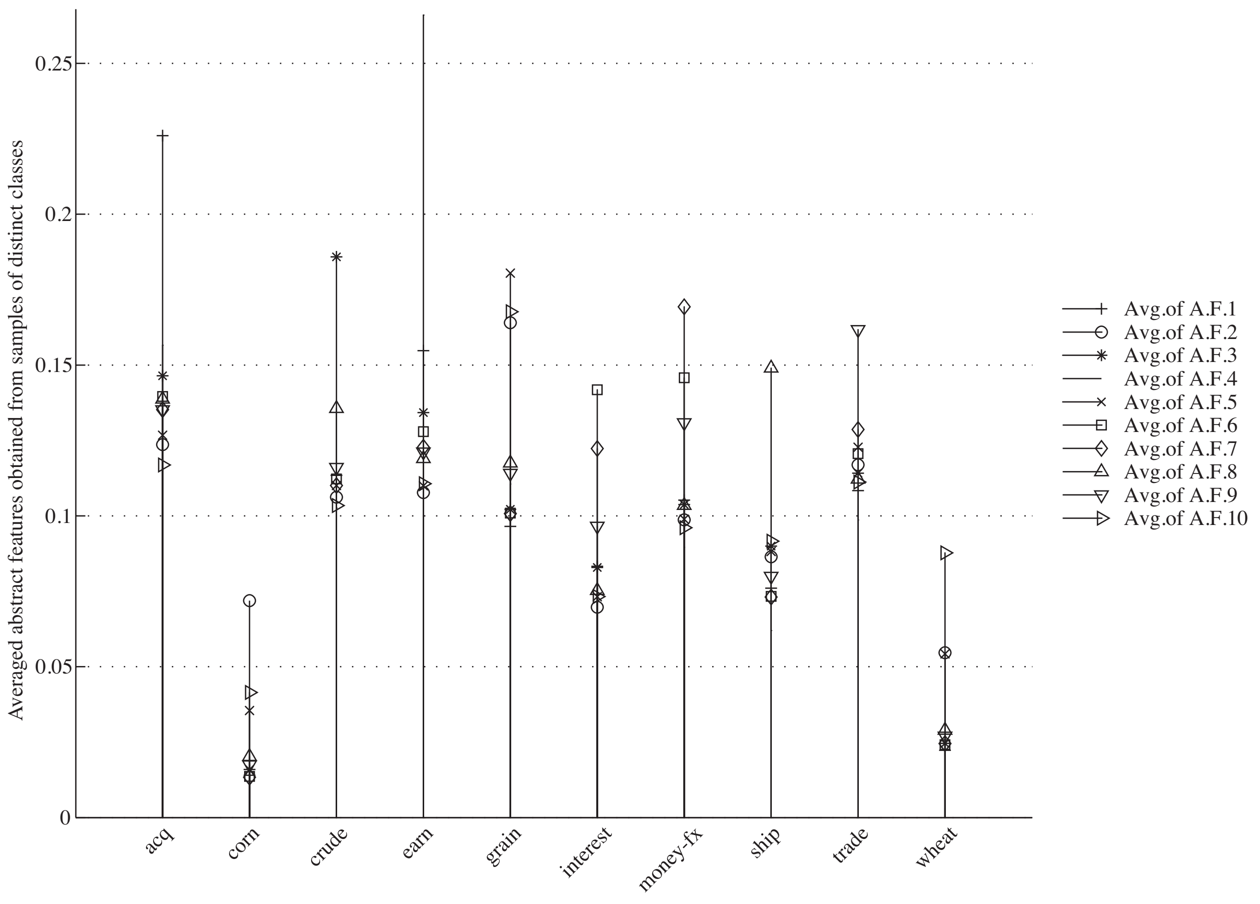

Figure 6. Averages of the abstract features extracted from the ModApte-10 split of the Reuters dataset, each obtained from the samples that belong to the corresponding class.

We can say that if the values of the abstract features are close to each other, the class separability is low. Contrarily, distinct values between the abstract features show us that the classes of the dataset are easy to distinguish. Examining the abstract features extracted from the Reuters, 20 Newsgroups, and ModApte-10 datasets, we see that the 20 Newsgroups dataset has the highest class separability, followed by the Reuters dataset. The class separability of the ModApte-10 dataset is the lowest compared with previous datasets.

7. Conclusions

We introduce a feature extraction method that summarizes the features of the samples, where the extracted features aggregate information about how much evidence there is in the features of the training samples for each class. In order to form the abstract features, high dimensional features of the samples are projected onto a new feature space having dimensions equal to the number of classes.

We choose text classification to evaluate the AFE and compare it with other popular feature selection and feature extraction schemes. Seven classifiers of different types are used to compensate the dependencies on the algorithm types and to effectively test the behaviors of the dimension reduction schemes. We examine performances of the classifiers on 3 standard and popular text collections: the Reuters-21578, 20 Newsgroups, and the ModApte-10 split of Reuters. We work on a vector space model, which causes an excess number of features. TFIDF term weighting is used to score the input features of the samples. Using the AFE, we project the words in documents onto a new feature space having dimensions equal to the number of classes. Comparison and test results show that the AFE scores the highest F_1 measure on the Reuters dataset with 96.9%, the 20 Newsgroups dataset with 94.8%, and the ModApte-10 with 96.1%. This means that the AFE achieves a better F_1 measure of 3.7% on the Reuters, 19.4% on the 20 Newsgroups, and 3.4% on the ModApte-10 than its nearest following non-AFE method. Looking at the average F_1 measures of the classifiers, we see that the AFE's score is 9.2% higher on Reuters, 33.0% higher on 20 Newsgroups, and 7.3% higher on ModApte-10 than the next best scored method.

Not only does AFE make it possible to prepare datasets for classification in an effective way, but it also gives information about class separability. The training samples include evidences about the classes. These evidences are hidden in the features. What the AFE reveals are these evidences. In other words, the abstract features extracted by the AFE can be seen as the membership probabilities of the samples to the classes. These features also describe the likelihood of a sample to other classes. We can infer that if the values of the abstract features are close to each other, class separability is low. As the distances between the abstract features increase, it becomes easier to distinguish the classes. Hence, we can comprehend the separability of the classes by using the AFE.

References

[1] L.M. Chan, Cataloging and Classification: An Introduction, New York, McGraw-Hill, 1994.

[2] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization", Proceedings of the 14th International Conference on Machine Learning, pp. 143-151, 1997.

[3] M. Efron, "Query expansion and dimensionality reduction: notions of optimality in Rocchio relevance feedback and latent semantic analysis", Information Processing & Management, Vol. 44, pp. 163-180, 2008.

[4] K. Bunte, B. Hammer, A. Wismüller, M. Biehl, "Adaptive local dissimilarity measures for discriminative dimension reduction of labeled data," Neurocomputing, Vol. 73, pp. 1074-1092, 2010.

[5] I. Fodor, A Survey of Dimension Reduction Techniques, US DOE Office of Scientific and Technical Information, Washington DC, 2002.

[6] Y. Yang, J. Pedersen, "A comparative study on feature selection in text categorization", Proceedings of the 14th International Conference on Machine Learning, pp. 412-420, 1997.

[7] I. Guyon, "An introduction to variable and feature selection", Journal of Machine Learning Research, Vol. 3, pp. 1157-1182, 2003.

[8] H. Soyel, H. Demirel, "Optimal feature selection for 3D facial expression recognition using coarse-to-fine-classification", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 18, pp. 1031-1040, 2010.

[9] J. Zhu, H. Wang, X. Zhang, "Discrimination-based feature selection for multinomial naïve Bayes text classification", Proceedings of the 21st International Conference on the Computer Processing of Oriental Languages, pp. 149-156, 2006.

[10] I. Guyon, H.M. Bitter, Z. Ahmed, M. Brown, J. Heller, "Multivariate non-linear feature selection with kernel methods", Studies in Fuzziness and Soft Computing, Vol. 164, pp. 313-326, 2005.

[11] A.M. Dragos, "Feature extraction – a pattern for information retrieval", Proceedings of the 5th Conference on Pattern Languages of Programs, pp. 391-412, 1998.

[12] I. Guyon, S. Gunn, M. Nikravesh, L. Zadeh, Feature Extraction: Foundations and Applications, Berlin, Springer, 2006.

[13] F.S. Tsai, "Dimensionality reduction techniques for blog visualization", Expert Systems with Applications, Vol. 38, pp. 2766-2773, 2011.

[14] R. Jensen, Q. Shen, Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches, New Jersey, Wiley, 2008.

[15] M. Hall, L. Smith, "Practical feature subset selection for machine learning", Proceedings of the 21st Australasian Conference on Computer Science, pp. 181-191, 1998.

[16] Z. Zheng, X. Wu, R. Srihari, "Feature selection for text categorization on imbalanced data", SIGKDD Explorations, Vol. 6, pp. 80-89, 2004.

[17] M. Hall, Correlation-Based Feature Selection for Machine Learning, PhD thesis, University of Waikato, New Zealand, 1998.

[18] I.T. Jolliffe, Principal Component Analysis, 2nd ed., New York, Springer, 2002.

[19] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd ed., San Diego, Academic Press, 1990.

[20] T.K. Landauer, S.T. Dumais, "A solution to Pluto's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge", Psychological Review, Vol. 104, pp. 211-240, 1997.

[21] M. Brand, "Fast low-rank modifications of the thin singular value decomposition", Linear Algebra and its Applications, Vol. 415, pp. 20-30, 2006.

[22] A. Martinez, A. Kak, "PCA versus LDA", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, pp. 228-233, 2001.

[23] İ. Kocabaş, B.T. Dinçer, B. Karaoğlan, "Investigation of Luhn's claim on information retrieval", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 19, pp. 1-12, 2011.

[24] M. Lan, C.L. Tan, J. Su, Y. Lu, "Supervised and traditional term weighting methods for automatic text categorization", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, pp. 721-735, 2009.

[25] M.F. Porter, "An algorithm for suffix stripping", Program, Vol. 14, pp. 130-137, 1980.

[26] A. McCallum, K. Nigam, A Comparison of Event Models for Naïve Bayes Text Classification, Palo Alto, AAAI Press, 1998.

[27] J.R. Quinlan, C4.5: Programs for Machine Learning, Waltham, Morgan Kaufmann, 1993.

[28] W.W. Cohen, "Fast effective rule induction", Proceedings of the 29th International Conference on Machine Learning, pp. 115-123, 1995.

[29] L. Breiman, "Random forests", Machine Learning, Vol. 45, pp. 5-32, 2001.

[30] C. Cortes, V. Vapnik, "Support-vector networks", Machine Learning, Vol. 20, pp. 273-297, 1995.

[31] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, C.J. Lin, "Liblinear: a library for large linear classification", Journal of Machine Learning Research, Vol. 9, pp. 1871-1874, 2008.

[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, "The Weka data mining software: an update", SIGKDD Explorations, Vol. 11, pp. 10-18, 2009.