Knowledge-Based Systems 118 (2017) 152–164

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

MDLText: An efficient and lightweight text classifier

Renato M. Silva [a], Tiago A. Almeida [b,*], Akebo Yamakami [a]

[a] Department of Systems and Energy, University of Campinas – UNICAMP Campinas, São Paulo, Brazil
[b] Department of Computer Science, Federal University of São Carlos – UFSCar, Sorocaba, São Paulo, Brazil

ARTICLE INFO

Article history:
Received 28 June 2016
Revised 15 November 2016
Accepted 25 November 2016
Available online 25 November 2016

Keywords:
Text categorization
Minimum description length
Classification
Machine learning
Natural language processing

ABSTRACT

In many areas, the volume of text information is increasing rapidly, thereby demanding efficient text classification approaches. Several methods are available at present, but most exhibit declining performance as the dimensionality of the problem increases, or they incur high computational costs for training, which limit their application in real scenarios. Thus, it is necessary to develop a method that can process high dimensional data in a rapid manner. In this study, we propose the `MDLText`, an efficient, lightweight, scalable, and fast multinomial text classifier, which is based on the minimum description length principle. `MDLText` exhibits fast incremental learning as well as being sufficiently robust to prevent overfitting, which are desirable features in real-world applications, large-scale problems, and online scenarios. Our experiments were carefully designed to ensure that we obtained statistically sound results, which demonstrated that the proposed approach achieves a good balance between predictive power and computational efficiency.

1. Introduction

The amount of digital information stored in text format is increasing every day. Books, newspapers, magazines, and many other examples of traditional printed resources are now available as digital media. Even preliminary documents must be scanned to facilitate their control, safety, and access. In addition, many existing communication services, such as email, online instant messengers, social networks, comments on websites and blogs generate textual information, which needs to be analyzed to ensure its security as well as to improve content organization in order to provide a satisfactory user experience.

Efficient text classification approaches are needed increasingly to deal with the rapidly growing volume of user-generated content and the vast amount of digital information that is currently available. Several machine learning-based methods have been used, such as the established support vector machines (SVM) [9,14], naïve Bayes [44], decision trees (DT) [13,50], and *k*-nearest neighbors (KNN) [15].

In real-world applications, the dimensionality of the data is usually huge, which affects the performance of most well-known machine learning algorithms [68]. High-dimensional data can lead to the so-called curse of dimensionality and it may also be difficult to train the classifiers without incurring high computational costs.

A text classification technique should ideally be trained rapidly, such as naïve Bayes, and sufficiently robust to prevent overfitting, such as SVM. However, the performance of naïve Bayes classifiers unsurprisingly degrades when applied to high-dimensional data [4]. Moreover, SVMs often require costly and time-consuming training, especially in real-world applications where the number of classes is large and the feature space is huge [68].

Some well-known machine learning methods (*e.g.*, SVM with traditional kernel functions) cannot be employed to deal with real-world text classification problems because they require that all of the examples should be stored in memory, or they must be presented simultaneously in a process known as batch or offline learning. Thus, various online machine learning approaches have been proposed. In general, these methods are simpler and faster than batch learning-based alternatives because they process and learn with one example at a time. However, if batch learning-based methods can be applied, they usually perform better than online learning techniques [16].

To address this problem, we propose `MDLText` as a novel multinomial text classification approach, which is based on the minimum description length (MDL) principle [53,54]. This method has the inherent ability to prevent overfitting because it selects a model that fits the data well, while it naturally favors less complex models. Other classifiers include some way for preventing overfitting, however, MDLText does this in a natural way and hence

* Corresponding author.
E-mail addresses: renatoms@dt.fee.unicamp.br (R.M. Silva), talmeida@ufscar.br (T.A. Almeida), akebo@dt.fee.unicamp.br (A. Yamakami).

http://dx.doi.org/10.1016/j.knosys.2016.11.018
0950-7051/

it avoids employing additional regularization alternatives. Furthermore, it has a low computational cost, even for problems comprising a large volume of documents and a high-dimensional feature space. It can also be applied to multi-class problems without tricks, which avoids using approaches such as one-against-one and variants [29]. Moreover, `MDLText` can be used in online and dynamic scenarios because it allows incremental learning.

To support our claims, we performed a comprehensive performance evaluation to determine the efficiency and effectiveness of the proposed text classifier in batch and online learning scenarios, where we employed 45 well-known text corpora from various domains. Moreover, we compared our method with benchmark machine learning algorithms, which were trained with text documents represented by three different established term weighting schemes: binary, term-frequency (TF), and term frequency-inverse document frequency (TF-IDF) [60,61].

The remainder of this paper is organized as follows. In Section 2, we briefly describe the related work available in literature. Basic concepts related to the MDL principle are presented in Section 3. In Section 4, we explain the proposed text classification method. The experimental settings are described in Section 5. Section 6 presents the results obtained from the batch and online learning tasks. Finally, we give our main conclusions and suggestions for future research in Section 7.

2. Related work

Machine learning methods have been used to solve several established text classification problems in a large amount of scenarios, such as sentiment analysis [47], news articles [25,33,57,69], scientific documents [33], and emails [4,57,65]. Recently, some researches have also applied machine learning methods to classify short message service (SMS) [1,3], blog spam [5], and social media [2].

The most widely used model for text classification is the vector space model (VSM) [61] where each text document is represented by a feature vector, where each feature is a word (term or token) and the feature's value is a term weight. The most used term weighting schemes in text classification are:

- *Binary*: the term weight in a document is either 1, when the term appears in the document, or 0 [4,20];
- *TF*: the term weight in a document is given by its frequency of occurrence in the document [2,20,25,26,37]; and
- *TF–IDF*: the term weight in a document is given by its frequency of occurrence in the document times the inverse frequency of the term in the training set [20,52,67,69].

There are other term weight schemes recently proposed in literature. For instance, [64] introduced a new term-weighting method called `ConfWeight` based on statistical estimation of the importance of a word for a specific categorization problem. Later, a new term-weighting scheme that uses class information was proposed by [38]. Basically, it uses the odds of positive and negative class probabilities. In other work, [20] proposed a genetic program to learn effective term-weighting schemes in the context of text classification.

One of the most employed machine learning approach as a baseline in text categorization problems is the established naïve Bayes classifier [4,42,57]. In addition to the well-known models (*e.g.*, multinomial naïve Bayes [44] and Bernoulli naïve Bayes [4,44]), new proposals have been appeared in the last years. For example, [37] introduced a topic document model approach for naïve Bayes text classification where the probability of a term given a class is estimated by averaging the occurrence probabilities in each training document belonging to that class. More recently, [32] proposed a discriminatively weighted naïve Bayes that is an improved naïve Bayes algorithm by discriminative instance weighting. In other work, [66] presented a new algorithm called multinomial naïve Bayes tree that deploys a multinomial naïve Bayes text classifier on each leaf node of a decision tree. Other models have also been proposed by [33] and [31].

Another widely used machine learning method as a baseline in text classification is the traditional *k*-nearest neighbors classifier [3]. New approaches based on KNN have also been proposed recently. For example, [26] proposed a weight adjusted *k*-nearest neighbor classification that learns feature weights based on a greedy hill climbing technique. Later, [35] proposed an approach that combine a constrained one pass clustering algorithm [34] with KNN.

Other well-known classification techniques, such as decision trees [13,50], Rocchio [56], random forest [69], and support vector machines [14] are also widely used in text classification [3,4,42,57,69].

Some works in literature have attempted to scale text classification problems, since in real-world scenarios the number of documents is usually big and the dimensionality is huge. For this, attribute selection techniques are the most used approaches. Many of these techniques are presented in [62]. Other two approaches are presented in [25], which proposed two variants of linear forward selection, an attribute selection technique, for classification. Moreover, [71] proposed a gain ratio-based hybrid feature selection approach to naïve Bayes text classifiers. It integrates the advantages of filter and wrapper approaches. [36] presented a new method called `fastText`, which extends `word2vec` for text classification and representation learning. According to the authors, such method can be trained with more than one billion words in a few minutes using a standard multi-core CPU and, moreover, it can classify half a million sentences in less than a minute.

Methods proposed to scale text categorization problems are also important in short text classification. Techniques for enriching document representation are usually applied and, as a consequence, the dimensionality of the feature space is often increased. For example, [30] proposed a method to extract concepts and category information from Wikipedia to used them for enriching the document representation. [48] proposed a framework to assist the classification of short and sparse text documents by using hidden topics discovered from external data collections. According to the authors, their framework reduces the sparsity and ambiguity of the text documents and provide a lot of unknown words from the external data collections, which benefits the classification performance. A similar approach was also proposed by [70]. All these techniques aim to assist the text classifiers by enriching the text representation, but they require classification methods able to process high dimensional data in a rapid manner.

3. The MDL principle

The MDL principle states that if we need to choose between two or more models to fit some data, we should select that with the smallest description length [53,54]. This means that less complex models are preferable [7,24]. This idea is a formalization of the principle of parsimony, which is also known as Occams razor [18].

The MDL is based on Kolmogorov complexity, which can briefly be defined as the smallest program size that describes data represented by a binary sequence [40]. The lower the Kolmogorov complexity for a given sequence, the more regularity is present. Therefore, the smallest program that can lead to a certain sequence is also the one that fits it better, and thus it should be chosen to represent that sequence.

In practice, this means that MDL-based classifiers naturally and simultaneously achieve a good trade-off between the complexity

of the model and the quality of its fit to the data. This characteristic is particularly desirable in machine learning because it has the inherent potential to avoid overfitting [23].

Given a set of potential models $M_1, M_2, \ldots, M_{|M|}$, the MDL would select one by:

$$M_{mdl} = \arg\min_{\forall M}[L(M) + L(X|M)], \tag{1}$$

where $L(M)$ is the description length of model M, which is represented by its complexity, and $L(X|M)$ is the description length of data X when encoded by model M, thereby representing how well M fits X. [53] demonstrated that $L(X|M)$ can be calculated using Shannon–Fano coding and thus $L(X|M) = -\log P(X|M)$, where $P(X|M)$ is the conditional probability of X given M. This is known as crude MDL or two-part MDL.

According to [23], a problem with crude MDL is finding a good code for model M, because $L(M)$ might become arbitrary as it can be very large for a given code and very small for another. Thus, [55] proposed the refined MDL by using universal codes to measure the description length of the model. Basically, only a one-part code with length $\bar{L}(X|M)$ is used.

Subsequently, some studies have proposed MDL-based approaches to solve machine learning problems where one of the pioneering methods was described by [51]. Related studies with similar applications were also proposed by [39] and [45]. In [41] and [22], the MDL was used to assist the learning process in Bayesian networks.

Feature selection is another traditional machine learning problem, which has also been addressed by MDL-based approaches. For instance, [10] employed the MDL as a criterion for selecting relevant features from micro-array datasets with a large feature space and few samples. According to the authors, the performance of MDL criterion for feature selection was comparable to entropy-based methods.

MDL-based techniques have also been proposed to address time series problems. For instance, [8] presented a parameter-free stopping criterion for semi-supervised learning, which was tested successfully with medical datasets.

Image analysis is another field that has benefited from the use of MDL. For instance, [49] presented an extension of MDL called representational MDL, which allows the construction of image analysis methods with a high capacity for learning based on the automatic optimization of representations.

The studies mentioned above represent only a small subset of the existing MDL-based approaches proposed to address machine learning problems. However, these studies interpreted and applied the MDL principle in very different ways, because there is no standard procedure for calculating the description length of a model. Some studies use MDL principle as a broad umbrella term for all types of inductive inference based on data compression. For others, the MDL principle is any method based on the idea of finding the model that describes the data with the minimum description length. There are also authors that use MDL principle to describe model selection criterion equivalent to the Bayesian information criterion or even interpret the MDL principle using a Bayesian perspective, such as [19] and [43]. Other researchers have compared MDL to the maximum a posteriori method, as [59] and [46]. Therefore, there is no standard way to apply MDL in machine learning problems and, specifically, in classification problems.

To the best of our knowledge, despite some MDL-based approaches have been proposed to address machine learning problems, there is no classification technique that employs the MDL principle as the main concept in its predictive model. Some methods use the MDL principle in secondary stages, such as [51], where the MDL principle is used to find the "best" decision tree to infer from the given data.

In this study, we propose `MDLText` as a new MDL-based multinomial text classifier. In the following, we provide an original interpretation of how to calculate the description length of a model (or class) in text classification tasks.

4. Mathematical basis of MDLText

Given an unlabeled text document d, the `MDLText` uses the main equation of the MDL principle (Eq. (1)) to predict the class of the document. The set of potential classes $c_1, c_2, \ldots, c_{|C|}$ represents the set of potential models M, while d represents the data X. Therefore, d receives the label j, which corresponds to class c_j with the minimum overall description length related to d:

$$c(d) = \arg\min_{\forall c} L(d|c_j). \tag{2}$$

We have ignored the description length of the potential classes (models) because, as we stated in Section 3, it might become arbitrary and difficult to find [55]. Therefore, we employed a straightforward application of the MDL principle only concerned with the description length of d for each potential class.

In a traditional MDL classifier, the description length of the document d with respect to c_j (Eq. (2)) can be computed by summing the description lengths of all the tokens in d, given by:

$$L(d|c_j) = \sum_{i=1}^{|d|} L(t_i|c_j), \tag{3}$$

where $|d|$ is the amount of tokens in document d and $L(t_i|c_j)$ is the description length of the token t_i with respect to c_j, computed by:

$$L(t_i|c_j) = \lceil -\log_2 \beta(t_i|c_j) \rceil. \tag{4}$$

$\beta(t_i|c_j)$ is calculated based on the TF-IDF weight of the token t_i in each document of the training set [67]. The TF–IDF of a token t_i in a document d is given by:

$$w(t_i, d) = \log(1 + TF(t_i, d)) \times \log\left(\frac{|\mathcal{D}| + 1}{DF_{t_i} + 1}\right), \tag{5}$$

where $TF(t_i, d)$ is the local term frequency of t_i, (*i.e.*, how many times t_i appears in document d), $|\mathcal{D}|$ is the amount of documents in the training set, and DF_{t_i} is the number of documents in the training set in which t_i appears.

Then, the L2 normalization is applied:

$$\hat{w}(t_i, d) = \frac{w(t_i, d)}{\|w(:, d)\|_2}. \tag{6}$$

In Eq. (4), $\beta(t_i|c_j)$ can be calculated as:

$$\beta(t_i|c_j) = \frac{n_{c_j,t_i} + \frac{1}{|\Omega|}}{\hat{n}_{c_j} + 1}, \tag{7}$$

where $n_{c_j,t_i} = \sum_{\forall d} \hat{w}(t_i, d|c_j)$ and $\hat{n}_{c_j}$ is the sum of n_{c_j,t_i} for all the tokens that appear in the training documents belonging to c_j. The parameter $|\Omega|$ is used to preserve a portion of the description length for tokens that have never appeared in the training set $\mathcal{D}$ (*i.e.*, tokens with $n_{c_j,t_i} = 0$).

The variables n and $\hat{n}$ are part of the training model. While n_{c_j,t_i} is a position of n that keeps the values of n_{c_j,t_i} for all classes and tokens that appeared in the training documents, $\hat{n}_{c_j}$ is a position of $\hat{n}$, that keeps the values of $\hat{n}_{c_j}$ for all classes.

According to [52], using the TF–IDF weight instead of the raw term frequencies can improve the performance of methods that employ MLE. The Eq. (7) is similar to naïve Bayes estimation without priors, using log probabilities rather than products.

As shown in Fig. 1, when $|\Omega|$ is higher, the conditional weight is smaller, and thus the description length of tokens with $n_{c_j,t_i} = 0$ is larger. Therefore, $|\Omega|$ regulates how tokens with $n_{c_j,t_i} = 0$ will

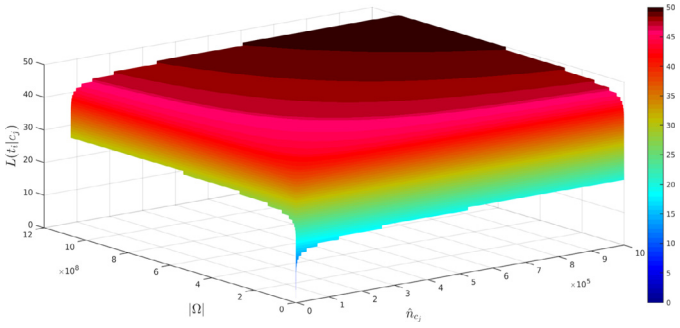


Fig. 1. Effects of $|\Omega|$ and $\hat{n}_{c_j}$ on the value of $L(t_i|c_j)$ for tokens with $n_{c_j,t_i} = 0$.

contribute to the description length of a document with respect to the class c_j. Fig. 1 also indicates that if $\hat{n}_{c_j}$ is higher, the description length of the token with respect to class c_j is larger, when $n_{c_j,t_i} = 0$.

Fig. 2 illustrates how $L(t_i|c_j)$ varies according to $\beta(t_i|c_j)$. If $\beta(t_i|c_j)$ is higher, then $L(t_i|c_j)$ is smaller. It is important to note that in practice, $\beta(t_i|c_j)$ is often less than 0.02 (gray region in Fig. 2a) because text classification problems usually have a large amount of high dimensional samples represented by a highly sparse vector. Thus, n_{c_j,t_i} is usually much smaller than $\hat{n}_{c_j}$.

We couple two other parameters to the original formula (Eq. (3)): a penalty function $\hat{S}(d, c_j)$ for each class and a penalty function $K(t_i)$ for each token, as follows:

$$L(d|c_j) = \hat{S}(d, c_j) \times \sum_{i=1}^{|d|} L(t_i|c_j) \times K(t_i). \tag{8}$$

The penalty function $\hat{S}(d, c_j)$ is based on the cosine similarity function between the document and a prototype vector of the class:

$$\hat{S}(d, c_j) = -\log_2\left(\frac{1}{2} \times S(d, \bar{c}_j)\right). \tag{9}$$

The cosine similarity $S(d, \bar{c}_j)$ measures the similarity between the document d and a prototype vector $\bar{c}_j$ in an analogous manner to the Rocchio algorithm [56]. The similarity $S(d, \bar{c}_j)$ ranges from 0 to 1, where higher values indicate greater similarity. The function $\hat{S}(d, c_j) > 1$ penalizes the class with lower similarity to the document. If the similarity is lower, the value obtained in $\hat{S}(d, c_j)$ is greater (see Fig. 3), and thus the description length of the document d given the class c_j is larger.

The cosine similarity can be calculated by:

$$S(d, \bar{c}_j) = \frac{\sum_{i=1}^{|d|} \hat{w}(t_i, d|c_j) \times \bar{c}_j(t_i)}{\|\hat{w}(:, d)\|_2 \times \|\bar{c}_j\|_2} \tag{10}$$

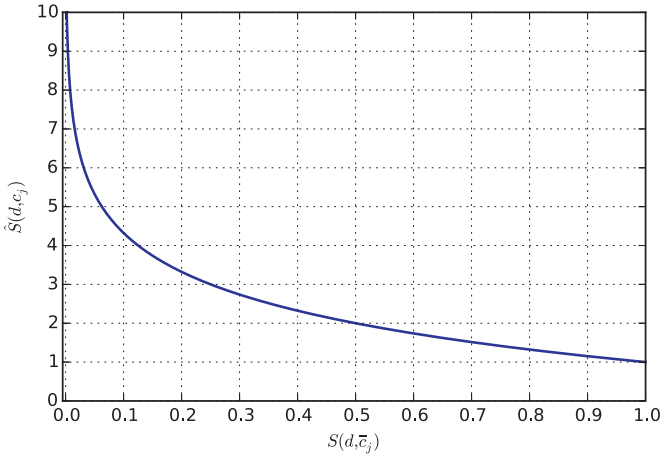


Fig. 3. Variation in $\hat{S}(d, c_j)$ according to $S(d, \bar{c}_j)$.

and the prototype vector $\bar{c}_j$ is an average vector of the tokens. Therefore, for each token t_i, $\bar{c}_j(t_i)$ is:

$$\bar{c}_j(t_i) = \frac{n_{c_j,t_i}}{|\hat{\mathcal{D}}_{c_j}|} \tag{11}$$

where $|\hat{\mathcal{D}}_{c_j}|$ is the amount of training documents in class c_j. $|\hat{\mathcal{D}}_{c_j}|$ is a position of $|\hat{\mathcal{D}}|$, that keeps the values of $|\hat{\mathcal{D}}_{c_j}|$ for all classes. The variable $|\hat{\mathcal{D}}|$ is also part of the training model.

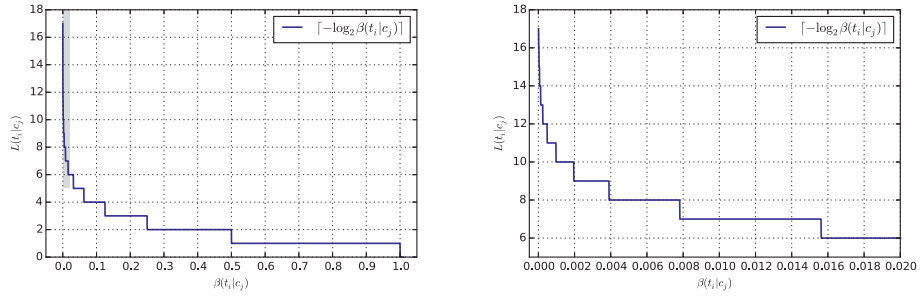
The penalty function $K(t_i)$ associated with each token can be calculated by:

$$K(t_i) = \frac{1}{(1+\alpha) - F(t_i)}, \tag{12}$$

where $0 \leq F(t_i) \leq 1$ is a score for token t_i and $\alpha > 0$ is a constant used to avoid a denominator of 0. In this study, we empirically set $\alpha = 10^{-3}$. Fig. 4 shows how the penalty function $K(t_i)$ varies according to $F(t_i)$.

The main goal of using the penalty function $K(t_i)$ is to enhance the ability to separate the classes. The idea behind this function is that tokens with high score will contribute more to $L(d|c_j)$. Specifically, suppose that a token $t^\star$ with a high score appears in several documents from one specific class (*e.g.*, $j = 1$) but in few documents from other classes ($j > 1$). Thus, $L(t^\star|c_1)$ will be much smaller than any $L(t^\star|c_j)$ for $j > 1$. $F(t^\star)$ is high, so $K(t^\star)$ is also high. As a consequence, $L(t^\star|c_1) \times K(t^\star)$ will be much smaller than $L(t^\star|c_j) \times K(t^\star)$ for $j > 1$. For tokens with low score (small $K(t_i)$), their contribution to $L(d|c_j)$ will be reduced. It is important to note that a token with high score does not necessarily have a high frequency.

Any score function that returns values between 0 and 1 can be used to calculate $F(t_i)$, since it takes into account the frequency of



(a) (b)

Fig. 2. Effects of $\beta(t_i|c_j)$ on the value of $L(t_i|c_j)$. Fig. 2b shows the gray region from Fig. 2a.

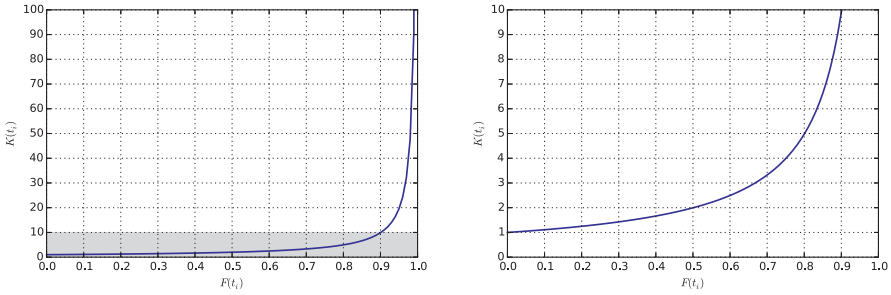


Fig. 4. Variation in $K(t_i)$ according to $F(t_i)$. Fig. 4b shows the gray region from Fig. 4a.

the token t_i in each class. However, in this study we employed the confidence factors (CF) [6] that assigns a score to a given token t_i by:

$$F(t_i) = \frac{1}{m-1} \sum_{\forall j | j \neq v} \frac{\left(\frac{(MH)^2 + pH - \frac{\lambda_1}{SH}}{(SH)^2} \right)^{\lambda_2}}{1 + (\frac{\lambda_3}{SH})}, \tag{13}$$

where v is the index of the most frequent class; $j = 1, \ldots, m$ is the index of classes; $SH = \phi_{c_v,t_i} + \phi_{c_j,t_i}$, $MH = \phi_{c_v,t_i} - \phi_{c_j,t_i}$ and $PH = \phi_{c_v,t_i} \times \phi_{c_j,t_i}$, where ϕ_{c_v,t_i} is the number of documents with token t_i in the most frequent class and ϕ_{c_j,t_i} is the amount of documents with token t_i in class c_j; and λ_1, λ_2, λ_3 are constants that adjust the decay speed. We used $\lambda_1 = 0.25$, $\lambda_2 = 10.0$, and $\lambda_3 = 8.0$, as originally proposed by Assis *et al.* [6].

To calculate $F(t_i)$ (Eq. (13)), we need to know the frequency of the token t_i for each class c_j, obtained from the training set. The frequency ϕ_{c_j,t_i} is a position of ϕ, that keeps the frequency ϕ_{c_j,t_i} of all tokens for all classes. The variable ϕ is also part of the training model.

Algorithms 1 and 2 summarize the training and classification stages of `MDLText`, respectively, where we consider that the input documents are bag-of-words representations using the TF-IDF weighting scheme.

As shown in Algorithms 1 and 2, the classification model is composed by the variables *dict*, n, $\hat{n}$, ϕ, and $|\hat{\mathcal{D}}|$. The variable *dict* keeps the list of the tokens that appeared in the training documents, n is the sum of the weights of each token in *dict* for each class in $\mathcal{C}$, $\hat{n}$ is the sum of the weights in n for each class in $\mathcal{C}$, ϕ is the frequency of each token in *dict* for each class in $\mathcal{C}$, and $|\hat{\mathcal{D}}|$ is the number of documents for each class in $\mathcal{C}$. These variables compose the classification model but are optional inputs in training stage (Algorithm 1) because our approach is naturally adapted to perform online and incremental learning. Therefore, if the variables of the classification model are given as input in Algorithm 1, they are updated with the information obtained with new training documents.

The source code for the `MDLText` as well as all the text collections used in our experiments are publicly available via `GitHub` at https://goo.gl/8FxFkT.

Algorithm 1 Training stage for `MDLText`.

1: **function** MDL_TRAIN($\mathcal{D}, \mathcal{C}, dict, n, \hat{n}, \phi, |\hat{\mathcal{D}}|$)
2: **Input:** $\mathcal{D}$ (training set),$\mathcal{C}$ (set of all possible classes),*dict* (list of tokens),n (sum of the weights of each token in *dict* for each class in $\mathcal{C}$),$\hat{n}$ (sum of the weights in n for each class in $\mathcal{C}$),ϕ (frequency of each token in *dict* for each class in $\mathcal{C}$), and$|\hat{\mathcal{D}}|$ (number of documents for each class in $\mathcal{C}$). Parameters *dict*, n, $\hat{n}$, ϕ, and $|\hat{\mathcal{D}}|$ are optional.
3: **Output:** *dict*, n, $\hat{n}$, ϕ, **and** $|\hat{\mathcal{D}}|$.
4: **if** *dict*, n, $\hat{n}$, ϕ, and $|\hat{\mathcal{D}}|$ were not given **then**
5: $dict \leftarrow \emptyset$ ▷ Create an empty dictionary.
6: $|\hat{\mathcal{D}}_{\forall_c}| \leftarrow 0$ ▷ Initialize the number of documents for each class in C.
7: **end if**
8: **for each** document d in $\mathcal{D}$ **do**
9: $c \leftarrow classLabel(d)$
10: $|\hat{\mathcal{D}}_c| \leftarrow |\hat{\mathcal{D}}_c| + 1$ ▷ Number of documents in class c.
11: **for each** token t_i in d **do**
12: **if** t_i is not in *dict* **then**
13: $dict \leftarrow dict + t_i$ ▷ Insert the token t_i into the dictionary.
14: $n_{\forall_{c,t_i}} \leftarrow 0$ ▷ Initialize the sum of the weights of t_i for each class in $\mathcal{C}$.
15: $\hat{n}_{\forall_c} \leftarrow 0$ ▷ Initialize the sum of the weights in n for each class in $\mathcal{C}$.
16: $\phi_{\forall_{c,t_i}} \leftarrow 0$ ▷ Initialize the frequency of t_i for each class in $\mathcal{C}$.
17: **end if**
18: $n_{c,t_i} \leftarrow n_{c,t_i} + \hat{w}(t_i, d)$ ▷ Sum of the weights of t_i in class c (n_{c,t_i} is a position of n).
19: $\hat{n}_c \leftarrow \hat{n}_c + n_{c,t_i}$ ▷ Sum of the weights in n for the class c ($\hat{n}_c$ is a position of $\hat{n}$).
20: $\phi_{c,t_i} \leftarrow \phi_{c,t_i} + 1$ ▷ Frequency of token t_i in class c (ϕ_{c,t_i} is a position of ϕ).
21: **end for**
22: **end for**
23: **return** *dict*, n, $\hat{n}$, ϕ, and $|\hat{\mathcal{D}}|$
24: **end function**

4.1. Computational complexity

In training step (Algorithm 1), each token of each document leads to incrementing exactly one count. Therefore, with $|D|$ documents and $|\bar{T}|$ tokens, where $|\bar{T}|$ is the average number of tokens, the training complexity is of linear order of $\mathcal{O}(|D||\bar{T}|)$.

In classification step (Algorithm 2), for an unlabeled document d, the `MDLText` needs to compute $K(t_i)$ by examining the frequency of each token in d with respect to each class. Next, the value of each token in each class is used to compute the description length of d. Finally, the value of each token in each class are also used to compute $\hat{S}(d, c_j)$ and to penalize the description length of d. Consequently, the computational complexity is of order of $\mathcal{O}(|C||\bar{T}| + |C||\bar{T}| + |C||\bar{T}|)$ or $\mathcal{O}(3|C||\bar{T}|)$, where $|C|$ is the number of classes. By ignoring the constant and $|C|$, since it has only linear impact on time complexity and it is much smaller than $|\bar{T}|$, the time complexity of the classification step is asymptotically of linear order of $\mathcal{O}(|\bar{T}|)$.

Algorithm 2 Classification stage for MDLText.

function MDL_CLASSIFY($d, \mathcal{C}, dict, n, \hat{n}, \phi$, and $|\hat{\mathcal{D}}|$)
Input: d (unlabeled document),$\mathcal{C}$ (set of all possible classes),$dict$ (list of tokens obtained in the training stage),n (sum of the weights of each token in $dict$ for each class in $\mathcal{C}$),$\hat{n}$ (sum of the weights in n for each class in $\mathcal{C}$),ϕ (frequency of each token in $dict$ for each class in $\mathcal{C}$), and$|\hat{\mathcal{D}}|$ (number of documents for each class in $\mathcal{C}$).
Output: $c(d)$ (the class predicted for document d).

for each token t_i in d **do**
  if t_i is in $dict$ **then**
    $F(t_i) \leftarrow token\_score(t_i, \phi_{\forall_{c,t_i}})$
  else
    $F(t_i) \leftarrow 0$
  end if
  Based on $F(t_i)$, use Eq. 12 to compute $K(t_i)$
end for
for each class c_j in $\mathcal{C}$ **do**
  $L(d|c_j) \leftarrow 0$
  for each token t_i in d **do**
    if t_i is not in $dict$ **then**
      $n_{c_j,t_i} \leftarrow 0$
      $\hat{n}_{c_j} \leftarrow 0$
    end if
    Based on n_{c_j,t_i} and $\hat{n}_{c_j}$, use Eq. 7 to compute $\beta(t_i|c_j)$
    Based on $\beta(t_i|c_j)$ use Eq. 4 to compute $L(t_i, c_j)$
    $L(d|c_j) \leftarrow L(d|c_j) + (L(t_i, c_j) \times K(t_i))$ ▷ Accumulate the sum of the description length of the tokens penalized by their scores. It is a piece of the Eq. 8.
    Based on n_{c_j,t_i} and $|\hat{\mathcal{D}}_{c_j}|$ use Eq. 11 to compute $\bar{c}_j(t_i)$ (it is a position of the prototype vector $\bar{c}_j$)
  end for
end for
for each class c_j in $\mathcal{C}$ **do**
  Calculate the similarity $S(d, \bar{c}_j)$ between d and $\bar{c}_j$ using Eq. 10
  Based on $S(d, \bar{c}_j)$ use Eq. 9 to compute $\hat{S}(d, c_j)$
  $L(d|c_j) \leftarrow \hat{S}(d, c_j) \times L(d|c_j))$ ▷ It is the complement piece of the Eq. 8 started on line 21.
end for
$c(d) = \arg\min_{\forall c} L(d|c_j)$ ▷ The class label with the minimum overall description length.
return $c(d)$
end function

5. Experimental setup

We carefully performed a comprehensive evaluation to assess the performance of the proposed MDLText, where we used a large number of well-known and public benchmark corpora, which comprised 45 datasets [57,72]. Table 1 summarizes the main statistics for each of the datasets used in this study. The amount of samples (text documents) is represented by $|\mathcal{D}|$, $|\vec{d}|$ corresponds to the number of features (vocabulary) and $|\mathcal{C}|$ is the number of classes. The last column presents the amount of documents in each class.

To ensure the credibility of the results obtained and to make our experiments completely reproducible, we describe all of the steps in the following, including preprocessing, parameter settings, training, classification, and evaluation.

5.1. Preprocessing and tokenization

The documents in *7sectors* comprised HTML pages, so we parsed them using the Beautiful Soup Library[1] available to Python.

For *TechTC-300*, we used the Exp-1092-135724 and Exp-1092-789236 datasets.

For *Reuters-21578*, we removed documents without labels and those labeled with more than one topic. We then selected documents from the top 10 most frequent topics.

The *RCV1* and *RCV2* collections contained documents that belong to three categories: industry, region, and codes. We used documents based on their topic codes, which were organized into four hierarchical groups: CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social) and MCAT (Markets). We removed documents without codes and those assigned to more than one hierarchical group. Moreover, for *RCV1*, we used documents published between August 20, 1996 and September 4, 1996.

For the *7sectors*, *TechTC-300*, *Reuters-21578*, *RCV1* and *RCV2* collections, we used lowercase conversion. Furthermore, we applied stemming and stop-words removal using the NLTk Library[2] for Python. For tokenization, we used non-alphanumeric characters as delimiters.

The remaining datasets were described and preprocessed by [57,58]. They were provided by the authors as bag-of-words models, where stemming was applied and the stop-words were removed.

5.2. Evaluations

We tested MDLText in two traditional tasks: batch learning (or offline learning) and online learning.

In *batch learning*, all of the training examples are presented to the classifier and a global learning model is created to make predictions for unseen data. The batch learning models are usually more powerful, but they are static.

By contrast, the examples are presented one at time in *online learning*. Therefore, online learning classifiers are appropriate for large-scale problems since there is typically no need to store the examples in memory.

The goal of these experiments was to assess the overall performance of the MDLText, so we compared it with several benchmark and state-of-the-art text classification methods in both scenarios.

5.2.1. Performance measures

To compare the results, we employed the well-known and established macro-average F-measure [63]:

$$\textbf{F-measure}_{\textbf{macro}} = 2 \times \frac{\text{Precision}_{\text{macro}} \times \text{Recall}_{\text{macro}}}{\text{Precision}_{\text{macro}} + \text{Recall}_{\text{macro}}}, \tag{14}$$

where $\text{Recall}_{\text{macro}} = \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j}{TP_j+FN_j}$ and $\text{Precision}_{\text{macro}} = \frac{1}{m}\sum_{j=1}^{m}\frac{TP_j}{TP_j+FP_j}$, where m, TP, FP, and FN are the numbers of classes, true positives, false positives, and false negatives, respectively.

5.3. Batch learning methods

We compared the results obtained by MDLText (Eq. (8)) and traditional MDL (Eq. (3)) with those produced using the following established batch learning methods: multinomial naïve Bayes (M.NB) [44], Bernoulli naïve Bayes (B.NB) [44], Rocchio [56], SVM [9,14], KNN [15], DT [13,50] and random forests (RF) [12]. These methods are widely used as baseline approaches for text classification.

We used the implementations of M.NB, B.NB, Rocchio, KNN, DT, and RF from the scikit-learn Library[3]. MDL and MDLText

[1] The Beautiful Soup Library is available at http://goo.gl/Yppldp.

[2] The NLTk Library is available at http://www.nltk.org/.

[3] The scikit-learn Library is available at http://scikit-learn.org/.

Table 1
Datasets used in our experiments.

Dataset	$\|\mathcal{D}\|$	$\|\vec{d}\|$	Sparsity	$\|C\|$	Class size
20Newsgroups	18,828	45,433	99.8%	20	628, 775, 799, 910, 940, 961, 972, 973, 980, 981, 982, 985, 987, 990, 990, 991, 994, 994, 997, 999
7Sectors	4581	41,718	99.7%	7	300, 355, 399, 515, 949, 964, 1099
ACM	3493	60,767	98.8%	40	50, 69, 70, 71, 71, 71, 71, 72, 74, 75, 80, 81, 82, 82, 83, 84, 85, 86, 87, 89, 90, 91, 91, 92, 93, 93, 95, 96, 96, 98, 98, 98, 101, 102, 103, 104, 104, 104, 105, 106
CSTR	299	1725	96.9%	4	25, 46, 100, 128
Dmoz-Business	18,500	8302	99.9%	37	500 (each class)
Dmoz-Computers	9500	5010	99.8%	19	500 (each class)
Dmoz-Health	6500	4216	99.7%	13	500 (each class)
Dmoz-Science	6000	4820	99.8%	12	500 (each class)
Dmoz-Sports	13,500	5681	99.8%	27	500 (each class)
Enron	13,199	18,193	99.7%	20	309, 339, 367, 370, 379, 397, 407, 420, 489, 526, 609, 657, 715, 715, 897, 1022, 1108, 1122, 1159, 1192
FBIS	2463	2000	92.0%	17	38, 43, 46, 46, 46, 48, 65, 92, 94, 119, 121, 125, 139, 190, 358, 387, 506
Industry-Sector	8817	21,489	99.6%	12	100, 282, 354, 397, 495, 505, 557, 635, 947, 959, 991, 2595
Irish economic	1660	8658	98.7%	3	431, 574, 655
La1s	3204	13,195	98.9%	6	273, 341, 354, 555, 738, 943
La2s	3075	12,432	98.8%	6	248, 301, 375, 487, 759, 905
Latimes	6279	10,019	99.6%	6	521, 642, 729, 1042, 1497, 1848
New3s	9558	26,832	99.1%	44	104, 105, 106, 110, 110, 115, 116, 120, 123, 124, 126, 130, 136, 139, 139, 141, 153, 159, 161, 171, 171, 174, 179, 181, 187, 196, 198, 211, 218, 238, 243, 253, 270, 276, 278, 281, 306, 326, 328, 330, 369, 493, 568, 696
NFS	10,524	3887	99.8%	16	130, 201, 307, 345, 355, 402, 442, 524, 603, 647, 739, 889, 990, 1202, 1339, 1409
Oh0	1003	3182	98.3%	10	51, 56, 57, 66, 71, 76, 115, 136, 181, 194
Oh10	1050	3238	98.3%	10	52, 60, 61, 70, 87, 116, 126, 148, 165, 165
Oh15	913	3100	98.1%	10	53, 56, 56, 66, 69, 98, 98, 106, 154, 157
Oh5	918	3012	98.2%	10	59, 61, 61, 72, 74, 85, 93, 120, 144, 149
Ohscal	11,162	11,465	99.5%	10	709, 764, 864, 1001, 1037, 1159, 1260, 1297, 1450, 1621
Opinosis	6457	2692	99.7%	51	45, 45, 48, 51, 52, 56, 57, 59, 59, 60, 62, 64, 64, 67, 73, 76, 79, 80, 82, 86, 87, 88, 89, 90, 92, 96, 103, 104, 110, 111, 112, 116, 119, 125, 130, 142, 145, 153, 155, 157, 159, 166, 171, 181, 191, 241, 266, 284, 304, 377, 528
Pubmed-Cancer	65,991	28,328	99.7%	12	104, 433, 557, 926, 2220, 2407, 2848, 6676, 7390, 9177, 15603, 17,650
Pubmed-Cancer-2000	18,355	14,424	99.6%	12	139, 483, 631, 1102, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000
RCV1	23,983	60,355	99.8%	4	2032, 5746, 6262, 9943
RCV2-Italian	13,555	18,937	99.7%	4	281, 1581, 3827, 7866
RCV2-Portuguese	3974	9341	99.1%	4	50, 129, 842, 2953
RCV2-Spanish	12,388	19,472	99.7%	4	30, 1038, 1324, 9996
Re0	1504	2886	98.2%	13	11, 15, 16, 20, 37, 38, 39, 42, 60, 80, 219, 319, 608
Re1	1657	3758	98.6%	25	10, 13, 15, 17, 18, 18, 19, 19, 20, 20, 27, 31, 31, 32, 37, 42, 48, 50, 60, 87, 99, 106, 137, 330, 371
Re8	7674	8900	99.6%	8	51, 144, 271, 293, 326, 374, 2292, 3923
Reuters	8246	20,529	99.7%	10	116, 143, 158, 161, 285, 307, 361, 408, 2362, 3945
Reviews	4069	22,926	99.2%	5	137, 412, 999, 1133, 1388
TechTC300-1092-135724	1008	30,903	99.4%	2	499, 509
TechTC300-1092-789236	1097	31,623	99.3%	2	509, 588
Tr11	414	6429	95.6%	9	6, 11, 20, 21, 29, 52, 69, 74, 132
Tr12	313	5804	95.3%	8	9, 29, 29, 30, 34, 35, 54, 93
Tr21	336	7902	94.1%	6	4, 9, 16, 35, 41, 231
Tr23	204	5832	93.4%	6	6, 11, 15, 36, 45, 91
Tr31	927	10,128	97.3%	7	2, 21, 63, 111, 151, 227, 352
Tr41	878	7454	97.4%	10	9, 18, 26, 33, 35, 83, 95, 162, 174, 243
Tr45	690	8261	96.6%	10	14, 18, 36, 47, 63, 67, 75, 82, 128, 160
Trec7-3000	6000	100,463	99.8%	2	3000 (each class)

were fully implemented in C++. We conducted experiments with linear SVM using the `LibSVM Library` for C++. The experiments were performed with a linear kernel, because [28] demonstrated that a linear kernel performs better than other kernel functions with high-dimensional problems.

The performance of some text classification methods can be affected greatly by the term weighting scheme used to represent the documents. Therefore, for B.NB, DT, M.NB, KNN, RF, and SVM, we performed grid search using 5-fold cross validation to find the best term weighting scheme, i.e., binary, TF, or normalized TF-IDF. For MDL, `MDLText`, and Rocchio, we only used the TF-IDF text representation, whereas for B.NB, we used the binary text representation, because these weighting schemes are intrinsic to these methods.

The performance of KNN, RF, SVM, MDL, and `MDLText` can also be affected greatly by the choice of parameters, so we also performed grid search to find the best values for their main parameters. For the other methods evaluated in this study, we used their default values.

In these steps, we tried to ensure that for each dataset, the best text representation and parameters were selected for use by each method. Thus, for each dataset, we guarantee that just the best overall performance for each evaluated approach was compared each other.

Table 2 presents the values obtained by grid search, which were used in our experiments, where K corresponds to the number of neighbors for KNN, T is the number of trees used in RF, and C is the parameter cost for SVM.

5.4. Online learning methods

We also compared the performance obtained by the proposed `MDLText` (Eq. (8)) and MDL (Eq. (3)) classifiers with the following well-known online learning methods: M.NB [44], B.NB [44], perceptron [21], and stochastic gradient descent (SGD-SVM) [73]. We

Table 2
Parameters obtained by grid search and used in the batch learning experiments.

	DT	KNN		M.NB	MDL	MDLText	RF		SVM	
	Weight	Weight	K	Weight	$\|\Omega\|$	$\|\Omega\|$	Weight	T	Weight	C
20Newsgroups	TF	TF-IDF	5	TF	2^{6}	2^{5}	Binary	100	TF-IDF	2^{1}
7Sectors	Binary	TF-IDF	5	TF	2^{10}	2^{17}	Binary	100	TF-IDF	2^{1}
ACM	TF	TF-IDF	15	TF	2^{11}	2^{6}	TF-IDF	100	Binary	2^{-2}
CSTR	TF	TF-IDF	25	TF	2^{7}	2^{7}	Binary	50	TF-IDF	2^{6}
Dmoz-Business	Binary	TF-IDF	30	TF-IDF	2^{5}	2^{5}	TF-IDF	100	TF-IDF	2^{0}
Dmoz-Computers	Binary	TF-IDF	30	TF-IDF	2^{5}	2^{5}	TF-IDF	90	TF-IDF	2^{0}
Dmoz-Health	Binary	TF-IDF	20	TF	2^{6}	2^{5}	TF-IDF	90	TF-IDF	2^{0}
Dmoz-Science	Binary	TF-IDF	25	TF-IDF	2^{5}	2^{5}	TF-IDF	100	TF-IDF	2^{0}
Dmoz-Sports	TF	TF-IDF	50	TF-IDF	2^{5}	2^{5}	TF-IDF	80	TF-IDF	2^{1}
Enron	TF-IDF	TF-IDF	25	TF	2^{7}	2^{7}	TF-IDF	100	TF-IDF	2^{0}
Fbis	TF	TF-IDF	10	TF	2^{9}	2^{9}	TF-IDF	90	TF-IDF	2^{1}
Industry-Sector	Binary	TF-IDF	25	TF	2^{12}	2^{15}	TF	80	TF-IDF	2^{2}
Irish Economic	Binary	TF-IDF	25	TF	2^{11}	2^{10}	TF-IDF	100	TF-IDF	2^{0}
La1S	TF-IDF	TF-IDF	10	Binary	2^{7}	2^{8}	TF-IDF	100	TF-IDF	2^{4}
La2S	TF-IDF	TF-IDF	15	TF	2^{6}	2^{7}	Binary	80	TF-IDF	2^{13}
Latimes	Binary	TF-IDF	10	Binary	2^{5}	2^{9}	TF-IDF	70	TF-IDF	2^{0}
NFS	TF	TF-IDF	15	Binary	2^{7}	2^{6}	TF-IDF	100	TF-IDF	2^{0}
New3S	TF-IDF	TF-IDF	5	TF	2^{8}	2^{7}	TF-IDF	90	TF-IDF	2^{1}
Oh0	TF-IDF	TF-IDF	25	TF	2^{8}	2^{5}	Binary	80	TF-IDF	2^{15}
Oh10	Binary	TF-IDF	30	Binary	2^{6}	2^{5}	Binary	90	TF-IDF	2^{5}
Oh15	TF	TF-IDF	45	TF	2^{6}	2^{6}	Binary	100	TF-IDF	2^{2}
Oh5	Binary	TF-IDF	35	TF	2^{6}	2^{6}	Binary	100	TF-IDF	2^{11}
Ohscal	TF	TF-IDF	50	TF	2^{5}	2^{5}	TF	100	TF-IDF	2^{-1}
Opinosis	TF	TF-IDF	50	TF	2^{6}	2^{5}	TF-IDF	70	TF	2^{-2}
Pubmed-Cancer	TF	TF-IDF	45	TF	2^{10}	2^{7}	TF-IDF	80	TF-IDF	2^{0}
Pubmed-Cancer-2000	Binary	TF	10	TF	2^{8}	2^{7}	TF	90	Binary	2^{-5}
RCV1	TF-IDF	TF-IDF	5	TF-IDF	2^{11}	2^{9}	TF-IDF	100	TF-IDF	2^{4}
Rcv2-Italian	TF	TF-IDF	15	TF	2^{25}	2^{22}	TF-IDF	90	TF-IDF	2^{1}
Rcv2-Portuguese	TF-IDF	TF-IDF	5	TF	2^{13}	2^{11}	TF	40	TF-IDF	2^{2}
Rcv2-Spanish	TF-IDF	TF-IDF	10	TF	2^{10}	2^{16}	TF	40	TF	2^{-3}
Re0	TF-IDF	TF-IDF	30	TF	2^{8}	2^{6}	TF-IDF	60	TF-IDF	2^{0}
Re1	TF	TF-IDF	10	TF	2^{8}	2^{8}	TF-IDF	100	TF-IDF	2^{5}
Re8	TF-IDF	TF-IDF	50	TF	2^{7}	2^{6}	TF-IDF	100	TF-IDF	2^{9}
Reuters	TF-IDF	TF-IDF	35	TF	2^{7}	2^{6}	TF-IDF	80	TF-IDF	2^{1}
Reviews	TF	TF-IDF	50	TF	2^{8}	2^{9}	TF-IDF	70	TF-IDF	2^{3}
Techtc300-1092-135724	TF-IDF	TF-IDF	5	TF	2^{15}	2^{21}	Binary	90	TF-IDF	2^{3}
Techtc300-1092-789236	Binary	TF-IDF	5	TF-IDF	2^{22}	2^{14}	TF	50	TF-IDF	2^{3}
Tr11	Binary	TF-IDF	15	TF	2^{9}	2^{7}	Binary	60	Binary	2^{3}
Tr12	Binary	TF-IDF	10	TF	2^{7}	2^{6}	TF-IDF	70	TF	2^{5}
Tr21	TF-IDF	TF-IDF	5	TF	2^{9}	2^{9}	Binary	10	TF	2^{7}
Tr23	Binary	TF-IDF	5	TF	2^{8}	2^{7}	TF-IDF	50	TF-IDF	2^{6}
Tr31	TF	TF-IDF	25	TF	2^{11}	2^{11}	TF-IDF	50	TF-IDF	2^{8}
Tr41	TF	TF-IDF	10	TF	2^{9}	2^{7}	TF-IDF	90	TF-IDF	2^{7}
Tr45	Binary	TF-IDF	5	TF	2^{8}	2^{9}	Binary	90	TF-IDF	2^{9}
Trec7-3000	Binary	TF-IDF	45	TF	2^{24}	2^{24}	TF-IDF	90	TF-IDF	2^{0}

implement the SGD-SVM classifier with the standard SVM loss (the hinge loss) and L2 regularization, which is similar to a linear SVM that can be updated online [11,27,73].

We implemented M.NB, B.NB, perceptron and SGD-SVM in `Python` using the `scikit-learn Library`.

We also performed a grid search to find the best term weighting scheme (Table 3). For MDL and `MDLText`, we used the same value for $|\Omega|$ shown in Table 2 and the TF-IDF weight scheme in all of the experiments. For B.NB, we have used the binary weight scheme.

All of the tests were performed using a PC with an Intel Core i7 2.93 GHz CPU, 8 GB RAM and Ubuntu 14.04.

6. Experiments and results

In the following, we describe the results obtained in our experiments using batch and online learning tasks.

6.1. Batch learning task

Table 4 shows the results obtained by each evaluated method for each of 45 datasets. We present the average macro F-measure obtained in 10 runs of stratified 5-fold cross-validation. The bold values indicate the best scores.

It is clear that SVM obtained the highest scores for most of the datasets, but `MDLText` outperformed all of the other benchmark methods with 25 datasets. Furthermore, `MDLText` performed much better than traditional MDL and it is more consistent across the datasets as the second best classifier.

The differences between the performance of MDL and `MDLText` were higher for datasets with highly imbalanced classes. For instance, for Pubmed-cancer, Pubmed-Cancer-2000, Opinosis, Tr21, and Re1, the absolute differences between the macro F-measure achieved by these two methods were 0.234, 0.189, 0.157, 0.144, and 0.144, respectively. To check if such differences are due to the fact that MDL is failing on some of the rarest classes, we compared the average micro F-measure [63]. In fact, the absolute differences between the micro F-measure obtained by MDL and `MDLText` were 0.151, 0.165, 0.131, 0.065, and 0.088, respectively. Given that the micro F-measure tends to be dominated by the majority classes, such differences indicate that MDL was more negatively affected by the imbalance of classes than `MDLText`.

To ensure that the results were not obtained by chance, we performed a statistical analysis using the nonparametric Friedman

Table 3
Parameters obtained by grid search in the online learning experiments.

	M.NB Weight	Perceptron Weight	SGD–SVM Weight
20Newsgroups	TF-IDF	TF-IDF	TF-IDF
7Sectors	TF	TF-IDF	TF-IDF
ACM	TF-IDF	TF-IDF	TF-IDF
CSTR	TF	TF-IDF	TF-IDF
Dmoz-Business	TF-IDF	TF-IDF	TF-IDF
Dmoz-Computers	TF-IDF	TF-IDF	TF-IDF
Dmoz-Health	TF-IDF	TF-IDF	TF-IDF
Dmoz-Science	TF-IDF	TF-IDF	TF-IDF
Dmoz-Sports	TF-IDF	TF-IDF	TF-IDF
Enron	TF	TF-IDF	TF-IDF
Fbis	TF	TF-IDF	TF-IDF
Industry-Sector	TF	TF-IDF	TF-IDF
Irish economic	TF-IDF	TF-IDF	TF-IDF
La1S	TF	TF-IDF	TF-IDF
La2S	TF	TF-IDF	TF-IDF
Latimes	TF	TF-IDF	TF-IDF
NFS	TF	TF-IDF	TF-IDF
New3S	TF	TF-IDF	TF-IDF
Oh0	TF	TF-IDF	TF-IDF
Oh10	TF	TF-IDF	TF-IDF
Oh15	TF	TF-IDF	TF-IDF
Oh5	TF	TF-IDF	TF-IDF
Ohscal	TF-IDF	TF-IDF	TF-IDF
Opinosis	TF	TF	TF-IDF
Pubmed-Cancer	TF	TF-IDF	TF-IDF
Pubmed-Cancer-2000	TF	TF-IDF	TF-IDF
RCV1	TF	TF-IDF	TF-IDF
Rcv2-Italian	TF	TF-IDF	TF-IDF
Rcv2-Portuguese	TF	TF-IDF	TF-IDF
Rcv2-Spanish	TF	TF-IDF	TF-IDF
Re0	TF	TF-IDF	TF-IDF
Re1	TF	TF-IDF	TF-IDF
Re8	TF	TF-IDF	TF-IDF
Reuters	TF	TF-IDF	TF-IDF
Reviews	TF	TF	TF-IDF
Techtc300-1092-135724	TF-IDF	Binary	TF-IDF
Techtc300-1092-789236	TF	Binary	TF-IDF
Tr11	TF	TF-IDF	TF-IDF
Tr12	TF	TF-IDF	TF-IDF
Tr21	TF	TF-IDF	TF-IDF
Tr23	TF	TF-IDF	TF-IDF
Tr31	TF	TF-IDF	TF-IDF
Tr41	TF	TF-IDF	TF-IDF
Tr45	TF	TF-IDF	TF-IDF
Trec7-3000	TF-IDF	TF-IDF	TF-IDF

test, where carefully followed the methodology described in [17]. For each evaluated method, Fig. 5 presents the maximum, minimum, and average ranking according to the macro F-measure, where a lower average ranking indicates better performance.

Let k be the amount of approaches evaluated and n be the amount of datasets. The Friedman test checks whether the null hypothesis, which states that all of algorithms compared are equivalent, can be rejected. The null hypothesis might be rejected if the critical value in χ^2 distribution, with $k-1$ degrees of freedom, is lower than χ_F^2, for n and k sufficiently large [17]. For a confidence interval $\alpha = 0.05$, $k = 9$, and $n = 45$, the critical value was 2.730. Therefore, the null hypothesis could be rejected as $\chi_F^2 = 215.621$.

Since the null hypothesis was rejected, we performed a pairwise comparison using a post-hoc test. As recommended by [17], we used the Bonferroni–Dunn test, which states that the performance of two algorithms differs significantly if the difference between their ranks is greater than or equal to a critical difference *CD*. For a confidence interval $\alpha = 0.05$ and $k = 9$, *CD* was equal to 1.579. Therefore, there was not enough statistical evidence to state that the performance obtained by SVM was significantly better than that of `MDLText`. However, we had sufficient statistical evidence to safely state that the performance obtained by `MDLText` was significantly better than that of MDL, naïve Bayes classifiers, k-NN, DT, RF, and Rocchio.

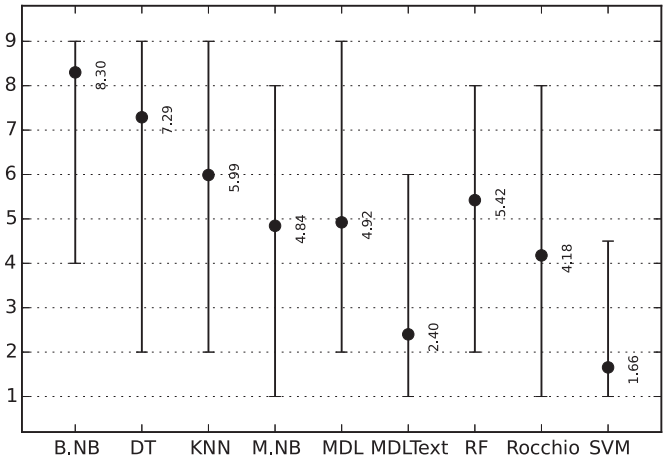


Fig. 5. Average ranking of methods based on their position with each dataset.

Despite SVM obtained the best results for most of the datasets, the proposed `MDLText` has features that favor its application in real-world scenarios and large-scale text classification problems, including its fast training and incremental learning.

Fig. 6 compares the average computational time required by `MDLText` and SVM for training and classifying the ten largest text collections. We note that on average SVM was 56 times slower than `MDLText`.

In addition, in online machine learning problems, as the traditional SVM does not support incremental learning, so the learning stage must be performed completely from time to time, which will become increasingly computational costly as the amount of samples and features increase. By contrast, the proposed `MDLText` is naturally multi-class, lightweight and it offer incremental learning, which allows its application to real-world, online, and dynamic scenarios.

6.2. Online learning task

In this experiment, we considered the following scenario: a small number of samples were available to train the classifiers (20% of the samples). Next, one sample was presented at time to the classifiers, which made their predictions. Then, the classifiers received feedback and if their prediction was wrong, the training model was updated with the true label. The number of updates was limited to 40% of the samples. In this way, we aim to reproduce a real-world and dynamic scenario instead of a static one.

Table 5 shows the results obtained by each evaluated method for each of the 45 datasets. We present the average macro F-measure obtained by each classifier in 10 runs. The bold values indicate the best scores.

For most of the datasets (35 out of 45), `MDLText` outperformed MDL, B.NB, M.NB, perceptron, and SGD-SVM. Moreover, `MDLText` performed better than MDL with all of the datasets.

We also performed a statistical analysis based on the average ranking of each classification method (Fig. 7). According to the nonparametric Friedman test, with $\alpha = 0.05$, $k = 6$, and $n = 45$, the critical value was 1.145. Given that $\chi_F^2 = 143.933$, we could reject the null hypothesis that all of the algorithms compared in this experiment are equivalent. According to the pairwise comparison using the Bonferroni–Dunn post-hoc test, with $\alpha = 0.05$, and $k = 6$, *CD* was equal to 1.016, and, thus we can safely state that the performance of `MDLText` was significantly better than any other classifier evaluated.

Table 4
Average macro F-measure obtained by each method for each dataset in 10 runs of stratified 5-fold cross-validation.

	B.NB	DT	KNN	M.NB	MDL	MDLText	RF	Rocchio	SVM
20Newsgroups	0.812	0.688	0.865	0.895	0.914	0.902	0.859	0.869	**0.930**
7Sectors	0.545	0.805	0.900	0.857	0.890	0.913	0.895	0.886	**0.957**
ACM	0.653	0.642	0.777	0.771	0.829	0.822	0.842	0.812	**0.883**
CSTR	0.718	0.674	0.852	**0.893**	0.860	0.892	0.773	0.886	0.860
Dmoz-Business	0.665	0.484	0.669	0.697	0.651	0.687	0.626	0.650	**0.712**
Dmoz-Computers	0.683	0.527	0.667	0.707	0.666	0.704	0.650	0.669	**0.723**
Dmoz-Health	0.797	0.712	0.784	0.819	0.785	0.826	0.799	0.792	**0.858**
Dmoz-Science	0.694	0.556	0.669	0.738	0.696	0.737	0.672	0.702	**0.747**
Dmoz-Sports	0.831	0.815	0.838	0.850	0.807	0.883	0.881	0.855	**0.906**
Enron	0.524	0.580	0.693	0.707	0.723	0.716	0.682	0.690	**0.739**
Fbis	0.605	0.653	0.725	0.745	0.749	0.790	0.767	0.785	**0.855**
Industry-Sector	0.457	0.502	0.710	0.744	0.779	0.811	0.702	0.748	**0.901**
Irish economic	0.657	0.477	0.622	0.668	0.651	0.672	0.622	0.667	**0.693**
La1S	0.761	0.719	0.841	0.861	0.878	0.877	0.833	0.870	**0.898**
La2S	0.777	0.743	0.864	0.877	0.876	0.883	0.855	0.872	**0.906**
Latimes	0.793	0.698	0.818	0.830	0.841	0.847	0.788	0.823	**0.856**
NFS	0.656	0.696	0.775	0.828	0.817	0.829	0.782	0.780	**0.843**
New3S	0.476	0.707	0.796	0.801	0.819	0.858	0.857	0.852	**0.910**
Oh0	0.719	0.797	0.861	0.874	0.867	**0.925**	0.892	0.924	0.909
Oh10	0.690	0.763	0.762	0.777	0.775	**0.819**	0.811	0.818	0.816
Oh15	0.691	0.722	0.769	0.825	0.792	0.859	0.843	**0.861**	0.858
Oh5	0.728	0.831	0.820	0.863	0.825	0.907	0.893	0.905	**0.913**
Ohscal	0.721	0.676	0.723	0.734	0.721	0.780	0.790	0.779	**0.808**
Opinosis	0.208	0.580	0.592	0.586	0.517	**0.674**	0.611	0.672	0.638
Pubmed-Cancer	0.666	0.935	0.705	0.762	0.686	0.920	0.849	0.903	**0.947**
Pubmed-Cancer-2000	0.647	0.862	0.142	0.729	0.659	0.848	0.850	0.833	**0.874**
RCV1	0.835	0.859	0.921	0.839	0.930	0.920	0.925	0.891	**0.961**
Rcv2-Italian	0.738	0.759	0.843	0.804	0.815	0.794	0.819	0.771	**0.875**
Rcv2-Portuguese	0.670	0.671	0.811	0.795	0.789	0.791	0.728	0.779	**0.829**
Rcv2-Spanish	0.678	0.742	0.827	0.717	0.841	0.841	0.841	0.793	**0.850**
Re0	0.289	0.648	0.692	0.699	0.742	0.815	0.713	0.815	**0.842**
Re1	0.248	0.717	0.700	0.696	0.668	**0.812**	0.668	**0.812**	0.804
Re8	0.606	0.827	0.871	0.903	0.876	0.914	0.861	0.882	**0.936**
Reuters	0.463	0.855	0.883	0.904	0.877	0.912	0.883	0.872	**0.951**
Reviews	0.770	0.839	0.900	0.905	0.898	0.903	0.889	0.894	**0.935**
Techtc300-1092-135724	0.897	0.952	0.954	0.963	0.977	**0.978**	0.953	0.968	0.976
Techtc300-1092-789236	0.891	0.973	0.969	0.978	0.982	**0.985**	0.969	0.962	0.979
Tr11	0.342	0.673	0.598	0.701	0.727	**0.827**	0.661	0.802	0.770
Tr12	0.492	0.767	0.734	0.814	0.840	0.879	0.804	**0.882**	0.841
Tr21	0.346	0.713	0.702	0.575	0.692	0.836	0.445	**0.873**	0.704
Tr23	0.282	0.867	0.736	0.814	0.837	0.921	0.543	**0.925**	0.853
Tr31	0.522	0.782	0.771	0.792	0.800	0.826	0.814	0.822	**0.836**
Tr41	0.546	0.870	0.908	0.891	0.894	0.924	0.802	0.912	**0.925**
Tr45	0.615	0.822	0.817	0.812	0.848	**0.903**	0.828	0.902	0.887
Trec7-3000	0.698	0.973	0.984	0.971	0.975	0.981	0.990	0.955	**0.994**

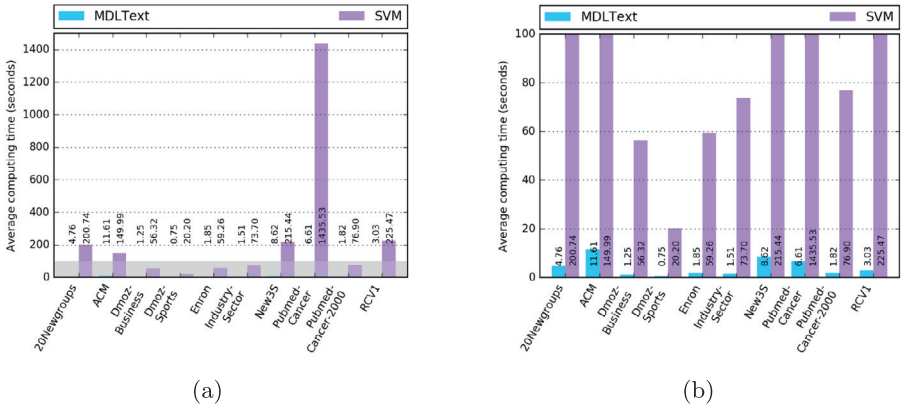


Fig. 6. Average computational time (in seconds) required to train and classify all of the samples in the ten largest text collections. Fig. 6b shows the gray region from Fig. 6a.

7. Conclusions and future work

In this study, we proposed a novel multinomial text classification technique based on the MDL principle. The proposed method has desirable features, such as incremental learning, low computational cost, and sufficient robustness to prevent overfitting, thereby leading to high predictive power and efficiency, which allow its application to real-world, online, and large-scale text classification problems.

To assess its performance, we conducted a comprehensive evaluation using 45 large, real, public, and well-known text corpora, where we considered two different scenarios: batch learning and

Table 5
Average macro F-measure obtained by each method with each dataset in 10 runs.

	B.NB	M.NB	MDL	MDLText	Perceptron	SGD-SVM
20Newsgroups	0.802	0.883	0.879	**0.889**	0.830	0.879
7Sectors	0.716	0.827	0.850	0.896	0.866	**0.904**
ACM	0.649	0.723	0.787	**0.805**	0.693	0.749
CSTR	0.676	0.844	0.787	**0.849**	0.744	0.790
Dmoz-Business	0.605	0.666	0.608	**0.667**	0.549	0.605
Dmoz-Computers	0.624	0.673	0.624	**0.678**	0.572	0.613
Dmoz-Health	0.766	0.804	0.749	**0.811**	0.737	0.775
Dmoz-Science	0.648	**0.696**	0.642	0.695	0.601	0.638
Dmoz-Sports	0.797	0.846	0.782	**0.872**	0.827	0.863
Enron	0.557	0.692	0.706	**0.717**	0.659	0.694
Fbis	0.605	0.725	0.713	**0.781**	0.703	0.745
Industry-Sector	0.477	0.721	0.731	0.791	0.751	**0.797**
Irish economic	0.639	0.636	0.600	**0.642**	0.579	0.600
La1S	0.808	0.847	0.856	**0.873**	0.801	0.838
La2S	0.816	0.861	0.850	**0.879**	0.821	0.854
Latimes	0.781	0.811	0.805	**0.839**	0.762	0.791
NFS	0.635	0.790	0.763	**0.798**	0.710	0.758
New3S	0.507	0.772	0.798	**0.859**	0.801	0.855
Oh0	0.723	0.831	0.800	**0.888**	0.787	0.834
Oh10	0.677	0.740	0.627	**0.796**	0.685	0.725
Oh15	0.666	0.777	0.683	**0.819**	0.711	0.756
Oh5	0.724	0.825	0.722	**0.874**	0.769	0.823
Ohscal	0.714	0.734	0.706	**0.774**	0.703	0.746
Opinosis	0.222	0.575	0.485	**0.646**	0.497	0.545
Pubmed-Cancer	0.666	0.783	0.697	0.921	0.896	**0.949**
Pubmed-Cancer-2000	0.671	0.741	0.662	**0.840**	0.793	0.828
RCV1	0.850	0.926	0.921	0.923	0.923	**0.946**
Rcv2-Italian	0.721	0.817	0.791	0.823	0.808	**0.834**
Rcv2-Portuguese	0.640	**0.804**	0.780	**0.804**	0.767	0.799
Rcv2-Spanish	0.657	0.726	0.802	**0.839**	0.781	0.824
Re0	0.307	0.711	0.607	**0.785**	0.688	0.741
Re1	0.249	0.645	0.418	**0.762**	0.659	0.712
Re8	0.594	0.889	0.834	0.909	0.865	**0.910**
Reuters	0.464	0.875	0.839	0.914	0.888	**0.928**
Reviews	0.795	0.891	0.881	**0.910**	0.870	0.907
Techtc300-1092-135724	0.937	0.958	0.962	**0.974**	0.944	0.957
Techtc300-1092-789236	0.964	0.974	0.976	**0.982**	0.964	0.969
Tr11	0.449	0.681	0.391	**0.789**	0.703	0.709
Tr12	0.543	0.738	0.453	**0.840**	0.745	0.812
Tr21	0.323	0.653	0.294	0.786	0.754	**0.796**
Tr23	0.313	0.741	0.320	**0.872**	0.782	0.822
Tr31	0.578	0.770	0.407	**0.818**	0.791	0.816
Tr41	0.563	0.867	0.737	**0.923**	0.865	0.903
Tr45	0.651	0.801	0.763	**0.902**	0.844	0.890
Trec7-3000	0.919	0.977	0.981	0.985	0.979	**0.989**

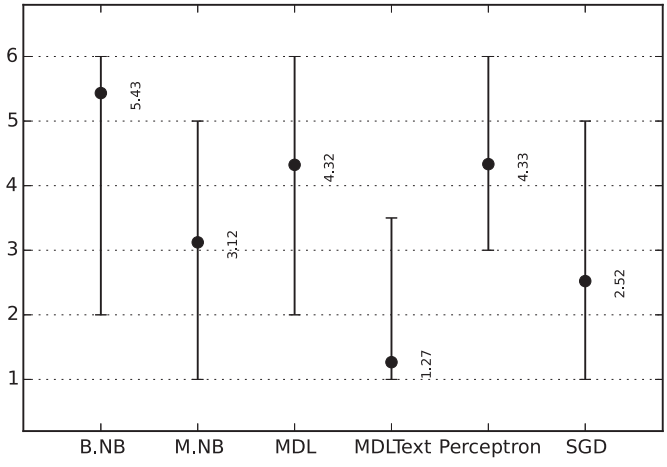


Fig. 7. Average ranking of online learning methods based on their position with each dataset.

online learning. Moreover, we compared the results obtained using the proposed method with those produced by some of the most widely used traditional machine learning algorithms that are currently available. To ensure a fair comparison, we performed an exhaustive grid search to set the best term weighting scheme and parameters for each method and dataset. In this manner, we guaranteed that the results compared represented the best performance that each method could achieve with each dataset.

The results obtained indicated that the proposed MDLText classifier provides a very good balance between predictive power and computational efficiency. In terms of its capacity of prediction, the statistical analysis of the results showed that MDLText outperformed naïve Bayes, Bernoulli naïve Bayes, *k*-nearest neighbors, decision trees, random forests and Rocchio. The same analysis did not indicate a significant difference between the performance achieved by the proposed approach and SVM, although the learning process is much more efficient in terms of speed using the proposed method (56 times faster on average).

In experiments conducted based on an online scenario, we compared MDLText with well-known incremental classifiers: the multinomial naïve Bayes, Bernoulli naïve Bayes, perceptron and stochastic gradient descent. The statistical analysis of the results indicated that we had sufficient statistical evidence to state that MDLText outperformed all of the methods that we evaluated.

In future research, we aim to study the impact of feature selection in the performance of the MDLText in both online and

offline learning contexts. Furthermore, we aim to adapt the proposed algorithm to deal with other real-world classification problems. Thus, we plan to extend the models to handle numerical and categorical features.

Acknowledgment

The authors are grateful for financial support from the Brazilian agencies FAPESP, Capes, and CNPq (grant 141089/2013-0).

References

[1] I. Ahmed, R. Ali, D. Guan, Y.-K. Lee, S. Lee, T. Chung, Semi-supervised learning using frequent itemset and ensemble learning for SMS classification, Expert Syst. Appl. 42 (3) (Feb. 2015) 1065–1073.
[2] T.C. Alberto, J.V. Lochter, T.A. Almeida, Tubespam: Comment Spam Filtering on Youtube, in: Proceedings of the 14th International Conference on Machine Learning and Applications (ICMLA'15), IEEE, Miami, FL, USA, Dec. 2015, pp. 138–143.
[3] T.A. Almeida, T.P. Silva, I. Santos, J.M.G. Hidalgo, Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering, Knowl. Based Syst. 108 (May 2016) 25–32.
[4] T.A. Almeida, A. Yamakami, J. Almeida, Spam filtering: how the dimensionality reduction affects the accuracy of naive bayes classifiers, J. Internet Serv. Appl. 1 (3) (Feb. 2011) 183–200.
[5] M. Alsaleh, A. Alarifi, F. Al-Quayed, A. Al-Salman, Combating Comment Spam with Machine Learning Approaches, in: Proceedings of the 14th International Conference on Machine Learning and Applications (ICMLA'15), IEEE, Miami, FL, USA, Dec. 2015, pp. 295–300.
[6] F. Assis, W. Yerazunis, C. Siefkes, S. Chhabra, Exponential Differential Document Count – a Feature Selection Factor for Improving Bayesian Filters Accuracy, in: Proceedings of the 2006 MIT Spam Conference (SP'06), 2006, pp. 1–6. Cambridge, MA, USA.
[7] A. Barron, J. Rissanen, B. Yu, The minimum description length principle in coding and modeling, IEEE Trans. Inf. Theory 44 (6) (Oct. 1998) 2743–2760.
[8] N. Begum, B. Hu, T. Rakthanmanon, E. Keogh, Towards a Minimum Description Length Based Stopping Criterion for Semi-supervised Time Series Classification, in: Proceedings of the 14th IEEE International Conference on Information Reuse and Integration (IRI'13), IEEE, San Francisco, CA, USA, Aug. 2013, pp. 333–340.
[9] B.E. Boser, I.M. Guyon, V.N. Vapnik, A Training Algorithm for Optimal Margin Classifiers, in: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT'92), ACM, Pittsburgh, PA, USA, Jul. 1992, pp. 144–152.
[10] A. Bosin, N. Dessì, B. Pes, High-dimensional Micro-array Data Classification Using Minimum Description Length and Domain Expert Knowledge, in: Advances in Applied Artificial Intelligence, Vol. 4031 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 790–799.
[11] L. Bottou, Large-scale Machine Learning with Stochastic Gradient Descent, in: Y. Lechevallier, G. Saporta (Eds.), Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010), Springer, Paris, France, Aug. 2010, pp. 177–187.
[12] L. Breiman, Random forests, Mach. Learn. 45 (1) (Oct. 2001) 5–32.
[13] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC Press, 1984.
[14] C. Cortes, V.N. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (Sep. 1995) 273–297.
[15] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (Jan. 1967) 21–27.
[16] K. Crammer, M. Dredze, F. Pereira, Confidence-weighted linear classification for text categorization, J. Mach. Learn. Res. 13 (1) (Jun. 2012) 1891–1926.
[17] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (Dec. 2006) 1–30.
[18] P. Domingos, The role of occam's razor in knowledge discovery, Data Min. Knowl. Discov. 3 (1999) 409–425.
[19] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd edition, Wiley-Interscience, 2000.
[20] H.J. Escalante, M.A. García-Limón, A. Morales-Reyes, M. Graff, M.M. y Gómez, E.F. Morales, J. Martínez-Carranza, Term-weighting learning via genetic programming for text classification, Knowl. Based Syst. 83 (C) (Jul. 2015) 176–189.
[21] Y. Freund, R.E. Schapire, Large margin classification using the perceptron algorithm, Mach. Learn. 37 (3) (Dec. 1999) 277–296.
[22] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (2–3) (Nov. 1997) 131–163.
[23] P.D. Grünwald, A Tutorial Introduction to the Minimum Description Length Principle, Advances in Minimum Description Length: Theory and Applications, MIT Press, 2005.
[24] P.D. Grünwald, I.J. Myung, M.A. Pitt, Advances in Minimum Description Length: Theory and Applications, The MIT Press, 2005.
[25] M. Gutlein, E. Frank, M. Hall, A. Karwath, Large-scale Attribute Selection Using Wrappers, in: Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining (CIDM'09), IEEE, Nashville, TN, USA, Mar. 2009, pp. 332–339.
[26] E.-H.S. Han, G. Karypis, V. Kumar, Text Categorization Using Weight Adjusted K-nearest Neighbor Classification, in: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'01), Springer Berlin Heidelberg, Hong Kong, China, Apr. 2001, pp. 53–65.
[27] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S.S. Keerthi, S. Sundararajan, A Dual Coordinate Descent Method for Large-scale Linear SVM, in: Proceedings of the 25th International Conference on Machine Learning (ICML'08), ACM, Helsinki, Finland, Jun. 2008, pp. 408–415.
[28] C. Hsu, C. Chang, C. Lin, A practical guide to support vector classification, Tech. rep., National Taiwan University, 2003.
[29] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. 13 (2) (Mar. 2002) 415–425.
[30] X. Hu, X. Zhang, C. Lu, E.K. Park, X. Zhou, Exploiting Wikipedia as External Knowledge for Document Clustering, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), ACM, Paris, France, Jun. 2009, pp. 389–396.
[31] L. Jiang, C. Li, S. Wang, L. Zhang, Deep feature weighting for naive bayes and its application to text classification, Eng. Appl. Artif. Intell. 52 (2016) 26–39.
[32] L. Jiang, D. Wang, Z. Cai, Discriminatively weighted naive bayes and its application in text classification, Int. J. Artif. Intell. Tools 21 (01) (2012) 1250007.
[33] L. Jiang, S. Wang, C. Li, L. Zhang, Structure extended multinomial naive bayes, Inf. Sci. 329 (C) (2016) 346–356.
[34] S. Jiang, Efficient Classification Method for Large Dataset, in: Proceedings of the 2006 International Conference on Machine Learning and Cybernetics (ICMLC'06), IEEE, Aug. 2006, pp. 1190–1194.
[35] S. Jiang, G. Pang, M. Wu, L. Kuang, An improved k-nearest-neighbor algorithm for text categorization, Expert Syst. Appl. 39 (1) (2012) 1503–1509.
[36] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, Jul. 2016. ArXiv e-prints abs/1607.01759. http://arxiv.org/abs/1607.01759.
[37] S. Kim, H. Rim, H. Lim, A New Method of Parameter Estimation for Multinomial Naive Bayes Text Classifiers, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'02), ACM, Tampere, Finland, Aug. 2002, pp. 391–392.
[38] Y. Ko, A new term-weighting scheme for text classification using the odds of positive and negative class probabilities, J. Assoc. Inf. Sci. Technol. 66 (12) (Jan. 2015) 2553–2565.
[39] I. Kohonenko, The Minimum Description Length Based Decision Tree Pruning, in: PRICAI'98: Topics in Artificial Intelligence, Vol. 1531 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 228–237.
[40] A.N. Kolmogorov, Three approaches to the quantitative definition of information, Probl. Inf. Transmis. 1 (1965) 1–7.
[41] W. Lam, F. Bacchus, Learning bayesian belief networks: an approach based on the MDL principle, Comput. Intell. 10 (3) (Aug. 1994) 269–293.
[42] D.D. Lewis, M. Ringuette, A Comparison of Two Learning Algorithms for Text Categorization, in: Proceedings of the 3rd annual symposium on document analysis and information retrieval (SDAIR'94), volume 33, Apr. 1994, pp. 81–93. Las Vegas, NV.
[43] D.J.C. MacKay, Information Theory, Inference & Learning Algorithms, Cambridge University Press, 2005. New York, NY, USA.
[44] A. McCallum, K. Nigam, A Comparison of Event Models for Naive Bayes Text Classification, in: Proceedings of the 15th AAAI Workshop on Learning for Text Categorization (AAAI'98), Jul. 1998, pp. 41–48. Madison, Wisconsin.
[45] M. Mehta, R. Agrawal, J. Rissanen, SLIQ: A Fast Scalable Classifier for Data Mining, in: Advances in Database Technology – EDBT'96, Vol. 1057 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1996, pp. 18–32.
[46] T.M. Mitchell, Machine Learning, 1st edition, McGraw-Hill, New York, NY, USA, 1997.
[47] A. Muhammad, N. Wiratunga, R. Lothian, Contextual sentiment analysis for social media genres, Knowl. Based Syst. (2016) 1–10.
[48] X.-H. Phan, C.-T. Nguyen, D.-T. Le, L.-M. Nguyen, S. Horiguchi, Q.-T. Ha, A hidden topic-based framework toward building applications with short web documents, IEEE Trans. Knowl. Data Eng. 23 (7) (Jul. 2011) 961–976.
[49] A.S. Potapov, Principle of representational minimum description length in image analysis and pattern recognition, Pattern Recognit. Image Anal. 22 (1) (Mar. 2012) 82–91.
[50] J.R. Quinlan, C4.5: Programs for Machine Learning, 1st edition, Morgan Kaufmann, San Mateo, CA, USA, 1993.
[51] J.R. Quinlan, R.L. Rivest, Inferring decision trees using the minimum description length principle, Inf. Comput. 80 (3) (Mar. 1989) 227–248.
[52] J.D. Rennie, L. Shih, J. Teevan, D.R. Karger, Tackling the Poor Assumptions of Naive Bayes Text Classifiers, in: Proceedings of the 20th International Conference on Machine Learning (ICML'03), volume 3, 616–623, Washington, DC, USA, Aug. 2003.
[53] J. Rissanen, Modeling by shortest data description, Automatica 14 (5) (Sep. 1978) 465–471.
[54] J. Rissanen, A universal prior for integers and estimation by minimum description length, Ann. Stat. 11 (2) (Jun. 1983) 416–431.
[55] J. Rissanen, Fisher information and stochastic complexity, IEEE Trans. Inf. Theory 42 (1) (Jan. 1996) 40–47.
[56] J.J. Rocchio, Relevance Feedback in Information Retrieval, in: G. Salton (Ed.), The Smart retrieval system - experiments in automatic document processing, Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 313–323.
[57] R.G. Rossi, A. de Andrade Lopes, S.O. Rezende, Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts, Inf. Process. Manage. 52 (2) (Mar. 2016) 217–257.

[58] R.G. Rossi, R.M. Marcacini, S.O. Rezende, Benchmarking text collections for classification and clustering tasks, Tech. rep. 395, Institute of Mathematics and Computer Sciences, University of Sao Paulo, 2013.
[59] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd edition, Prentice Hall, 2010.
[60] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manage. 24 (5) (Aug. 1988) 513–523. http://dx.doi.org/10.1016/0306-4573(88)90021-0.
[61] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, Mag. Commun. ACM 18 (11) (Nov. 1975) 613–620.
[62] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (1) (Mar. 2002) 1–47.
[63] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Inf. Process. Manage. 45 (4) (Jul. 2009) 427–437.
[64] P. Soucy, G.W. Mineau, Beyond Tfidf Weighting for Text Categorization in the Vector Space Model, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05), Jul. 2005, pp. 1130–1135. Morgan Kaufmann, Edinburgh, Scotland.
[65] A.K. Uysal, S. Gunal, A novel probabilistic feature selection method for text classification, Knowl. Based Syst. 36 (Dec. 2012) 226–235.
[66] S. Wang, L. Jiang, C. Li, Adapting naive bayes tree for text classification, Knowl. Inf. Syst. 44 (1) (Jul. 2015) 77–89.
[67] W.J. Wilbur, W. Kim, The ineffectiveness of within-document term frequency in text classification, Inf. Retr. Boston 12 (5) (Oct. 2009) 509–525.
[68] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd edition, Morgan Kaufmann, 2011. San Francisco, CA, USA.
[69] Q. Wu, Y. Ye, H. Zhang, M.K. Ng, S. Ho, Forestexter: an efficient random forest algorithm for imbalanced text categorization, Knowl. Based Syst. 67 (Sep. 2014) 105–116.
[70] H. Zhang, G. Zhong, Improving short text classification by learning vector representations of both words and hidden topics, Knowl. Based Syst. 102 (Jun. 2016) 76–86.
[71] L. Zhang, L. Jiang, C. Li, A new feature selection approach to naive bayes text classifiers, Int. J. Pattern Recognit Artif Intell. 30 (02) (2016) 1650003.
[72] L. Zhang, L. Jiang, C. Li, G. Kong, Two feature weighting approaches for naive bayes text classifiers, Knowl. Based Syst. 100 (2016) 137–144.
[73] T. Zhang, Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms, in: Proceedings of the 21th International Conference on Machine Learning (ICML'04), ACM, Banff, Alberta, Canada, Jul. 2004, pp. 116–123.