An HMM-based Text Classifier Less Sensitive to Document Management Problems

A. Seara Vieira, E. L. Iglesias, and L. Borrajo

Computer Science Dept., Univ. of Vigo, Escola Superior de Enxeñería Informática,
Ourense, Spain
{adrseara,eva,lborrajo}@uvigo.es

Abstract. The performance of the text classification techniques is commonly affected by the characteristics and representation of the document corpora itself. Of all the problems arising from the corpus, there are three major difficulties which the classifiers must deal with: the feature selection issues, the class imbalance problem and the size of the training set. The objective of this paper is to present a based-content text classifier called T-LHMM that is less sensitive to the text representation and the size of the corpus, and more efficient in terms of running time than other classification techniques. To demonstrate it, we present a set of experiments performed on well-known biomedical text corpora. Our classifier is also compared with k-NN and SVM models.

Keywords: Based-content text classification, Class imbalance, Feature selection, Hidden Markov Model, k-Nearest Neighbours, Support Vector Machines

1 Introduction

With the rapid growth of corporate and digital databases, text classification has become one of the key techniques for handling and organizing text data. Text classification is the task of automatically assigning documents to a predefined set of classes or topics [1]. Different classification techniques have been applied to text classification problems, including Naive Bayes, k-Nearest Neighbours (kNN), and Support Vector Machines (SVMs), which have proven their effectiveness in many scenarios [2–4]. However, the performance of these techniques is commonly affected by the characteristics and representation of the text corpora itself.

Of all the problems arising from the corpus, there are three major difficulties which the classifiers must deal with: the feature selection issues, the class imbalance problem and the size of the training set.

1.1 Feature selection

The representation of a document has a strong impact on the generalization accuracy of a learning system. Documents, which are typically strings of characters, have to be transformed into a format that automatic classifiers can handle.

The most common technique in document classification tasks is the *bag-of-words* approach [5]. In this case, every document is represented by a vector where elements describe the word frequency (number of occurrences) of a certain term. The final selection of words that will be used to represent the documents is called *feature words.*

This classical text representation technique is hindered by the practical limitations of big text corpora. The complexity of the text classification in terms of time and space depends on the size of the aforementioned vectors. Generally, not all words have a significant contribution to discriminately represent a text. *Feature reduction algorithms* can be applied to the initial feature word set in order to reduce its dimensionality. Some of these techniques can remove redundant or irrelevant features from the dataset based on statistical filtering methods such as *Information Gain (InfoGain)* [6]. Another type of feature reduction methods creates new attributes as a combination of the original attributes, such as the *Principal Component Analysis* (PCA) technique [7]. In general, the application of these algorithms can improve classification accuracy and reduce the computational cost of the whole process, although it may be too costly to compute for high-dimensional datasets.

1.2 Class imbalance

The class imbalance problem corresponds to domains where one class is represented by only a few number of instances, while the others present a much larger number of examples [8, 9]. The performance of standard classifiers such as those mentioned is reduced when classifying this kind of corpus due to an assumption of a balanced distribution of classes. Learning methods tend to ignore rare classes and achieve low performance on them in favor of larger classes because of the size effect [10].

Sampling strategies such as over- and undersampling are popular in tackling the problem of class imbalance [11]. The undersampling algorithms artificially decrease the number of samples that belong to larger classes, while the oversampling algorithms can be used to increase the number of instances of rare classes by taking the larger classes into account. In general, these techniques can improve the classification performance when the corpus is not balanced. However, it is possible to lose valuable information for the classifier (undersampling case) or increase the time needed for the training phase and overfit the model (oversampling case).

1.3 Training set size and learning curves

Generally, in a supervised text classification process, the more documents the classifier can learn from in the training phase, the higher the classifier accuracy [12]. This connection between the training sample size and the model accuracy can be depicted by a *learning curve.* On a typical learning curve, the horizontal axis represents the number of instances used for training, while the vertical axis represents a measure of the accuracy that the model obtained in a test process using a different set of instances.

Learning curves typically have a fast increasing portion early in the curve, followed by a relatively slow increasing portion, and finally a plateau portion late in the curve [13]. The plateau shows that the curve has converged and occurs when the model accuracy no longer increases with more training data.

The computational complexity of a learning algorithm depends on the number of training instances, with a linear dependency being the best case. The usage of small datasets increases the efficiency of the classification process. The convergence point of the learning curve of a classifier may indicate the minimum number of instances required to perform well in terms of accuracy.

In summary, the problems related to the text corpora as described above, can strongly affect the performance of any classifier.

The objective of this paper is to present a novel classifier called T-LHMM that classifies documents based on their content. T-LHMM is an expansion of the T-HMM technique [14], where the model is adapted to use general learning algorithms that are specific to the HMM theory. The new approach helps to reduce its sensitivity to the changes of the dataset and also to reduce the parameterization cost. To show the robustness of the model, we test its variations when applying different techniques that solve the above mentioned problems. Specifically, the experiments are focused on showing the influence level that these limitations have in the T-LHMM classifier. In addition, the SVM and *k*-NN techniques are also used in the tests to provide a comparative viewpoint.

2 T-LHMM: The novel text classifier

Hidden Markov Models (HMM) have been used to describe the statistical properties of a sequential random process. They are known for their application in language problems like speech recognition and pattern matching [15]. However, their application has been extended to fields of text processing such as information extraction [16–18], information retrieval [19], text categorization [20–22], and text classification [23, 24].

One of the common tasks that can be addressed by deploying an HMM in text analysis is to infer the internal structure of the documents. This task is related to the decoding problem for HMMs. For instance, Frasconi *et al.* [20] use HMM to categorize pages in a multi-page document, exploiting the internal organization of the document to provide the learner with better information. The HMM is trained with page sequences to subsequently find the most probable structure for a new given document.

Another text processing task that is solved with the use of HMMs is to perform a word sequence classification. In this case, the HMMs are treated as word generators with an inner state transitions that defines the sequences they can produce. This way, they can be used to determine the probability that a given input sequence is the result of the HMM. This task is related to the evaluation problem for HMMs. For example, Kairong Li *et al.* [21] research the text categorization process based on Hidden Markov Models. The main idea of the article lies in setting up an HMM classifier, combining χ^2 and an improved TF-IDF (Term Frequency-Inverse Document Frequency, statistic that reflects how important a word is to a document in a collection or corpus [25]) method, and reflecting the semantic relationship in the different categories. The process shows the semantic character in different documents to make the text categorization process more stable and accurate.

Kwan Yi *et al.* [23] use the previous idea in a similar approach. They describe the text classification as the process of finding a relevant class c for a given document d. They implement a Hidden Markov Model to represent each class. Thus, given a document d, the probability that a document d belongs to the class c is computed on the specific HMM model c. In their system, a document is treated as a wordlist, and the HMM for each class is viewed as a generator of a word sequence.

In general, HMMs need to be adapted in order to be used in a text classification process. Previous HMM related techniques provide a way of applying these models to a text processing task; however, they have a query expansion approach [19] or need to use prior knowledge to achieve significant results [23].

In order to obtain a simpler and more effective classifier system than existing methods, we propose the T-LHMM model based on the Hidden Markov Model, an improved version of the T-HMM classifier [14]. This model aims to classify documents according to their content.

In our context, given a training set $T = \{(d_1, c_1), (d_2, c_2)...(d_n, c_n)\}$, which consists of a set of preclassified documents in classes, we want to build a classifier using HMM to model the implicit relationship between the characteristics of the document and its class, in order to be able to accurately classify new unknown documents.

Each document d_i has a binary class attribute c_i which can have a value of Relevant or Non-relevant. Our work is therefore focused on building a classifier based on the training set that can classify new documents as relevant or non-relevant without previously knowing their class information.

Following the idea proposed by Kwan Yi *et al.* [23], we use HMM as a document generator. In the T-LHMM model, an HMM is implemented for each class: Relevant and Non-Relevant. Each model is then trained with documents belonging to the class that the model represents. When a new document needs to be classified, the system evaluates the probability (likelihood) of this document being generated by each of the Hidden Markov models. As a result, the class with the maximum likelihood value is selected and considered as the output class for the document.

Furthermore, since T-LHMM aims to classify documents according to their content, input data needs to be expressed in a format that HMM algorithms can handle. As stated in the Introduction Section, the most common approach is to represent every document by a vector where elements describe word frequencies. Additionally, some adjustments such as TF-IDF may be made in the word frequency value in order to improve the document representation. Fig. 1(a) shows an example of a corpus processed in this way.

In our case, a document set represented with the previous approach is given as input. We format the documents so that they are represented by a wordlist ordered by their relevance. In this case, the TF-IDF value is taken as the relevance measure. Words with a higher TF-IDF value are more relevant because they are considered the best representation of the document semantics. In this way, words are placed in descending order according to their relevance in representing each document, as seen in Fig. 1(b).

When building an HMM model, it is important to consider what “hidden states” stand for. In our model, hidden states reflect the difference in relevance (ranking) between words in a document. Each state represents a relevance level for words appearing in the corpus. That is, the most probable observations for the first state are the most relevant words in the corpus. The most probable observations for the second state are the words holding the second level of relevance in the corpus, and so on. This can be seen in Fig. 1(c). It should be noted that the number of states of the HMM sets the maximum cardinality of the wordlist. Words with a relevance value equal to zero and those whose ranking exceeds the number of states are ignored. The number of states N is a modifiable parameter that depends on the training corpus and how much flexibility we want to add to the model.

The HMMs for each class must be trained with documents belonging to the class they represent. In order to adjust an HMM to a training dataset, a procedure called Baum-Welch (BW) re-estimation may be used [15]. By applying the algorithm with the dataset, the parameters of an initial required model can be made to converge on values which are locally optimal for the given documents.

Considering that each document is ultimately represented by a vector or a wordlist ranked in decreasing order, and ignoring words with zero value, an initial Hidden Markov model is proposed to represent a predefined class c as follows:

1. The union of words from the training corpus is taken as the set of observation symbols V. For each word, there is a symbol v_k. The set of possible observations is the same for every HMM, taking into account all words in the corpus, regardless of their class.
2. As mentioned above, states represent ranking positions. Therefore, states are ordered from the first rank to the last one. The state transitions are ordered sequentially in the same way, forming a linear HMM [15]. The elements of the transition probability matrix A are defined as:

$$a_{ij} = \begin{cases} \frac{1}{2} & \text{if } j = i + 1 \\ \frac{1}{2} & \text{if } j = i \\ 0 & \text{in other case} \end{cases}$$

   where a_{ij} represents the transition probability of the state i to the state j.
3. The observation output probability distribution of each state is defined according to the training corpus and the class c. A word/observation v_k has a higher output probability at a given state s_i if the word appears frequently

with the same ranking position that s_i represents. In addition, all states, regardless of the rank they represent, have a probability of emitting words appearing in documents with the class c for which the HMM was built. Given a class c and a dataset D_c of documents that belong to that class, the elements of the output probability matrix B for an HMM that represents the class c is defined as follows:

$$b_i(v_k) = \frac{\sum_{d \in D_c} R_d(v_k, i) + \sum_{d \in D_c} A_d(v_k) + 1}{\sum_{d \in D_c} E_d(i) + \sum_{j=0}^{|V|} \left(\sum_{d \in D_c} A_d(v_j) \right) + |V|}$$

(a) $b_i(v_k)$ stands for the probability of the word/observation v_k being emitted at the state s_i

(b) $R_d(v_k, i) = \begin{cases} 1 \text{ if word } v_k \text{ appears at } i^{th} \text{ rank position in document } d \\ 0 \text{ in other case} \end{cases}$

(c) $E_d(i) = \begin{cases} 1 \text{ if there is any word with } i^{th} \text{ rank position in document } d \\ 0 \text{ in other case} \end{cases}$

(d) $A_d(v_k) = \begin{cases} 1 \text{ if word } v_k \text{ appears at least one time in document } d \\ 0 \text{ in other case} \end{cases}$

(e) $|V|$ is the number of feature words.

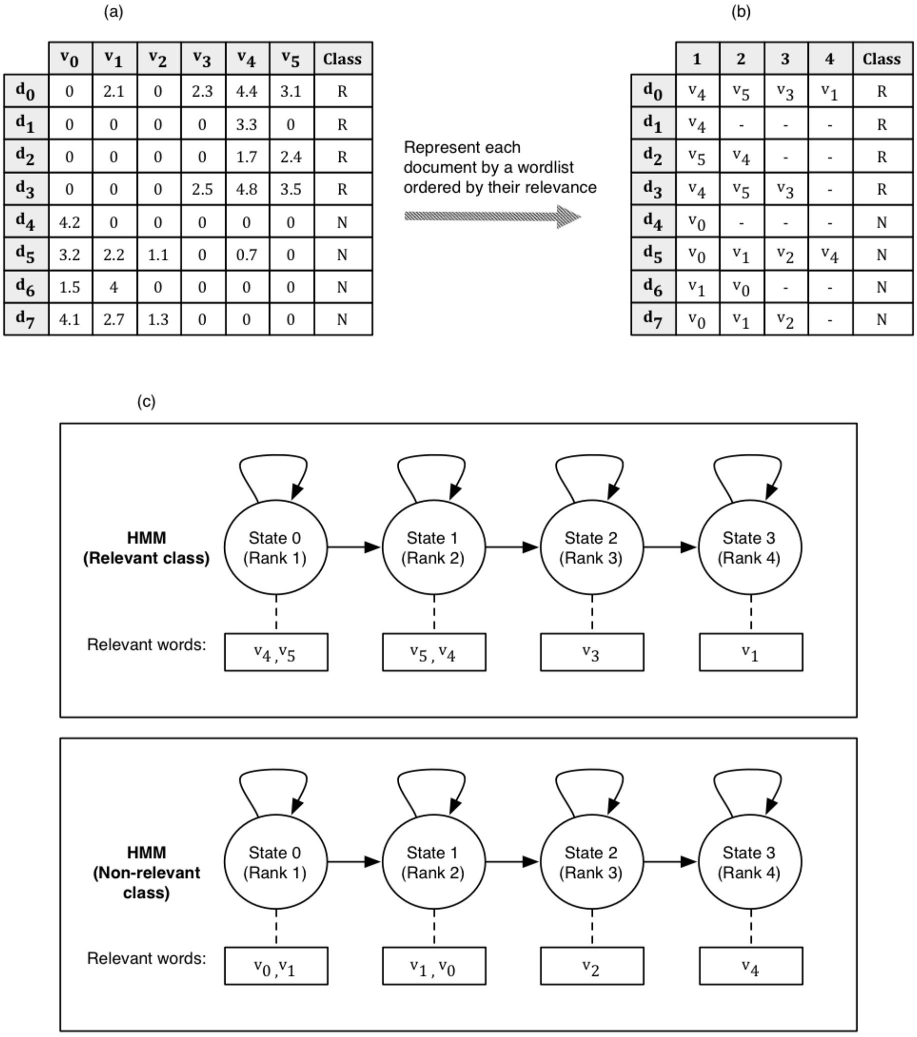
4. The initial probability distribution π is defined by giving probability 1 to the first state s_0.

As previously mentioned, this is considered the initial HMM model representing a class c for the application of the Baum-Welch algorithm. This process requires adjusting to a set of observation sequences. In this case, the documents from the training set formatted as previously explained (see Fig. 1(b)) are taken as the set of observation sequences. The resultant HMM after applying the algorithm with D_c is considered to be the trained HMM representing class c.

2.1 Classifying new documents

Once the two Hidden Markov models are created and trained (one for each class), a new document d can be classified by, first of all, formatting it into an ordered wordlist L_d in the same way as in the training process. Then, as words are considered observations in T-LHMM, we calculate the probability (likelihood) of the word sequence L_d being produced by the two HMMs. That is, $P(L_d|\lambda_R)$ and $P(L_d|\lambda_N)$ need to be computed, where λ_R is the model for relevant documents and λ_N the model for non-relevant documents. The final output class for document d is the class represented by the HMM with the highest calculated likelihood.

The calculation of the likelihood measures is made by applying the forward-backward algorithm explained in the Rabiner article [15].



(a)

	v_0	v_1	v_2	v_3	v_4	v_5	Class
d_0	0	2.1	0	2.3	4.4	3.1	R
d_1	0	0	0	0	3.3	0	R
d_2	0	0	0	0	1.7	2.4	R
d_3	0	0	0	2.5	4.8	3.5	R
d_4	4.2	0	0	0	0	0	N
d_5	3.2	2.2	1.1	0	0.7	0	N
d_6	1.5	4	0	0	0	0	N
d_7	4.1	2.7	1.3	0	0	0	N

(b)

	1	2	3	4	Class
d_0	v_4	v_5	v_3	v_1	R
d_1	v_4	-	-	-	R
d_2	v_5	v_4	-	-	R
d_3	v_4	v_5	v_3	-	R
d_4	v_0	-	-	-	N
d_5	v_0	v_1	v_2	v_4	N
d_6	v_1	v_0	-	-	N
d_7	v_0	v_1	v_2	-	N

Fig. 1. Example of the HMM structure and input data. (a) Input document vector matrix where documents are represented by word frequencies adjusted with TF-IDF. (b) Adaptation of the input data to the T-LHMM classifier. Each document is represented by a wordlist ordered by their relevance. (c) An HMM is built for each class (Relevant and Non-relevant). Since states represent rankings of words, the most relevant words for a rank have a higher output probability in their state.

3 Corpus description

3.1 TREC Genomics Corpus

One of the tasks in TREC Genomics 2005 Track [26] was to automatically classify a full-text document collection with the train and test sets, each consisting of about 6,000 biomedical journal articles.

Systems were required to classify full-text documents from a two-year span (2002-2003) of three journals, with the documents from 2002 comprising the train data, and the documents from 2003 making up the test data.

The categorization task assessed how well systems can categorize documents in four separate categories: A (Alelle), E (Expression), G (GO annotation), and T (Tumor). In this paper, Allele and GO annotation categories are used to test the performance of the classifiers in the different scenarios. A different corpus is created for each category, where documents can be classified as relevant or non-relevant.

Allele and GO were selected because they result in the best and the worst efficiency, respectively, for the classifiers tested.

3.2 OHSUMED Corpus

The OHSUMED test collection, initially compiled by Hersh *et al.* [27], is a subset of the MEDLINE database, a bibliographic database of important medical literature maintained by the National Library of Medicine. OSHUMED contains 348,566 references consisting of fields such as titles, abstracts, and MeSH descriptors from 279 medical journals published between 1987 and 1991.

The collection includes 50,216 medical abstracts from the year 1991, which were selected as the initial document set. Each document in the set has one or more associated categories (from the 23 disease categories). In order to adapt them to a scheme similar to the TREC corpus, which consists of distinguishing relevant documents from non-relevant ones, we select one of these categories as relevant and consider the others as non-relevant. If a document has been labeled with two or more categories and one of them is considered relevant, then the document itself is relevant and is excluded from the set of non-relevant documents.

Five categories are chosen as relevant: Neoplasms(C04), Digestive (C06), Cardio (C14), Immunology (C20) and Pathology (C23), since they are by far the most frequent categories of the OHSUMED corpus. The other 18 categories are considered as the common bag of non-relevant documents. For each one of the five relevant categories, a different corpus is created in the way mentioned above, resulting in five distinct matrices.

4 Experiments

The goal of the experiments is to test the performance of text classifiers under different conditions that affect the training dataset. In addition to the accuracy of the classifier, the time efficiency is also measured and compared in the same scenarios. This can show how much the classifiers are dependent on the characteristics of the corpus.

Fig. 2 shows the tests performed for a specific Corpus. These experiments are divided into four different parts: Corpus preprocessing and splitting, Feature selection, Document Sampling and Corpus Size Reduction. The last three correspond to the aforementioned classification problems. These parts are explained below along with the selection of techniques to solve each problem and the classifiers selected for comparison.

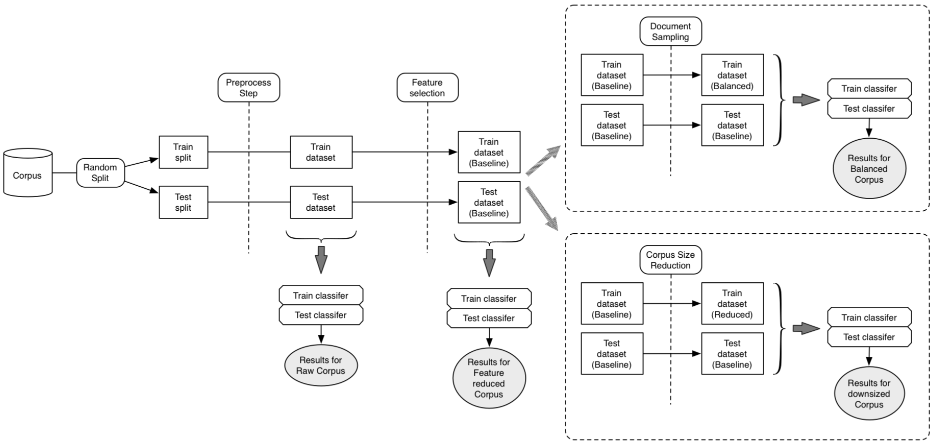


Fig. 2. Experiments performed for a specific corpus.

4.1 Corpus preprocessing and splitting

Initially, the document corpora need to be preprocessed. In an initial step, called Random Split in Fig. 2, the initial corpus is divided into two different splits: the train split and the test split. This division is performed randomly while maintaining the class proportion in the original corpus. These splits are used in the training and testing phases of each evaluation step for each classifier.

In a subsequent step and following the bag-of-words approach, we format every document into a vector of features in which elements describe the word occurrence frequencies adjusted using the TF-IDF statistic. All the different words that appear in the training corpus are candidates for feature words. In order to reduce the initial feature size, standard text pre-processing techniques are used. A predefined list of stopwords (common English words) is removed from the text, and a stemmer based on the Lovins stemmer [28] is applied. Finally, words occurring in fewer than five documents of the entire training corpus are also removed.

Table 1 shows the characteristics of the corpora after applying the initial preprocess step.

	Number of documents		Number of features
Corpus	Relevant	Non-Relevant	
TREC Allele	591	10204	13578
TREC GO	862	9933	13493
Ohsumed C04	2630	7755	10671
Ohsumed C06	1220	8430	10413
Ohsumed C14	2550	8030	10661
Ohsumed C20	1220	8239	10348
Ohsumed C23	3952	6778	10328

Table 1. Description of the datasets

At this point, an evaluation of the classifiers is performed to retrieve the results achieved with the raw corpus (see Fig. 2).

4.2 Feature selection

As stated previously, feature reduction algorithms can be applied to the initial feature set in order to further reduce its dimensionality. In this part of the experiments, the goal is to test the degree of dependence of the classifiers on the number of features.

The feature selection method based on the InfoGain that is implemented in WEKA [29] is used as the feature reduction algorithm for this study, since it was previously employed and its effectiveness proved in similar text classification tasks [6,30]. This algorithm uses a threshold to determine the size of the reduced feature set. In this case, the threshold is set to the minimum value, so that every feature word with a non-zero value of the InfoGain is included in the resultant feature set.

After this feature selection phase, the output train and test datasets are created in which the number of feature words has been reduced. We call these new datasets the “Baseline” (see Fig. 2), since they are used in the subsequent parts of the experiments. This is done for reasons of time complexity, as the original size of the feature set is very large for classification with classifiers like SVM.

Finally, an evaluation of the classifiers is performed to retrieve the results achieved with the Baseline corpus (see Fig. 2).

4.3 Document Sampling

All the selected corpora suffer from some degree of imbalance in the class distribution, as it can be seen in Table 1. The goal of this part of the experiments is to test how the class imbalance problem affects the classifiers. In order to achieve this, document sampling techniques are applied to the Baseline train dataset to evaluate the changes in performance between the imbalanced corpus (Baseline) and the output balanced corpus.

In this case, the document sampling methods selected to perform the evaluations are ROS (Random Over-Sampling) [8], RUS (Random Under-Sampling) [8] and SMOTE (Synthetic Minority Over-sampling Technique) [31]. It is important to note that the sampling techniques only modify the train dataset used in the training phase of the classifiers. In addition, the target class balance is set to 50% for all the methods and corpus.

Finally, an evaluation of the classifiers is performed to retrieve the results achieved with the "Balanced" corpus (see Fig. 2).

4.4 Corpus Size Reduction

The goal of this part is to test how the number of training documents affects the performance of the classifiers. These experiments aim to output a learning curve for each classifier where the minimum number of documents needed to converge can be approximated.

In order to achieve this, the Baseline train dataset is reduced to the following percentage of documents: [2%,4%,6%,8%,10%,15%,20%,25%,30%,40%,50%,75%]. It is important to note that reductions are performed randomly and conserve the class balance present in the Baseline corpus.

Finally, an evaluation of the classifiers is performed to retrieve the results achieved with the "Downsized" corpus (see Fig. 2).

4.5 Classifiers

In this study, the following classification algorithms are trained and tested to perform the document classification tasks in the evaluation processes:

- T-LHMM: The text classifier developed by the authors is the baseline classifier to test and compare. In this case, the number of states N is set to 60 for all the corpus. This value leads to the best average results in the classification of the raw corpora, as shown in Fig. 3. This indicates that not all the words help to improve the classification results, especially those with less relevance value in each document. In addition, since the training document set is large, a single iteration of the Baum-Welch algorithm is enough to achieve its best performance. Further iterations overfit the model and lead to worse results. In order to carry out the tests, an implementation of HMM in Java (JAHMM) is used.
- Support Vector Machine (SVM): The implementation of the SVM used in this case is LIBSVM [32] and the parameters are those utilized by default in the WEKA environment [33], applying both a RBF and a Sigmoidal kernel.
- k-NN: The number of neighbours is set to $k = 3$ since it leads to the best performance of the algorithm in the tested corpus. In addition, the Euclidean distance is used to measure the distances between documents in the classifier.

The complete collection of evaluation tests sum up 476:

- 28 tests for Raw corpus: 7 Corpus x 4 Classifiers (T-LHMM, SVM (RBF), SVM (Sig.), *k*-NN).
- 28 tests for Feature reduced corpus (Baseline): 7 Corpus x 1 Technique (InfoGain) x 4 Classifiers.
- 84 tests for Balanced corpus: 7 Corpus x 3 Techniques (ROS, SMOTE, RUS) x 4 Classifiers.
- 336 tests for Downsized corpus: 7 Corpus x 12 Size reduction percentages x 4 Classifiers.

In addition, for statistical comparison purposes, and since the corpus is randomly split in the first phase, the complete experiment scheme is executed ten times starting at the Random Split point (see Fig. 2). This makes the entire set of tests rise to 4760.

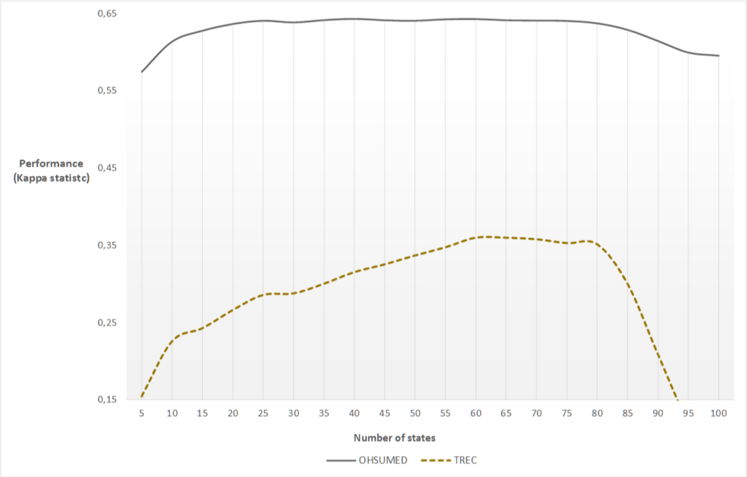


Fig. 3. Influence of the number of states in the T-LHMM. Five classification tests are executed by corpus and N-value. Results are averaged for OHSUMED and TREC corpora. The evaluation measure shown (y-axis) is the Kappa statistic.

5 Results

To evaluate the effectiveness of the models, the Kappa statistic, an evaluation measure commonly utilized in text classification and information retrieval, is used. Kappa is a single metric that takes the output confusion matrix of an evaluation and reduces it to one value [34]. Kappa statistic is interesting because it compares the accuracy of the system to the accuracy of a random system.

Possible values of Kappa range between -1 and 1, where 1 is perfect agreement between classifier and gold-standard, and negative values indicate agreement less than would be expected by chance.

Furthermore, in order to demonstrate that the observed results are not just a chance effect in the estimation process, we use a statistical test that gives confidence bounds to predict the true performance from a given test set. A Student's t-test is performed on Kappa measures achieved in each evaluation test.

The results for the feature selection and document sampling evaluations are shown in Tables 2 and 3, while the results for the corpus size reduction evaluation are shown in Fig. 5 with learning curves.

5.1 Feature selection

Table 2 shows the results achieved for each classifier with all the Raw and feature reduced corpus by the InfoGain algorithm. The main values correspond to the average Kappa value achieved for the total of 10 executions with each corpus/technique/classifier combination. The values in brackets and their symbols ($\bullet$, $\circ$) correspond to the statistical test part.

One test is performed for each pair of collection results achieved in each technique. In this case, the baseline scenario is the use of the Raw Corpus, which is compared in each classifier with the results obtained when applying the InfoGain technique. The difference in a given confidence level is checked to determine if it exceeds the confidence limit stated by a Student's distribution. If so, the *null-hypothesis* (the difference is due to chance) is rejected, proving that the application of the InfoGain technique in the corpus makes a real difference in the output of the classifier. The values in brackets correspond to the t-value calculated in the comparison of the value collections. t-value gives a measure of how large the difference is between applying or not the InfoGain technique: higher absolute values indicate more difference.

According to the results, the T-LHMM classifier is the least affected by the reduction in the number of features, since it shows a statistical difference in 4 of the 7 corpus.

In addition, the t-values show minor differences when there is a statistical difference, in the case of T-LHMM. The reverse occurs with the other classifiers, especially in the SVM (Sigmoidal kernel), where the t-values show a big difference when the feature selection technique is or is not applied, achieving much lower classification results when not applied.

In order to demonstrate the independence of the number of features for the T-LHMM, additional tests have been performed. In this case, the classification process has been executed removing an increasing number of features using the InfoGain algorithm. The results are shown in Fig. 4. For all the tested corpus, the T-LHMM algorithm records an approximately constant Kappa statistic that only decreases when the number of features reaches a certain point. In all the cases, the performance begins to hinder when the number of attributes are below the 5% of the total.

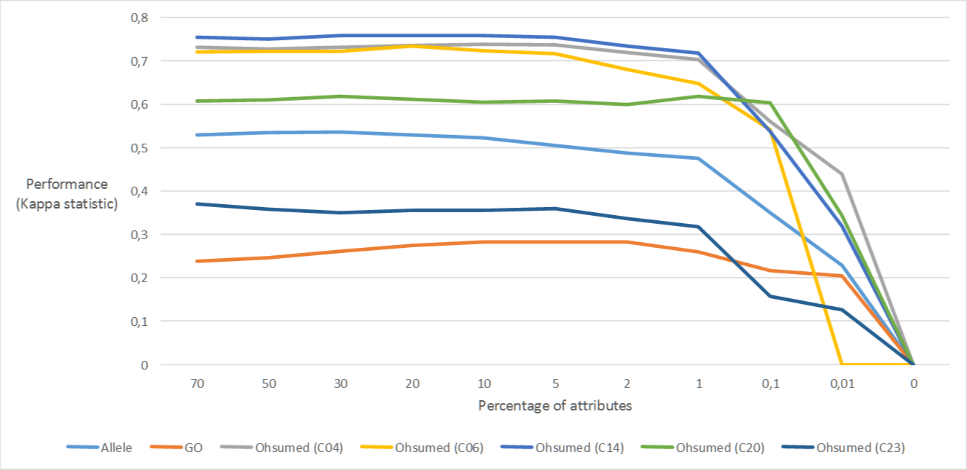


Fig. 4. Results for T-LHMM removing an increasing number of features. The horizontal axis indicates the percentage of attributes used to execute the classification process.

Corpus/Technique	HMM	SVM (*RBF*)	SVM (*Sig.*)	*k*-NN, k=3
Allele				
Raw	0,486	0,085	0,185	0,001
InfoGain	0,512 ●(-2,88)	0,072 ○(1,14)	0,503 ●(-25,01)	0,063 ●(-5,75)
GO				
Raw	0,200	0,000	0,001	0,000
InfoGain	0,266 ●(-9,86)	0,057 ●(-8,50)	0,116 ●(-19,86)	0,039 ●(-6,39)
Ohsumed C04				
Raw	0,738	0,673	0,666	0,044
InfoGain	0,740 ○(-0,29)	0,743 ●(-13,69)	0,752 ●(-22,88)	0,339 ●(-8,22)
Ohsumed C06				
Raw	0,711	0,587	0,578	0,066
InfoGain	0,702 ○(1,32)	0,705 ●(-15,19)	0,730 ●(-21,32)	0,377 ●(-18,40)
Ohsumed C14				
Raw	0,754	0,726	0,717	0,167
InfoGain	0,762 ●(-2,35)	0,768 ●(-12,03)	0,776 ●(-15,40)	0,317 ●(-4,03)
Ohsumed C20				
Raw	0,606	0,501	0,515	0,106
InfoGain	0,607 ○(-0,34)	0,673 ●(-19,14)	0,700 ●(-27,62)	0,340 ●(-6,01)
Ohsumed C23				
Raw	0,376	0,310	0,300	0,062
InfoGain	0,351 ●(5,65)	0,377 ●(-15,42)	0,356 ●(-14,01)	0,189 ●(-5,92)

●: Measured difference with Raw is statistically significant (p-value=0.05)
○: Measured difference with Raw is statistically significant (p-value=0.05)

Table 2. Results achieved for each classifier with all the Raw and feature-reduced corpora by using the InfoGain algorithm. The main values correspond to the average Kappa value achieved for the total of 10 executions with that corpus/technique/classifier. The values in brackets correspond to the t-value calculated in the comparison.

5.2 Document sampling

Table 3 shows the results achieved for each classifier with the entire over- and undersampled corpus by the selected techniques. The values are organized in the same way as the previous Table for feature selection results. Since the document sampling techniques are applied to the output dataset from the InfoGain algorithm, this is considered the baseline technique in all the statistical tests performed. That is, the application of the sampling techniques (ROS, RUS and SMOTE) is compared with the previous results for the imbalanced corpus (Baseline/InfoGain).

According to the results, the T-LHMM classifier is again the least affected by the class balance in the training corpus, since it only shows a statistical difference in 6 of the 21 cases, and achieves t-values very close to the confidence limits. The other classifiers show a higher degree of dependence on the class imbalance problem (especially with both Allele and GO corpus) reaching the point where without sampling techniques the classification performance is near to zero (in the case of SVM with RBF kernel).

Corpus/Technique	HMM	SVM (*RBF*)	SVM (*Sig.*)	*k*-NN, k=3
Allele				
Baseline	0,512	0,072	0,503	0,063
ROS	0,514 ○(-0,29)	0,364 ●(-23,54)	0,404 ●(10,60)	0,248 ●(-9,17)
SMOTE	0,548 ●(-5,06)	0,520 ●(-40,89)	0,435 ●(6,22)	0,345 ●(-19,14)
RUS	0,479 ●(4,75)	0,349 ●(-25,25)	0,543 ●(-3,45)	0,294 ●(-8,61)
GO				
Baseline	0,266	0,057	0,116	0,039
ROS	0,263 ○(0,45)	0,223 ●(-13,98)	0,176 ●(-8,68)	0,118 ●(-7,78)
SMOTE	0,261 ○(0,59)	0,230 ●(-15,18)	0,181 ●(-9,12)	0,139 ●(-11,12)
RUS	0,250 ○(2,06)	0,206 ●(-15,25)	0,254 ●(-16,20)	0,133 ●(-9,40)
Ohsumed C04				
Baseline	0,740	0,743	0,752	0,339
ROS	0,737 ○(0,63)	0,767 ●(-4,27)	0,755 ○(-0,80)	0,365 ○(-1,47)
SMOTE	0,732 ○(1,82)	0,753 ○(-1,98)	0,770 ●(-5,14)	0,376 ○(-1,96)
RUS	0,732 ○(1,63)	0,718 ●(4,09)	0,770 ●(-4,93)	0,359 ○(-1,05)
Ohsumed C06				
Baseline	0,702	0,705	0,730	0,377
ROS	0,700 ○(0,24)	0,738 ●(-4,22)	0,584 ●(15,25)	0,529 ●(-6,99)
SMOTE	0,695 ○(0,92)	0,711 ○(-0,75)	0,640 ●(10,58)	0,570 ●(-10,64)
RUS	0,674 ●(4,14)	0,555 ●(21,00)	0,742 ○(-1,91)	0,473 ●(-3,95)
Ohsumed C14				
Baseline	0,762	0,768	0,776	0,317
ROS	0,760 ○(0,50)	0,788 ●(-5,91)	0,730 ●(9,36)	0,379 ●(-4,58)
SMOTE	0,760 ○(0,51)	0,780 ●(-3,42)	0,765 ●(2,48)	0,365 ●(-3,91)
RUS	0,743 ●(5,07)	0,574 ●(39,62)	0,734 ●(8,74)	0,284 ○(1,73)
Ohsumed C20				
Baseline	0,607	0,673	0,700	0,340
ROS	0,608 ○(-0,19)	0,711 ●(-5,55)	0,616 ●(11,99)	0,459 ●(-11,36)
SMOTE	0,613 ○(-1,11)	0,714 ●(-6,00)	0,686 ○(1,96)	0,500 ●(-12,99)
RUS	0,601 ○(1,35)	0,617 ●(7,99)	0,722 ●(-3,75)	0,409 ●(-6,05)
Ohsumed C23				
Baseline	0,351	0,377	0,356	0,189
ROS	0,349 ○(0,33)	0,367 ○(1,94)	0,333 ●(7,86)	0,165 ●(3,27)
SMOTE	0,345 ○(1,17)	0,269 ●(20,22)	0,269 ●(16,94)	0,178 ○(1,57)
RUS	0,341 ○(1,81)	0,081 ●(54,42)	0,121 ●(48,60)	0,121 ●(8,15)

●: Measured difference with Baseline is statistically significant (p-value=0.05)

○: Measured difference with Baseline is not statistically significant (p-value=0.05)

Table 3. Results achieved for each classifier with all Baseline and balanced corpus by the sampling techniques. The main values correspond to the average Kappa value achieved for the total of 10 executions with that corpus/technique/classifier. The values in brackets correspond to the t-value calculated in the comparison.

5.3 Document size reduction

Fig. 5 shows the learning curves created based on the results obtained for the size reduction experiments. The plotted performance values (y-axis) correspond to the average Kappa value achieved for each of the 10 executions with each size percentage.

According to the results, the T-LHMM classifier reaches its convergence plateau before any other classifier for all the corpora. The higher slopes in its curves in the initial size increments indicate that the minimum number of training documents required to perform at its "best" is lower than that required in the other classifiers.

5.4 Running time efficiency

In order to prove whether the T-LHMM classifier produces better results in terms of running efficiency, the execution times for the experiments were also saved. This includes each train/test process made for each evaluation. Table 4 shows the average CPU time (in miliseconds) achieved by the different models.

Technique	T-LHMM	SVM (RBF)	SVM (*Sig.*)	k-NN
Raw	28.067,05	43.910,30	25.053,62	62.612,65
InfoGain	3.936,56	8.016,79	4.352,67	12.551,10
ROS	6.383,95	23.639,70	11.213,25	19.735,54
SMOTE	7.874,73	33.530,65	13.483,56	78.147,79
RUS	2.085,68	3.291,83	2.069,11	4.763,24

Table 4. Results for user CPU time in miliseconds, representing the average execution time for training and testing.

According to the results, T-LHMM is more efficient in terms of time than the other approaches. This is not a surprising conclusion, since the model proposed is focused on working with documents, while the other models have general purposes. From the entire set of feature words, the T-LHMM system ignores, for each document, those words having a frequency value equal to zero. This means that for a given document, a total of only $50 - 150$ features are actually considered in the T-LHMM calculations.

6 Conclusions

In this paper, an improved version of the T-HMM text classification model [14] is presented. The main difference between T-LHMM and the previous T-HMM lies

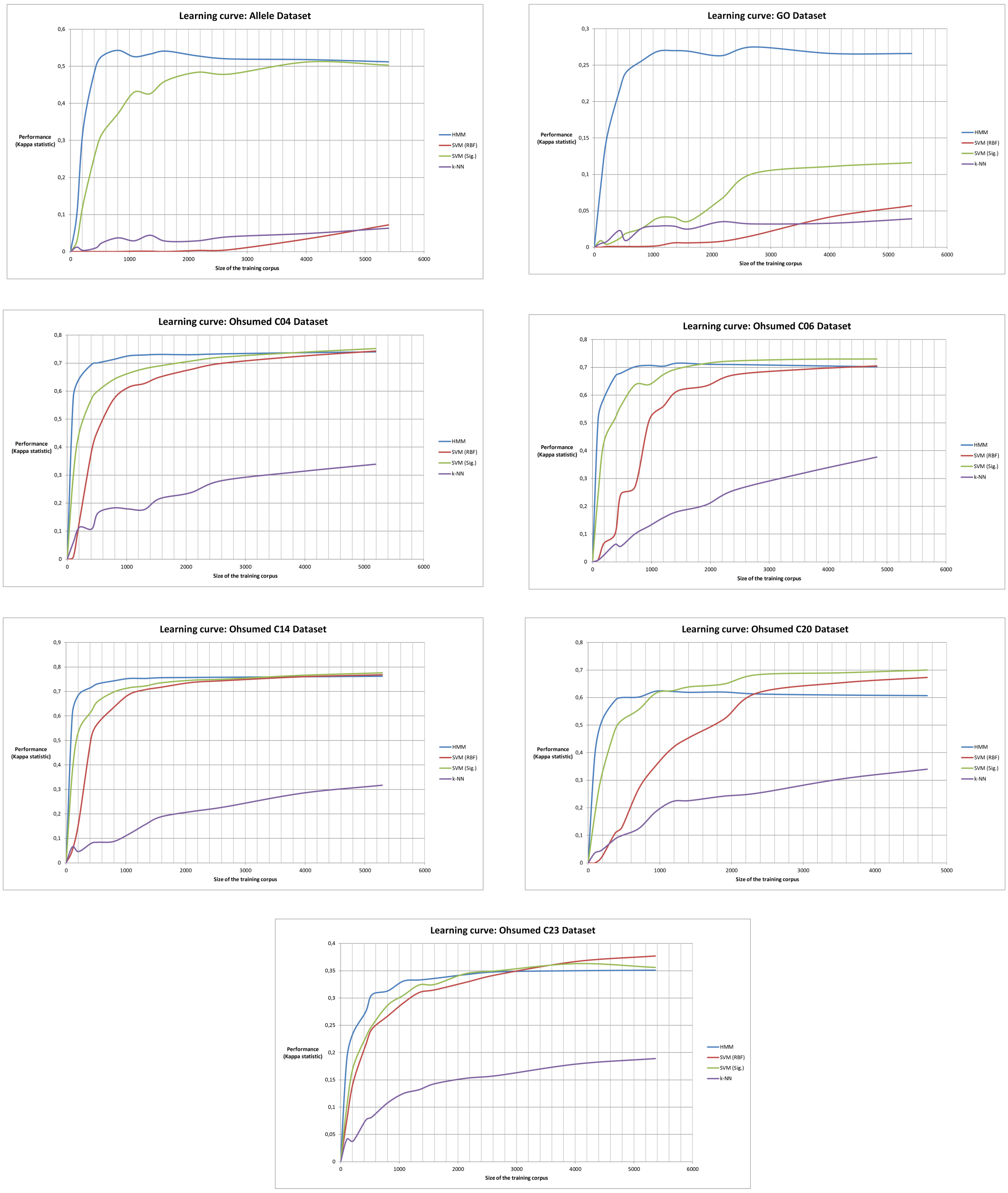


Fig. 5. Learning curves for the classifiers in all the tested corpus.

in its structure. T-LHMM is based on a linear state HMM with self-loops, while T-HMM has a left-right HMM without self-transitions. The learning process in

T-HMM is fixed by a static formula with a parameterization factor; however, in T-LHMM the proposed formula is used to create the initial input for the Baum-Welch algorithm, which is an HMM specific expectation-maximization algorithm for training purposes. This allows the T-LHMM to be more independent and reduce the parameterization cost.

The experimental and statistical results of this study show that the proposed HMM-based text classifier is less sensitive to the class imbalance, the size of the document corpora and the vocabulary (number of feature words) than the common classifiers. In addition, T-HMM is more efficient in terms of running time than SVM and k-NN techniques.

Acknowledgements

This work has been funded from the European Union Seventh Framework Programme [FP7/REGPOT-2012-2013.1] under grant agreement n 316265, BIOCAPS, and the "Platform of integration of intelligent techniques for analysis of biomedical information" project (TIN2013-47153-C3-3-R) from Spanish Ministry of Economy and Competitiveness.

References

1. Sebastiani F. Text categorization. In: Text Mining and its Applications to Intelligence, CRM and Knowledge Management. WIT Press; 2005. p. 109–129.
2. Joachims T. Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Norwell, MA, USA: Kluwer Academic Publishers; 2002.
3. Ko Y, Seo J. Text classification from unlabeled documents with bootstrapping and feature projection techniques. Inf Process Manage. 2009;45(1):70–83.
4. Sebastiani F. Machine learning in automated text categorization. ACM Comput Surv. 2002;34(1):1–47.
5. Nikolaos T, George T. Document classification system based on HMM word map. In: Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology. CSTST '08. New York, NY, USA: ACM; 2008. p. 7–12.
6. Janecek AG, Gansterer WN, Demel MA, Ecker GF. On the Relationship Between Feature Selection and Classification Accuracy. In: JMLR: Workshop and Conference Proceedings 4; 2008. p. 90–105.
7. Jolliffe IT. Principal Component Analysis. Springer Series in Statistics. Springer; 2002.
8. Japkowicz N. The Class Imbalance Problem: Significance and Strategies. In: In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI; 2000. p. 111–117.
9. Japkowicz N, Stephen S. The class imbalance problem: A systematic study. Intell Data Anal. 2002;6(5):429–449.
10. Yang Y, Liu X. A Re-Examination of Text Categorization Methods. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. SIGIR '99. New York, NY, USA: ACM; 1999. p. 42–49.

11. Borrajo L, Romero R, Iglesias EL, Marey CMR. Improving imbalanced scientific text classification using sampling strategies and dictionaries. Journal of Integrative Bioinformatics. 2011;8(3).
12. Gu B, Hu F, Liu H. Modelling Classification Performance for Large Data Sets. In: Proceedings of the Second International Conference on Advances in Web-Age Information Management. WAIM '01. London, UK, UK: Springer-Verlag; 2001. p. 317–328.
13. Provost F, Jensen D, Oates T. Efficient Progressive Sampling. In: In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining. ACM Press; 1999. p. 23–32.
14. Seara Vieira A, Iglesias EL, Borrajo L. T-HMM: A Novel Biomedical Text Classifier Based on Hidden Markov Models. In: 8th International Conference on Practical Applications of Computational Biology & Bioinformatics. vol. 294 of Advances in Intelligent Systems and Computing. Springer; 2014. p. 225–234.
15. Rabiner LR. Readings in speech recognition. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1990. p. 267–296.
16. Barros FA, Silva EFA, Cavalcante Prudêncio RB, Filho VM, Nascimento ACA. Combining Text Classifiers and Hidden Markov Models for Information Extraction. International Journal on Artificial Intelligence Tools. 2009;18(2):311–329.
17. Freitag D, Mccallum AK. Information Extraction with HMMs and Shrinkage. In: In Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction; 1999. p. 31–36.
18. Leek TR. Information extraction using hidden Markov models. UC San Diego; 1997.
19. Miller DRH, Leek T, Schwartz RM. A hidden Markov model information retrieval system. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. SIGIR '99. New York, NY, USA: ACM; 1999. p. 214–221.
20. Frasconi P, Soda G, Vullo A. Hidden Markov Models for Text Categorization in Multi-Page Documents. Journal of Intelligent Information Systems. 2002;18:195–217.
21. Li K, Chen G, Cheng J. Research on Hidden Markov Model-based Text Categorization Process. International Journal of Digital Content Technology and its Application. 2011;5(6):244–251.
22. Manne S, Fatima SS. An extensive empirical study of feature terms selection for text summarization and categorization. In: Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology. CCSEIT '12. New York, NY, USA: ACM; 2012. p. 606–613.
23. Yi K, Beheshti J. A hidden Markov model-based text classification of medical documents. Journal of Information Science. 2009;35(1):67–81.
24. Chen D, Liu Z. A new text categorization method based on HMM and SVM. vol. 7; 2010. p. V7383–V7386.
25. Baeza-Yates RA, Ribeiro-Neto B. Modern Information Retrieval. Addison-Wesley Longman; 1999.
26. Hersh W, Cohen A, Yang J, Bhupatiraju RT, Roberts P, Hearst M. TREC 2005 genomics track overview. In: In TREC 2005 notebook; 2005. p. 14–25.
27. Hersh WR, Buckley C, Leone TJ, Hickam DH. OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research. In: SIGIR; 1994. p. 192–201.
28. Lovins JB. Development of a stemming algorithm. Mechanical Translation and Computational Linguistics. 1968;11:22–31.

29. Witten IH, Frank E. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Series in Data Management Sys. Morgan Kaufmann; 2005.
30. Caporaso JG, Jr WAB, Cohen KB, Johnson HL, Paquette J, Hunter L. Concept Recognition and the TREC Genomics Tasks. In: Voorhees EM, Buckland LP, editors. TREC. vol. Special Publication 500-266. National Institute of Standards and Technology (NIST); 2005. .
31. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research. 2002;16:321–357.
32. Chang C, Lin C. LIBSVM: A library for support vector machines. ACM Trans Intell Syst Technol. 2011;2(3):27:1–27:27.
33. Sierra Araujo B. Aprendizaje automático: conceptos básicos y avanzados: aspectos prácticos utilizando el software Weka. Pearson Prentice Hall; 2006.
34. Viera AJ, Garrett JM. Understanding interobserver agreement: the kappa statistic. Family Medicine. 2005;37(5):360–3.