Expert Systems with Applications 40 (2013) 7184–7192

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

# An HMM-based over-sampling technique to improve text classification

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

*Computer Science Dept., Univ. of Vigo, Escuela Superior de Ingeniería Informática, Campus Universitario As Lagoas, 32004 Ourense, Spain*
*Instituto de investigación Biomédica de Vigo (IBIV), Hospital Meixoeiro – Edificio Anexo, Lg. do Meixoeiro s/n, 36214 Vigo, Spain*

ARTICLE INFO

*Keywords:*
Hidden Markov Model
Text classification
Oversampling techniques

ABSTRACT

This paper presents a novel over-sampling method based on document content to handle the class imbalance problem in text classification. The new technique, COS-HMM (Content-based Over-Sampling HMM), includes an HMM that is trained with a corpus in order to create new samples according to current documents. The HMM is treated as a document generator which can produce synthetical instances formed on what it was trained with.

To demonstrate its achievement, COS-HMM is tested with a Support Vector Machine (SVM) in two medical documental corpora (OHSUMED and TREC Genomics), and is then compared with the Random Over-Sampling (ROS) and SMOTE techniques. Results suggest that the application of over-sampling strategies increases the global performance of the SVM to classify documents. Based on the empirical and statistical studies, the new method clearly outperforms the baseline method (ROS), and offers a greater performance than SMOTE in the majority of tested cases.



## 1. Introduction

Text classification is the task of automatically organizing a document corpus into a predefined set of classes or topics (Sebastiani, 2005). A large number of text classification techniques have been developed, including Naive Bayes, k-Nearest Neighbours (kNN), and Support Vector Machines (SVMs), which have proven their effectiveness in many scenarios (Joachims, 2002; Ko & Seo, 2009; Sebastiani, 2002). However, these techniques must be improved to ensure their efficiency when applied over datasets with imbalanced class distributions.

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 (Japkowicz, 2000; Japkowicz & Stephen, 2002). The performance of standard classifiers such as those mentioned above 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 (López, Fernández, Moreno-Torres, & Herrera, 2012; Yang & Liu, 1999).

Over-sampling techniques provide a solution to handle the class imbalance problems. These methods can be used to increase the number of instances of rare classes by randomly selecting and duplicating instances (Japkowicz, 2000). A simple over-sampling approach to equalize class membership consists of randomly selecting and duplicating examples from the underrepresented classes until they are balanced. This method is called Random Over-Sampling (ROS) (Maimon & Rokach, 2005; Tan, Steinbach, & Kumar, 2006).

Although standard over-sampling techniques can decrease the impact of imbalanced classes, they come with some side effects. Duplicated samples can result in overfitting and hinder the classification performance. Some techniques like SMOTE (Synthetic Minority Over-sampling Technique) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) avoid this problem to a certain extent by adding a random Gaussian noise to the generated samples, creating new "synthetic" examples based on similar instances rather than simply adding samples by replacement. However, most over-sampling methods were built to deal with general datasets rather than to work specifically with text (Chen, Lin, Xiong, Luo, & Ma, 2011).

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 (Rabiner, 1990). However, their application has recently been extended to fields of text processing such as information extraction (Freitag & Mccallum, 1999; Leek, 1997), information retrieval (Miller, Leek, & Schwartz, 1999), text categorization (Frasconi, Soda, & Vullo, 2002; Li, Chen, & Cheng, 2011), and text classification (Yi & Beheshti, 2009).

In this paper, we propose an over-sampling HMM technique based on the content of a document corpus. HMM is used as a

* Corresponding author at: Computer Science Dept., Univ. of Vigo, Escuela Superior de Ingeniería Informática, Campus Universitario As Lagoas, 32004 Ourense, Spain. Tel.: +34 988387019; fax: +34 988387001.

*E-mail addresses:* eva@uvigo.es (E.L. Iglesias), asvieira@uvigo.es (A. Seara Vieira), lborrajo@uvigo.es (L. Borrajo).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.07.036

document generator, which allows the model to generate new synthetical samples based on the previous documents. The proposed mechanism, named COS-HMM (Content-based Over-Sampling HMM), is then compared with standard over-sampling techniques using a Support Vector Machine to classify biomedical text corpora (OHSUMED and TREC Genomics). The remainder of the paper is organized as follows. The basic concepts of HMM are described in Section 2. In Section 3 the proposed technique is presented. The analysis of experimental tests is shown in Section 4, and finally Section 5 contains the most relevant conclusions.

## 2. Material and methods

### 2.1. Hidden Markov Models definition

A Hidden Markov Model is a statistical tool used to model generative sequences that can be characterised by an underlying hidden process (Rabiner, 1990; Stamp, 2004). It can be seen as a state diagram that consists of a set of states and transitions between them, where each state can emit an output observation with a certain probability. Thus, HMMs are considered a "double random process". The first process describes the unobservable state sequence, i.e. the hidden layer represented by the state transition probability. The second process links each state to observations, creating the observable layer represented by the output observation probability (Li et al., 2011).

The formal definition of an HMM is as follows:

$$\lambda = \{N, V, A, B, \pi\}$$

1. $N$ is the number of states in HMM model. The state set is denoted by

$$S = \{s_0, s_1, \ldots, s_N\}$$

2. $V$ is the set of possible observations $V = \{v_0, v_1, \ldots, v_M\}$, where $M$ is the number of observations. We define $Q$ as a fixed state sequence of length $T$, and its corresponding observation sequence as $O$:

$$Q = \{q_0, q_1, \ldots, q_T\}$$
$$O = \{o_0, o_1, \ldots, o_T\}$$

3. $A$ is the state transition probability matrix of dimension $N$x$N$. It stores the probability of state $j$ following the state $i$ in the $a_{ij}$ cell:

$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$$

4. $B$ is the output observation probability matrix of dimension $N$x$M$. We define $b_i(v_k)$ as the probability of observation $v_k$ being produced at state $s_i$, which is independent (the probability) of time instant $t$.

$$b_i(v_k) = P(o_t = v_k | q_t = s_i)$$

5. $\pi$ is the initial state probability array.

$$\pi = (\pi_0, \pi_1, \ldots, \pi_N)$$

$$\pi_i = P(q_0 = s_i)$$

There are two important assumptions made by the model (Li et al., 2011). The first is called the *Markov assumption* and specifies that each state is dependent only on the previous one when calculating the state transition probability matrix.

The second assumption states that the output observation probability depends only on the current state of the model, i.e. it is independent of previous observations.

### 2.2. Common calculations

There are three problems that must be solved with an HMM for the model to be useful in a real application (Rabiner, 1990; Stamp, 2004):

1. Evaluation: Given an HMM model $\lambda$ and an observation sequence $O = \{o_0, o_1, \ldots, o_T\}$, we would like to compute $P(O|\lambda)$, i.e. the probability of $O$ being produced by $\lambda$. This problem is solved by the forward–backward algorithm.
2. Decoding: Given an HMM model $\lambda$ and an observation sequence $O = \{o_0, o_1, \ldots, o_T\}$, we would like to discover the hidden state sequence in $\lambda$ that most likely produces $O$. This problem is solved by using the Viterbi algorithm.
3. Learning: Given an HMM model $\lambda$ and an observation sequence $O = \{o_0, o_1, \ldots, o_T\}$, we would like to adjust the model parameters to maximize $P(O|\lambda)$. In short, given a set of examples from a process, we want to estimate the model $\lambda$ that best describes the process. This problem can be solved by using the Baum–Welch algorithm. In our approach, this model is used to adjust the parameters to the learning of document sets.

For further information and details on the principles and algorithms of HMM, see (Rabiner, 1990).

## 3. Over-sampling with HMM

Over-sampling techniques increase the number of instances of a certain class to deal with class imbalance. The proposed approach in this paper generates new instances of underrepresented classes based on document content. In order to handle the input documents, their structure must first be defined.

The most common approach to represent documents in text classification tasks is the bag-of-words approach (Nikolaos & George, 2008). In this case, every document is represented by a vector where elements describe the weight or relevance of the words in the document. Usually, the weight represents the frequency (number of occurrences) of that word in the document, but it may be adjusted or simplified, for example, denoting the word appearance with a binary representation (0,1) (Baeza-Yates & Ribeiro-Neto, 1999).

The time and space complexities of text classification depend on the size of the document vectors. In order to reduce their dimension, a text preprocessing step is required, where rare words and those which do not provide any useful information (such as prepositions, determiners or conjunctions) are removed. This step is further explained in the Experiments Section, where some adjustments such as Tf-Idf are also performed in the word weight. The final words selected to represent the documents are called feature words (Baeza-Yates & Ribeiro-Neto, 1999).

Fig. 1 shows a preprocessed corpus. It is important to note that each document $d_j$ has a weight for every feature word $v_i$ that is present in the corpus, even if it does not appear in the document. In that case, the word weight has a zero value. This is considered when building an over-sampling strategy, as only the non-zero valued words are taken into account to represent the documents.

As a probabilistic model, an HMM can help the task of creating new instances based on the document corpus content. The HMM is treated in our approach as a word generator. The set of possible observations is associated with the feature word set. The model is trained with the documents that belong to a certain class in order to guide the random process of generating word sequences to create new documents of the class.

The complete architecture of the proposed method is described in the next Section.

| | $v_0$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | Class |
|---|---|---|---|---|---|---|---|
| $d_0$ | 0 | 4.4 | 0 | 2.1 | 0 | 3.1 | R |
| $d_1$ | 0 | 3 | 1.2 | 2.9 | 0 | 0 | R |
| $d_2$ | 0 | 0 | 4 | 0 | 0 | 2.4 | R |
| $d_3$ | 1.1 | 2 | 0 | 3.5 | 4.8 | 0 | R |
| $d_4$ | 4.2 | 0 | 0 | 2.1 | 3.3 | 1.4 | N |
| $d_5$ | 3.2 | 1.2 | 0 | 0 | 0 | 0 | N |
| $d_6$ | 1.5 | 4 | 0 | 0.4 | 0 | 0 | N |
| $d_7$ | 0 | 4.1 | 0 | 1.1 | 0 | 3.3 | N |

**Fig. 1.** Example of a preprocessed corpus with eight instances (documents) and six feature words. The last column represents the class (R: relevant, N: Non-relevant) of each document.

### 3.1. Architecture

As stated before, the goal of this research is to create a mechanism that can generate new instances (documents) of a particular class based on the documents in the corpus, resulting in a new corpus that does not present imbalance problems.

Fig. 2 shows the architecture of the novel over-sampling technique. For each class in the corpus, a different over-sampling model is created. Each model is only trained with documents that belong to the class they represent.

Taking as input a document set $D_c = \{d_0, d_1 \ldots d_x\}$that belongs to class $c$, an over-sampling model $R_c$ for this class is defined. The model can create any number of new instances (documents) based on $D_c$ as the union of two generative processes (see Fig. 3):

1. Word generation
2. Word weight generation

Both parts constitute the spawning process of a new document $d'$ which belongs to class $c$, and are explained above.

#### 3.1.1. Word generation

Firstly, the word set of the new document $d'$ needs to be created. The word generation problem is handled by a Hidden Markov Model, which in this case is treated as a word sequence generator.

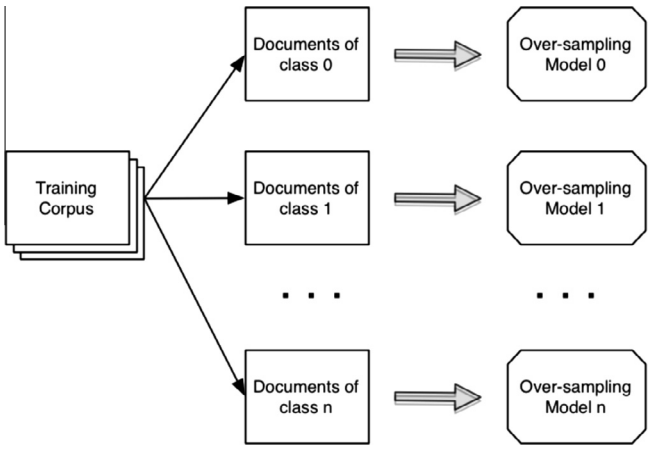


**Fig. 2.** Architecture of the COS-HMM model.

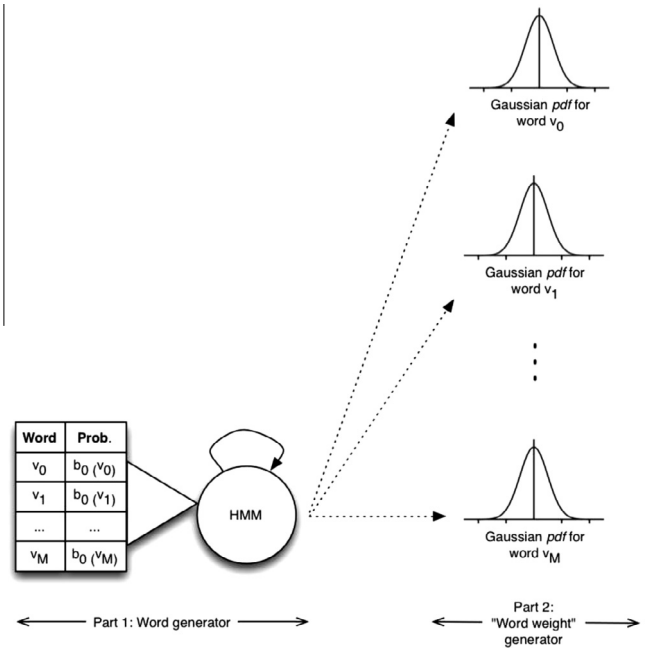


**Fig. 3.** The COS-HMM over-sampling model for a class $c$. In this case, the HMM that handles the word generation problem consists of a unique state with a self-transition.

The parameters of the HMM are adjusted in order to guide the random process of spawning a word sequence.

Given a document set $D_c$, the word generator HMM for the class $c$ is defined as:

1. The words from the corpus are taken as the set of observations $V$. For each word, there is an observation $v_k$.
2. States represent different information sources or topics in a same class. If there are several parts of the document in which the word relevance or the vocabulary are different (or if we are using MESH terms, for example), then various states can be used to represent the information sources. In the case of having more than one state, the state transition probabilities represent the relevance of each source. The relevance value of an information source is taken as the transition probability from any state to the state that represents that source. The relevance is established by the user based on the corpus and determines what the importance of the source is. The higher the relevance value is, the higher the probability of generating a word from that source. Fig. 4 shows an example with three states for three different information sources. It is important to note that the relevance values are considered probability values, so they range from 0 to 1, and the sum of all values must be equal to 1. For the corpora used in this paper (where only the abstracts are processed as they are considered the only information source), the HMM consists of a unique state with a self-transition (see Fig. 3), which lets the model generate any word independently of other words previously generated.
3. The output observation probability distribution of each state $s_i$ depends on the document set $D_c$. A word/observation $v_k$ has a higher output probability if the word appears frequently in the document set. In the case of having different information sources, requiring more than one state, the document set $D_c$ is partitioned and formatted taking into account each source of information, as shown in Fig. 5. Each information source has the same feature set as the original corpus, but the word values are calculated taking into

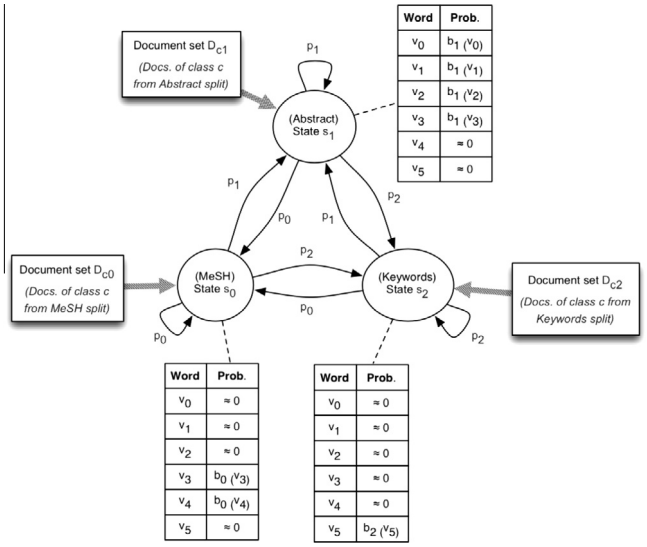


**Fig. 4.** Example of a HMM with three states. Each state represents a different information source: Abstract, MeSH terms and Keywords. In this example, the relevance of each source $i$ is defined by $p_i$, which is a value established by the user. The transition probability from any state to the state $s_i$ (that represents the source $i$) is equal to $p_i$.

account only the content of this information source. Specifically, given a document set $D_c$, the components of the output observation probability matrix for the state $s_i$ are defined as follows:

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

(a) $b_i(v_k)$ is the probability of the word/observation $v_k$ being emitted in the state $s_i$

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

(c) $N$ is the number of documents in the set $D_c$

Once the HMM is created and trained for a class, the word generation problem can be solved by inducing a random sequence of observations with the HMM. Observations are associated with the feature words, and each observation has a probability of being emitted depending on the document set, as stated in the calculation equation above. Therefore, while word generation has a random component, it is guided by the original corpus. Finally, the size of the set $G$ of new words is selected randomly between

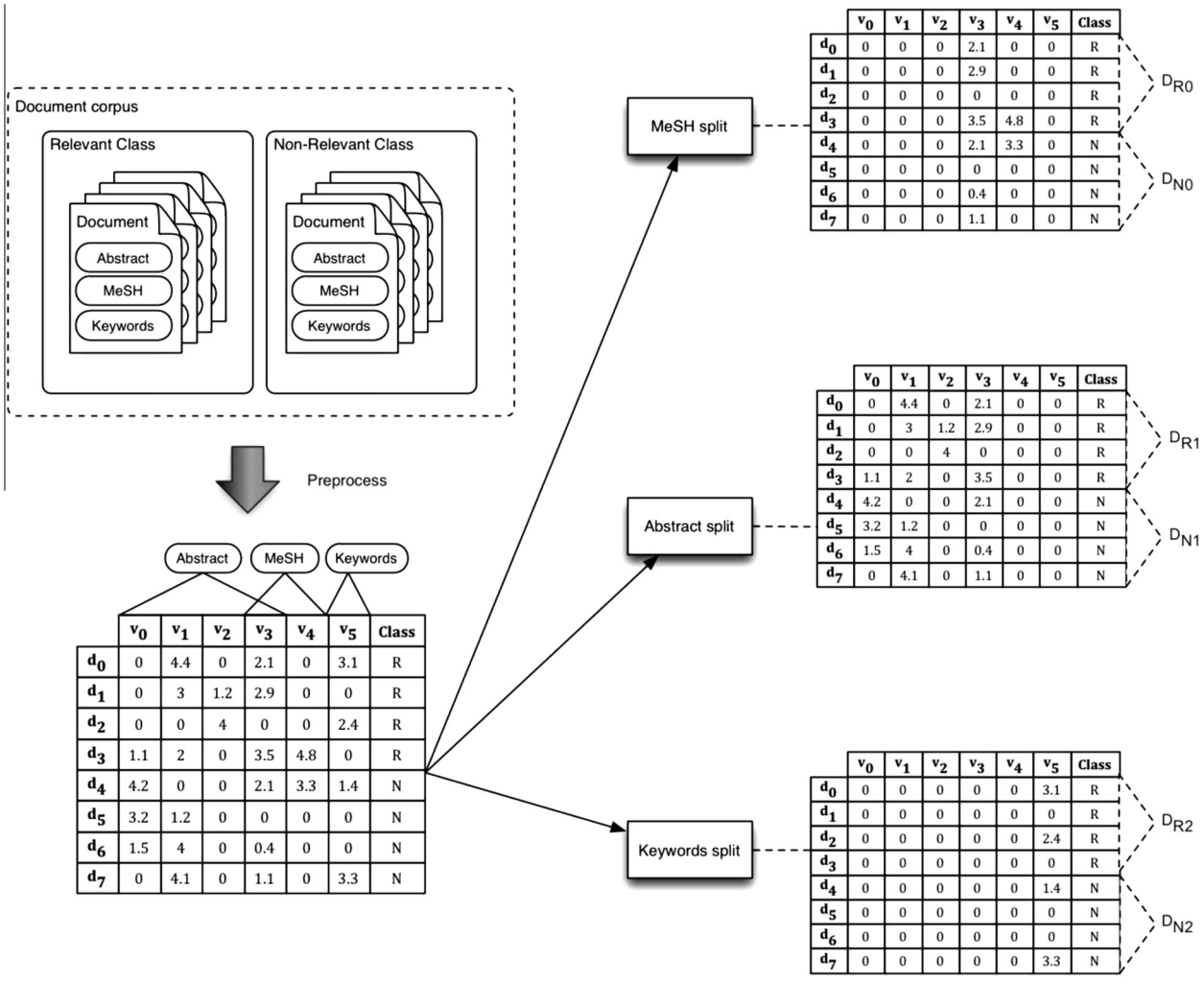


**Fig. 5.** Example of corpus split considering three different information sources: Abstract, MeSH and Keywords. Initially, the preprocessed corpus has the feature words from all the information sources. The complete feature word set conforms the observation set $V$ for every HMM built (a HMM is created for each class).

the sizes (number of feature words with non-zero values) of all documents belonging to the class.

*3.1.2. Word weight generation*

Once the word set $G$ is generated by the HMM, a weight must be assigned to each word $g_k$ based on the document set $D_c$. In the case of the corpus shown in the Experiments Section, this weight corresponds to the adjusted word frequency in the document (Baeza-Yates & Ribeiro-Neto, 1999).

In order to be able to assign a new weight $w_k \in \Re$ for each generated word $g_k$, a Gaussian probability density function (pdf) is built. It describes the values of each feature word $v_k$ in the document set $D_c$. Gaussian pdfs are adjusted to the non-zero values that $v_k$ has in $D_c$.

Taking the example from the Fig. 1 as the initial corpus, and the class $R$ (relevant) as the over-sampling model class, six Gaussian pdfs are built, one for each feature word: $\{v_0, v_1, \ldots, v_5\}$. The set comprises four documents: $D_c = \{d_0, d_1, d_2, d_3\}$. Each Gaussian pdf has to fit to the non-zero values in the documents for the word it was built for. For example, the Gaussian pdf for the word $v_0$ is adjusted with the list [1.1], the Gaussian pdf for the word $v_1$ is adjusted with the list [4.4;3.0;2.0], and so on (see Fig. 6).

The meaning of "fit" or "adjust" should be *have the maximum likelihood*. The Gaussian pdf mean and variance are modified with the word weights to increase the probability of outputting a similar value when a random real value is generated. The algorithm used to adjust the pdfs is explained by Rabiner (1990) in Eqs. (53) and (54). These formulas fit a Gaussian pdf to a balanced (non empty) observation set. All the words are assigned with the same weight and they are used as the observation set to adjust the pdfs.

Once the pdfs are created, the weight generation problem can be solved as follows: for each word $g_k$ in $G$ a weight $w_k$ is generated with the Gaussian pdf that represents this word in $D_c$. The random generation process with a certain Gaussian function is handled by the Box-Muller polar method (Box & Muller, 1958).

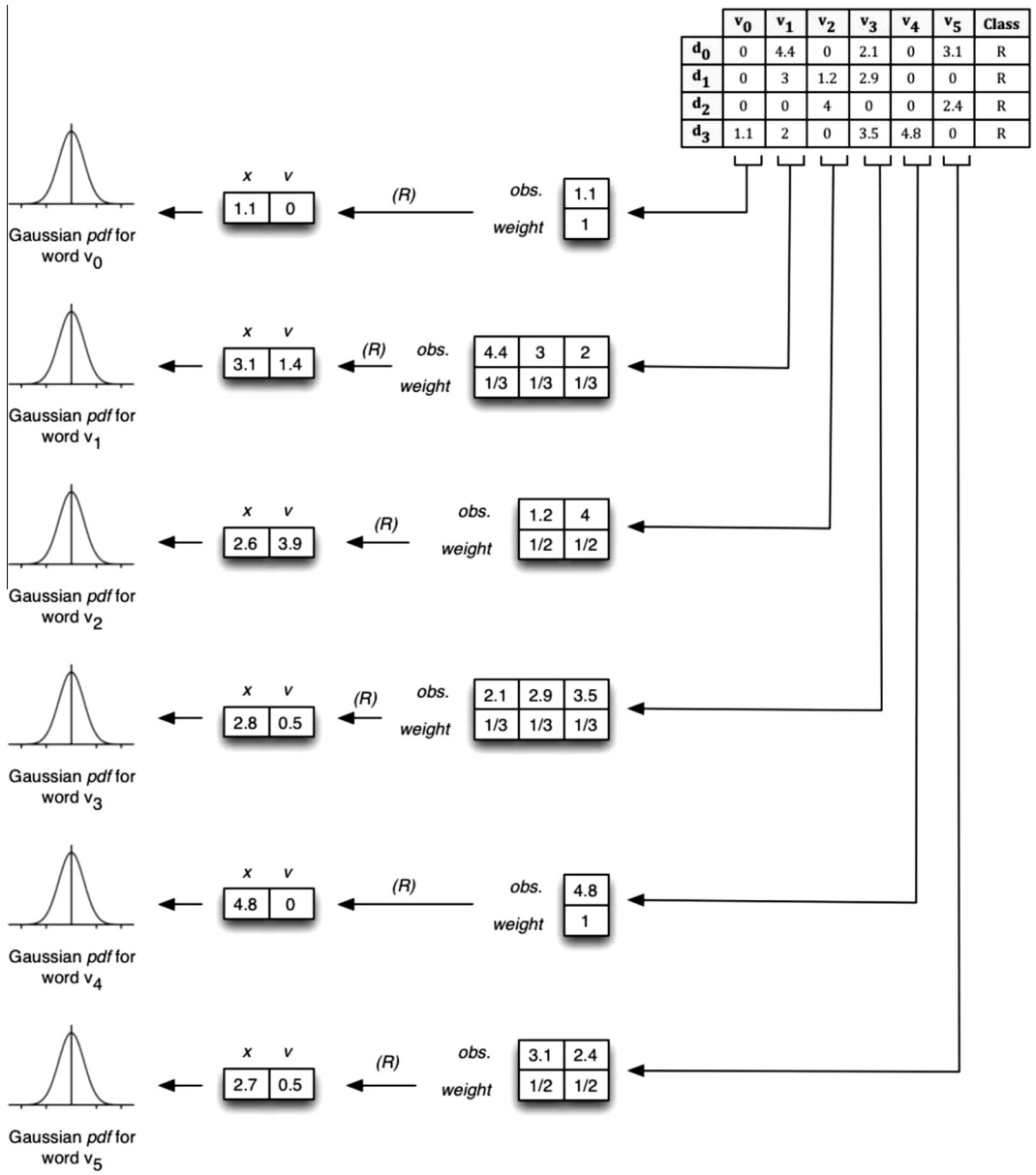


**Fig. 6.** Gaussian pdfs for the relevant documents of the Fig. 1. $x$ denotes the Gaussian pdf mean, $v$ is the Gaussian pdf variance, and ($R$) stands for the Rabiner method.

Considering that the entire over-sampling approach $R_c$ for class $c$ has been built with the document subset $D_c$ following the previously mentioned steps, the complete process of generating a new instance (document) is resumed in the Algorithm 1:

**Algorithm 1.** Generate a new document $d'$ to balance the class $c$.

**input**: A trained over-sampling model $R_c$ for the class $c$
**output**: A new document $d'$ that belongs to the class $c$
$M$ = number of feature words in the corpus;
$d'$ = vector of size $M$;
*Initialize $d'$ with 0 weight for the M feature words*;
$G$ = empty set {};
$doc_{random}$=a randomly choosen document from $D_c$;
$r$ = size of $doc_{random}$ (number of feature words with non-zero value);
**repeat**
  $x$ = random word generated with the HMM built from $R_c$;
  $G = G \cup x$;
**until** $|G| = r$;
**foreach** $x_k$ in $G$ **do**
  $w_k$=weight generated with the Gaussian pdf for the word $x_k$;
  $d'[i]=w_k$, where $i$ is the index of the word $x_k$,$i$:=0, ... , $(M-1)$;
**end**
**return** $d'$

As mentioned at the beginning of this section, a distinct over-sampling model is built for each imbalanced or wanted class. The sample generation process can be repeated indefinitely, so any number of documents can be generated for a class.

## 4. Calculation

Fig. 7 represents the workflow followed in the COS-HMM over-sampling approach. Initially, the corpus is preprocessed applying the steps described above, resulting in a matrix with the same structure as that previously defined in Fig. 1. This corpus is used as a base for the over-sampling method. The process generates new instances to be added to the original corpus in order to solve the imbalance problem (creating more instances for the imbalanced classes). The final corpus is ready to be used on a classifier, which in this case is a Support Vector Machine (SVM).

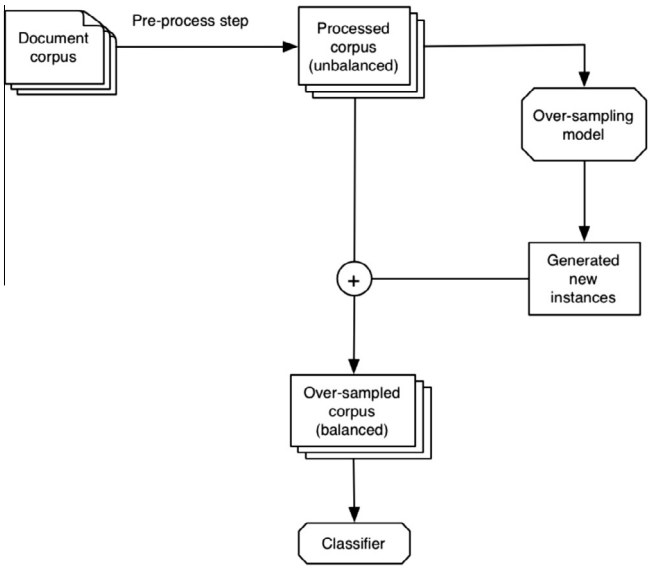


**Fig. 7.** Workflow that represents the application of an over-sampling method in a document corpus.

The goal of the experiments is to test the performance of the proposed over-sampling technique in two unbalanced document corpora, TREC Genomics and OHSUMED. Collections are described in the next sections, and they are used in a classification process with an SVM. In order to compare the effectiveness of the proposed over-sampling technique, the two corpora are also classified with a SVM without solving the unbalance problem. The same classifying process is applied with the over-sampled collections from the COS-HMM system, the ROS and the SMOTE models.

### 4.1. OHSUMED corpus

The OHSUMED test collection, initially compiled by Hersh, Buckley, Leone, and Hickam (1994), is a subset of the MEDLINE database, which is 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). Six categories are chosen to build the corpus used in this paper: C02 (Virus), Neoplasms (C04), Digestive (C06), Cardio (C14), Immunology (C20) and Pathology (C23). Additionally, if a document was assigned to two or more categories, the document is excluded from the final set. This is a step required to avoid the appearance of identical documents with different associated classes, as the classifier is not meant to manage multi-labeled corpus.

### 4.2. TREC Genomics corpus

One of the tasks in TREC Genomics 2005 Track (Hersh et al., 2005) 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 proposed model. A different corpus is created for each category, where documents can be classified as relevant or non-relevant.

### 4.3. Data processing

Once the documents in each corpus are organized, we must format them into a vector of feature words in which elements describe the word occurrence frequencies. All the different words that appear in the corpus are candidates for input features.

Standard text preprocessing techniques are used in order to reduce the input feature size to train the classifier. A predefined list of stopwords (common English words) is removed from the text, and a stemmer based on the Lovins stemmer (Lovins, 1968) is applied. Then, words occurring in less than ten documents of the corpus are also removed. Additionally, the IG (Information Gain) feature selection algorithm is used to select the top 1,000 words as feature words.

Finally, a matrix representing each corpus is obtained. Rows correspond to documents and columns to feature words. The value (weight) of an element in a matrix is determined by the number of occurrences of that feature word (column) in the document (row). This value is adjusted using the Tf-Idf statistic in order to measure the word relevance. The application of Tf-Idf decreases the weight of terms that occur very frequently in the collection and increases the weight of terms that occur rarely (Baeza-Yates & Ribeiro-Neto, 1999).

Once the preprocess step is finished for each corpus, the following structures are obtained:

- For the OHSUMED corpus, the final matrix contains 9,460 documents which is randomly splitted into a train and a test datasets preserving the class proportion (with 4,730 instances each). The train dataset is used as the input to the over-sampling model to solve its unbalance problem. The content of the train corpus is showed in Table 1. The unbalanced classes in the original corpus are C02 (Virus), C06 (Digestive) and C20 (Immunologic).
- For the TREC corpus, the train and test split is applied following the Genomic Track guidelines (Hersh et al., 2005), ending up with 5,837 instances in the train dataset and 4,958 in test dataset for both corpus. The content of the train datasets for Allele and GO categories is shown in the Tables 2 and 3, respectively. In both cases, the imbalanced class in the original corpus is the class of relevant documents.

**Table 1**
Document distribution for the OHSUMED collection. The "Oversampled Corpus" column represents the target corpus after using an over-sampling method.

| Class | Original corpus | Oversampled corpus | |
|---|---|---|---|
| | Documents | Documents | Increase |
| C02 (Virus) | 125 | 962 | +837 |
| C04 (Neoplasms) | 1256 | 1256 | – |
| C06 (Digestive) | 419 | 962 | +543 |
| C14 (Cardio) | 1438 | 1438 | – |
| C20 (Immunologic) | 530 | 962 | +432 |
| C23 (Pathological) | 962 | 962 | – |
| Total | 4730 | 6542 | +1812 |

**Table 2**
Document distribution for the Allele TREC collection. The "Oversampled Corpus" column represents the target corpus after using an over-sampling method.

| Class | Original corpus | Oversampled corpus | |
|---|---|---|---|
| | Documents | Documents | Increase |
| Relevant | 338 | 5499 | +5160 |
| Non-relevant | 5499 | 5499 | – |
| Total | 5837 | 10998 | +5160 |

**Table 3**
Document distribution for the GO TREC collection. The "Oversampled Corpus" column represents the target corpus after using an over-sampling method.

| Class | Original corpus | Oversampled corpus | |
|---|---|---|---|
| | Documents | Documents | Increase |
| Relevant | 462 | 5375 | +4913 |
| Non-relevant | 5375 | 5375 | – |
| Total | 5837 | 10758 | +4913 |

### 4.4. Results and discussion

SVM is adopted as the base classifier in our evaluation experiments because of its good general performance and dependence of balanced classes (Li, Wang, & Bryant, 2009). The over-sampling strategies are applied in the previous process of training the classifier. The over-sampling methods used are ROS, SMOTE and the proposed model COS-HMM using only one state that represents the abstract of the documents, as stated in the Architecture Section.

It is important to note that the over-sampling method only affects the train dataset, as it is the only one which will be increased to solve the imbalance problems that appear when training a classifier with this kind of corpora. Tables 1–3 show the balance of an over-trained corpus obtained after applying an over-sampling technique (increasing the documents in imbalanced classes until the corpus is totally balanced).

As there is a random component in the over-sampling methods used (ROS, SMOTE and COS-HMM), more than one execution is needed to test their behaviour. Each technique is executed 10 times for each corpus, varying the initial train and test sets. Train and test splits are generated randomly in each test but retain the number of instances and the class distribution explained in the previous Section. Finally, a statistical test is applied over the 10 results for each method in order to be able to compare them.

The implementation of the SVM used in this study is LIBSVM (Chang & Lin, 2011) and the parameters are those utilized by default in WEKA environment (Sierra Araujo, 2006), applying a RBF kernel. Finally, the *F*-measure was selected as the evaluation measure to verify the effectiveness of the SVM classifier in each test set. This *F*-measure is a combination of recall ($R$) and precision ($P$) with an equal weight as follows:

$$F = \frac{2 \cdot R \cdot P}{R + P}$$

### 4.5. Experimental results

Tables 4–6 show the results from executing the OHSUMED, Allele and GO TREC collections, respectively. Values in the

**Table 4**
Results achieved for OHSUMED corpus with the over-sampling methods.

| Class | Pure SVM | ROS | SMOTE | COS-HMM |
|---|---|---|---|---|
| C02 (Virus) | 0.390 | 0.516 | 0.535 | 0.665 |
| C04 (Neoplasms) | 0.871 | 0.877 | 0.872 | 0.871 |
| C06 (Digestive) | 0.717 | 0.742 | 0.756 | 0.803 |
| C14 (Cardio) | 0.867 | 0.868 | 0.886 | 0.885 |
| C20 (Immunologic) | 0.735 | 0.751 | 0.746 | 0.781 |
| C23 (Pathological) | 0.714 | 0.725 | 0.721 | 0.720 |

**Table 5**
Results achieved for Allele TREC corpus with the over-sampling methods.

| Class | Pure SVM | ROS | SMOTE | COS-HMM |
|---|---|---|---|---|
| Relevant | 0.170 | 0.369 | 0.478 | 0.494 |
| Non-relevant | 0.976 | 0.976 | 0.975 | 0.950 |

**Table 6**
Results achieved for GO TREC corpus with the over-sampling methods.

| Class | Pure SVM | ROS | SMOTE | COS-HMM |
|---|---|---|---|---|
| Relevant | 0.000 | 0.251 | 0.284 | 0.410 |
| Non-relevant | 0.958 | 0.943 | 0.919 | 0.921 |

**Table 7**
Results for Student t-tests performed on the collections of F-measure values acquired on the 10 executions of the ROS and COS-HMM methods.

| Class | ROS | | COS-HMM | | *t*-value | *pnull* |
|---|---|---|---|---|---|---|
| | Mean | Sdev. | Mean | Sdev. | | |
| C02 (OHSUMED) | 0.516 | $3.649 \cdot 10^{-2}$ | 0.665 | $2.309 \cdot 10^{-2}$ | 10.90 | <0.0001 |
| C06 (OHSUMED) | 0.742 | $1.945 \cdot 10^{-2}$ | 0.803 | $9.921 \cdot 10^{-3}$ | 8.81 | <0.0001 |
| C20 (OHSUMED) | 0.751 | $1.222 \cdot 10^{-2}$ | 0.781 | $1.458 \cdot 10^{-2}$ | 4.95 | <0.0001 |
| Relevant (Allele) | 0.369 | $2.854 \cdot 10^{-2}$ | 0.494 | $1.022 \cdot 10^{-2}$ | 13.10 | <0.0001 |
| Relevant (GO) | 0.251 | $1.890 \cdot 10^{-2}$ | 0.410 | $8.547 \cdot 10^{-3}$ | 24.20 | <0.0001 |

*null-hypothesis*: There is no difference between models
*pnull*: Probability of assuming the *null-hypothesis*

**Table 8**
Results for Student t-tests performed on the collections of F-measure values acquired on the 10 executions of the SMOTE and COS-HMM techniques.

| Class | SMOTE | | COS-HMM | | *t*-value | *pnull* |
|---|---|---|---|---|---|---|
| | Mean | Sdev. | Mean | Sdev. | | |
| C02 (OHSUMED) | 0.535 | $2.607 \cdot 10^{-2}$ | 0.665 | $2.309 \cdot 10^{-2}$ | 11.80 | <0.0001 |
| C06 (OHSUMED) | 0.756 | $1.304 \cdot 10^{-2}$ | 0.803 | $9.921 \cdot 10^{-3}$ | 8.98 | <0.0001 |
| C20 (OHSUMED) | 0.746 | $1.730 \cdot 10^{-2}$ | 0.781 | $1.458 \cdot 10^{-2}$ | 4.80 | <0.0001 |
| Relevant (Allele) | 0.478 | $2.530 \cdot 10^{-2}$ | 0.494 | $1.022 \cdot 10^{-2}$ | 1.90 | 0.073 |
| Relevant (GO) | 0.284 | $1.443 \cdot 10^{-2}$ | 0.410 | $8.547 \cdot 10^{-3}$ | 23.60 | <0.0001 |

*null-hypothesis*: There is no difference between models
*pnull*: Probability of assuming the *null-hypothesis*

over-sampling method columns correspond to the average *F*-measure resulting from the 10 executions of the techniques.

In general, the proposed method outperforms the ROS and SMOTE techniques and clearly improves the effectiveness of the SVM classification in the entire tested corpus.

For the OHSUMED corpus, every class shows a better performance when the proposed model is applied in the train set. When an over-sampling method increases the imbalanced classes, the classifier performance on the remainder of the classes may also be affected, depending on how similar the documents of the different classes are. In this case, only the imbalanced classes show significant improvement in performance, while the others remain stable.

For the TREC Genomics corpus, the imbalance problem is severe in the relevant class. This is reflected in the pure SVM execution, where the *F*-measure achieved is minimal and equal to zero in the case of the GO category. The relevant documents have very few differences with the non-relevant ones, which makes the classification process a challenging task. As stated in the Genomic Track Overview (Hersh et al., 2005), most of the implemented techniques focus on the preprocess step of feature selection to improve the results on this corpus. However, COS-HMM is capable of slightly increasing the performance of the SVM without using non-standard preprocessing techniques such as extended medical vocabulary or gene synonymous.

### 4.6. Statistical test

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 true performance from a given test set.

A Student *t*-test is performed on the collection of *F*-measures achieved by ROS, SMOTE and COS-HMM in order to prove their differences. The previously mentioned results show that the proposed model has a higher mean of *F*-measure values than ROS and SMOTE methods.

Table 7 shows the results for the executed *t*-tests comparing the ROS method and the proposed model. We perform a test for the collection results achieved in each imbalanced class in OHSUMED, Allele, and GO TREC corpora. The *null-hypothesis* (the difference is due to chance) is rejected when its probability (*pnull*) is too small (less than 0.05). In this case, it is rejected in every single test, proving that the proposed technique offers greater performance than the ROS technique in the tested corpus since it has a higher mean value.

On the other hand, Table 8 shows the results achieved in the executed *t*-tests comparing the COS-HMM and SMOTE techniques. The achieved values show that the proposed model outperforms SMOTE in the entire tested corpus except one: the Allele corpus. In this case, the *t*-value indicates that the two over-sampling approaches offer a similar performance and the difference between them cannot be demonstrated statistically.

## 5. Conclusions

In this paper, we propose an over-sampling method based on the document content to handle the class imbalance problem in text classification. The model includes an HMM that is trained with the corpus in order to create new samples based on previous existing documents. The HMM is a document generator which can produce synthetical instances based on what it was trained with.

The proposed technique is tested with a Support Vector Machine (SVM) classifier in two medical document collections (OHSUMED and TREC Genomics) and then compared with the Random Over-Sampling (ROS) and the SMOTE techniques. The results show that the application of any of these over-sampling methods increases the global performance of the SVM. Additionally, based on the statistical results, the new method clearly outperforms the baseline method (ROS), and offers a greater performance than SMOTE in the majority of tested cases.

## Acknowledgment

This work has been funded from the European Union Seventh Framework Programme [FP7/REGPOT-2012–2013.1] under grant agreement n 316265, BIOCAPS.

## References

Baeza-Yates, R. A., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison-Wesley Longman.

Box, G. E. P., & Muller, M. E. (1958). A note on the generation of random normal deviates. *The Annals of Mathematical Statistics, 29*, 610–611.

Chang, C., & Lin, C. (2011). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology, 2*, 27:1–27:27.

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Chen, E., Lin, Y., Xiong, H., Luo, Q., & Ma, H. (2011). Exploiting probabilistic topic models to improve text categorization under class imbalance. *Information Processing and Management, 47*, 202–214.

Frasconi, P., Soda, G., & Vullo, A. (2002). Hidden markov models for text categorization in multi-page documents. *Journal of Intelligent Information Systems, 18*, 195–217.

Freitag, D., & Mccallum, A.K. (1999). Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 workshop on machine learning for information extraction* (pp. 31–36).

Hersh, W.R., Buckley, C., Leone, T.J., & Hickam, D.H. (1994). Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR* (pp. 192–201).

Hersh, W., Cohen, A., Yang, J., Teja Bhupatiraju, R., Roberts, P., & Hearst, M. (2005). Trec 2005 genomics track overview. In *TREC 2005 notebook* (pp. 14–25).

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 international conference on artificial intelligence (ICAI)* (pp. 111–117).

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analisys, 6*, 429–449.

Joachims, T. (2002). *Learning to classify text using support vector machines: methods, theory and algorithms*. Norwell, MA, USA: Kluwer Academic Publishers.

Ko, Y., & Seo, J. (2009). Text classification from unlabeled documents with bootstrapping and feature projection techniques. *Information Processing and Management, 45*, 70–83.

Leek, T.R. (1997). Information extraction using hidden Markov models.

Li, K., Chen, G., & Cheng, J. (2011). Research on hidden markov model-based text categorization process. *International Journal of Digital Content Technology and its Application, 5*, 244–251.

Li, Q., Wang, Y., & Bryant, S. (2009). A novel method for mining highly imbalanced high-throughput screening data in pubchem. *Bioinformatics, 25*, 3310–3316.

López, V., Fernández, A., Moreno-Torres, J. G., & Herrera, F. (2012). Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications, 39*, 6585–6608.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics, 11*, 22–31.

Maimon, O., & Rokach, L. (2005). *Data mining and knowledge discovery handbook*. *The Kluwer international series in engineering and computer science*. Springer.

Miller, D. R. H., Leek, T., & Schwartz, R. M. (1999). 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* (pp. 214–221). New York, NY, USA: ACM.

Nikolaos, T., & George, T. (2008). 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* (pp. 7–12). New York, NY, USA: ACM.

Rabiner, L. R. (1990). *Readings in speech recognition*. *A tutorial on hidden Markov models and selected applications in speech recognition*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. chapter.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*, 1–47.

Sebastiani, F. (2005). Text categorization. In *Text mining and its applications to intelligence, crm and knowledge management* (pp. 109–129). WIT Press.

Sierra Araujo, B. (2006). *Aprendizaje automático: conceptos básicos y avanzados: aspectos prácticos utilizando el software Weka*. Pearson Prentice Hall.

Stamp, M. (2004). A revealing introduction to hidden markov models. *Science*, 1–20.

Tan, P., Steinbach, M., & Kumar, V. (2006). Data mining: Introduction To. Boston.

Yang, Y., & Liu, X. (1999). 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* (pp. 42–49). New York, NY, USA: ACM.

Yi, K., & Beheshti, J. (2009). A hidden markov model-based text classification of medical documents. *Journal of Information Science, 35*, 67–81.