Accepted Manuscript

Please cite this article as: Seara Vieira, A., Borrajo, L., & Iglesias, E. L. (2016). Improving the text classification using clustering and a novel HMM to reduce the dimensionality. *Computer Methods and Programs in Biomedicine*, 136, 119-130. doi:10.1016/j.cmpb.2016.08.018

Link to published version: https://doi.org/10.1016/j.cmpb.2016.08.018

Improving the text classification using clustering and a novel HMM to reduce the dimensionality

A. Seara Vieira[a], L. Borrajo[a], E. L. Iglesias[a,*]

[a]*Department of Computer Science, Higher Technical School of Computer Engineering, University of Vigo, 32004 Ourense, Spain*

Abstract

In text classification problems, the representation of a document has a strong impact on the performance of learning systems. The high dimensionality of the classical structured representations can lead to burdensome computations due to the great size of real-world data. Consequently, there is a need for reducing the quantity of handled information to improve the classification process. In this paper, we propose a method to reduce the dimensionality of a classical text representation based on a clustering technique to group documents, and a previously developed Hidden Markov Model to represent them. We have applied tests with the k-NN and SVM classifiers on the OHSUMED and TREC benchmark text corpora using the proposed dimensionality reduction technique. The experimental results obtained are very satisfactory compared to commonly used techniques like InfoGain and the statistical tests performed demonstrate the suitability of the proposed technique for the preprocessing step in a text classification task.

Keywords: Hidden Markov Model, Text classification, Dimensionality Reduction, Document Clustering, Similarity-based classification

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 a document set to a predefined set of classes or topics [1].

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 representation suitable for the learning algorithm and the classification task. In a general 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 d_i in categories c_j, a classifier is trained to model the implicit relationship between the characteristics of the document and its class, in order to be able to accurately classify new documents.

To achieve this classification, input data need to be expressed in a format that classifiers can handle. The most common technique in document classification tasks is the *bag-of-words* approach

*Corresponding author

Email addresses: adrseara@uvigo.es (A. Seara Vieira), lborrajo@uvigo.es (L. Borrajo), eva@uvigo.es (E. L. Iglesias)

[2]. 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. While some simple document classification tasks can be accurately performed with vocabulary sizes numbering fewer than one hundred, many complex tasks on real-world data from the Web, UseNet and newswire articles do best with vocabulary sizes in the thousands.

The complexity of the text classification in terms of time and space depends on the size of the mentioned vectors. Generally, not all words have a significant contribution to effectively represent a document. Therefore, before generating the feature vectors, 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 [3]. Additionally, *feature reduction algorithms* can be applied to the initial feature word set in order to further 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* [4, 5]. Another type of feature reduction method creates new attributes as a combination of the original attributes, such as the *Principal Component Analysis* (PCA) technique [5, 6]. 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.

In order to represent a dataset, the *similarity-based paradigm* can also be used [7]. In this paradigm, objects (documents, in our case) are described using pairwise (dis)similarities, i.e. distances from other objects in the dataset. In this way, documents are not limited to being represented in a feature word space, and all that is needed is a way to compute distances between documents [8]. The goal of this paradigm is then to train and test a classifier using only these relational data.

Figure 1 shows an example of a typical document representation and a similarity-based representation. Given a base corpus (Fig. 1(a)), new datasets (Fig. 1(b)) can be transformed into a similarity-based representation (Fig. 1(c)) by calculating each pairwise similarity between documents. In this case, the number of synthetical attributes a_d_j of the transformed dataset is the number of documents in the base corpus. Each document is then represented by a vector of values that indicates the distance between this document and each additional document in the original base dataset.

The main problem of the similarity-based approach is the high dimensionality of the resulting similarity space [8]. In a basic approach, the dimensionality is equal to the cardinality of the base/training corpus, which can lead to computational problems in large datasets. In addition, the way in which distances between documents are measured also influences the whole process.

The system described in [8] uses Hidden Markov Models (HMMs) to measure the similarity between pairs of data objects. HMMs, which are commonly employed as probabilistic models of sequential data [9], seem to be particulary well suited to the similarity-based classification, as it can be seen as a natural extension of the standard HMM classification scheme. Specifically, the standard approach assigns an unknown sequence O to the class whose HMM shows the highest *likelihood*. This *likelihood-based* measure is used in [8] as the distance measure in the similarity-based paradigm. However, the system is not built for text classification purposes and the main contribution uses HMMs to compute pairwise similarities between single objects, thus training an HMM per representative sequence.

Following the idea of the similarity-based representation and the use of HMMs to measure distances, we propose a novel feature reduction technique called DR-HMM, which combines the

(a) **Base corpus /Training corpus**

	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	4.2	0	0	2.1	3.3	1.4	N
d_3	1.5	4	0	0.4	0	0	N

(b) **Corpus to transform**

	v_0	v_1	v_2	v_3	v_4	v_5	Class
d_4	0	0	4	0	0	2.4	?
d_5	1.1	2	0	3.5	4.8	0	?
d_6	3.2	1.2	0	0	0	0	?
d_7	0	4.1	0	1.1	0	3.3	?

(c) **Converted corpus**

	a_d_0	a_d_1	a_d_2	a_d_3	Class
d_4	$s(d_4,d_0)$	$s(d_4,d_1)$	$s(d_4,d_2)$	$s(d_4,d_3)$	?
d_5	$s(d_5,d_0)$	$s(d_5,d_1)$	$s(d_5,d_2)$	$s(d_5,d_3)$	?
d_6	$s(d_6,d_0)$	$s(d_6,d_1)$	$s(d_6,d_2)$	$s(d_6,d_3)$	?
d_7	$s(d_7,d_0)$	$s(d_7,d_1)$	$s(d_7,d_2)$	$s(d_7,d_3)$	?

Figure 1: Basic similarity-based corpus transformation. **(a)** Base/Training corpus using a typical bag-of-words representation. It contains six feature words $(v_0, ..., v_5)$. The *Class* attribute is binary and can take the values: Relevant (R) and Non-relevant (N) **(b)** New dataset to transform into a similarity-based representation **(c)** Dataset converted into a similarity-based representation based on the training corpus. Documents are represented in terms of distances/similarities (s function) to the original documents in the base corpus.

X-means clustering algorithm to reduce the size of the base dataset with an HMM-based document representer to compute the distances between documents. The main idea of the proposed method is to use groups of documents as the representation base instead of using single documents, so that new documents are represented by distances to groups of documents. In order to achieve that, the clustering algorithm is applied to create the groups/clusters of documents that are considered the base set of the new representation. In addition, the HMM to represent a document group is based on the previously developed T-HMM [10, 11].

This paper is organized as follows. Section 2 presents the basics of Hidden Markov Models. Section 3 presents the proposed feature reduction method. In Section 4, the experimental results with two classification methods, k-NN and SVM, are shown. Finally, the last Section presents the conclusions of this study.

2. Materials and Methods

2.1. Basics of Hidden Markov Models

A Hidden Markov Model is a statistical tool used to model generative sequences that can be characterised by an underlying hidden process [9, 12, 13]. It can be seen as a state diagram that consists of a set of states and transitions between them, where each state can also emit an output observation with a certain probability. Thus, two processes take place when generating sequences in an HMM. 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 [14].

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

2. V is the set of possible observations $V = \{v_0, v_1, ..., v_M\}$, 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:

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

3. A is the state transition probability matrix of dimension NxN. 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 NxM. We define $b_i(v_k)$ as the probability of observation v_k being produced at state i, which is independent (the probability) of time instant t.

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

5. π is the initial state probability array.

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

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

There are two important assumptions made by the model [14]. The first is called the *Markov assumption* and specifies that each state is dependent only on the previous state, instead of the history of all previous states. The second assumption asserts that the output observation probability depends only on the current state of the system, i.e. it is independent of previous observations.

2.2. DR-HMM: feature reduction method proposal

As a text-oriented feature reduction method, DR-HMM aims to transform the representation of an initial document set into a more useful representation. This usefulness can be measured in terms of computational cost and accuracy in an automatic classification process. In this case, the goal of the technique is to lower the dimensions of the resultant feature space and transform the representation into one more useful for classifying algorithms, such as k-NN or SVM. Specifically, the DR-HMM is designed to take input datasets represented in a bag-of-words approach and transform them into a customized similarity-based representation.

The DR-HMM model is built around a training corpus that is used as the document set on which the output similarity space is based. Figure 2 shows the main workflow applied to create the DR-HMM.

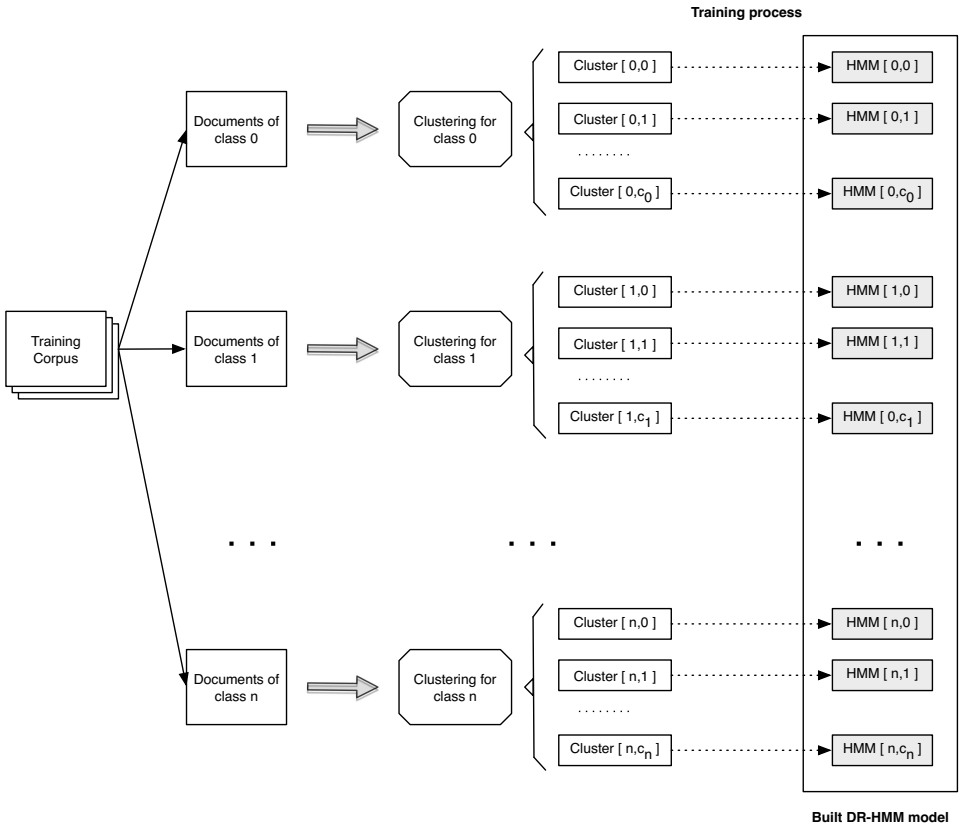


Figure 2: General workflow to build the DR-HMM model which uses an initial training corpus as input. Firstly, a clustering process is applied for each class. An HMM is then created and trained for each generated cluster in order to represent the documents within that cluster. The resultant set of HMMs confirms the DR-HMM method.

In the building process, two main phases can be distinguished:

1. Document clustering of the base corpus.
2. Creation and training of the models to represent each cluster.

Clustering is a useful technique that automatically organizes a collection into a number of groups. Text document clustering groups similar documents into a coherent cluster [15]. In order to avoid using all the single documents in the training corpus as the document base in the similarity space, document clusters are taken as the base set. This implies that the dimension of the new feature space has a magnitude dependant on the number of document clusters created with the training corpus.

Any clustering algorithm can be used. Of all the types of clustering methods, partitional clustering algorithms like K-means have been recognized to be better suited for handling document datasets, due to their relatively low computational requirements [16]. In our work, the X-means algorithm [17] is chosen as the clustering method. This algorithm is an extension of the standard K-means technique in which the number of clusters does not need to be previously determined.

The clustering algorithm is applied per each class in the training corpus, as seen in Fig. 2. Documents are split into groups according to their class value. The clustering process is then performed for each group, producing a resultant cluster set R_i for each class i, with the following characteristics:

- R_i is a document cluster set derived from Z_i, where Z_i is the document set of the class i in the original training corpus.
- $|R_i| = c_i$, where c_i is the number of clusters generated with the clustering algorithm applied over Z_i. c_i ranges from 1 to $|Z_i|$.

Since the document clusters are the base of the new feature space, an additional way of computing distances between new documents and these clusters needs to be defined. This is where the HMM is integrated in the method. A generative model based in HMMs is created and trained with each cluster, as is shown in Fig. 2.

The next section shows how the HMM model is defined and trained in order to represent the documents within a cluster.

2.3. Using HMM to represent a document cluster

In a previous work, the authors have developed an HMM based document classifier called T-HMM [10, 11]. In this proposed model (DR-HMM), we use the same type of HMMs as a document generator. The HMM is trained with all the documents in the cluster. Whenever the distance between a new document and the cluster needs to be computed, the system evaluates the probability (density) of this document being generated by the Hidden Markov Model. This value is called the likelihood and is considered the similarity measure in the DR-HMM. The rest of this section summarizes the T-HMM definition developed in the previous work, since HMMs have the same structure in DR-HMM.

Hidden states in T-HMM 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. 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 relevance measure for words depends on the document representation. In this case, every document is represented with a bag-of-words approach where elements in the vector describe the word frequency, as it is shown in Fig. 3(a). After a pre-processing step such as described in Section

3.3, words with a higher number of occurrences are more relevant because they are considered the best representation of the document semantic. For training purposes, words are placed in descending order according to their ranking to represent each document (see Fig. 3(b)). It should be noted that in this example, as the HMM has only three states, the 4th rank of words is ignored when the training process takes place.

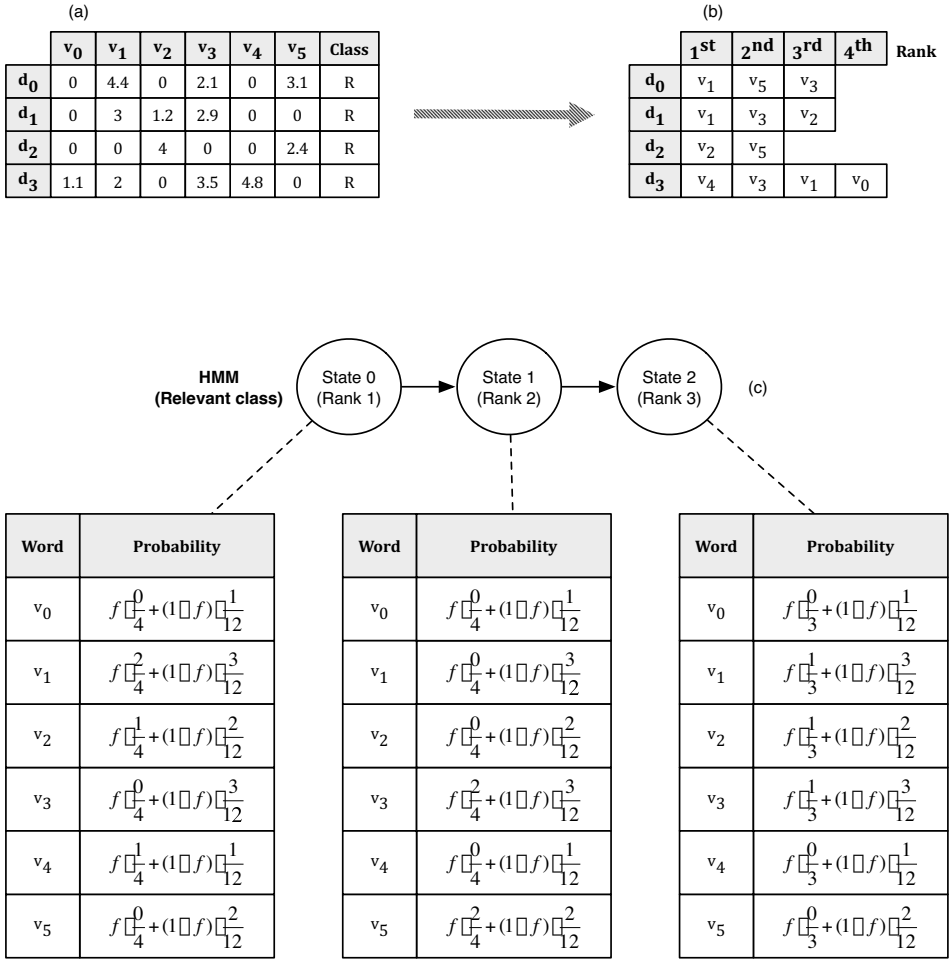


Figure 3: Example of the HMM training process for relevant documents. (a) Document vector matrix (b) Relevant document vectors (tagged as R) with their words ordered by relevance (c) Output matrices for states.

Considering that each document is represented by a vector or a wordlist ranked in decreasing order, and ignoring words with zero value, a Hidden Markov Model is used to represent a document cluster as follows:

1. The union of words from the document cluster is taken as the set of observation symbols V. For each word, there is a symbol v_k.

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 [9] without self-state loops, in which the probability of state S_{i+1} after state S_i is 1. The transition probability matrix A is then defined as:

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

3. The observation output probability distribution of each state is defined according to the document content in the cluster. 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, also have a probability of emitting words appearing in documents in the cluster that HMM was built for. The weight (importance) of these two separate probabilities is controlled by a f parameter.
Given the document set D_c of the cluster, each element of the output probability matrix B for an HMM that represents that cluster is defined as follows:

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

   (a) $b_i(v_k)$ stands for the probability of the word/observation v_k being emitted at state s_i
   (b) $f \in [0, 1]$
   (c) $R_d(v_k, i) = \begin{cases} 1 & \text{if word } v_k \text{ appears at } i\text{th rank position in document } d \\ 0 & \text{in other case} \end{cases}$
   (d) $E_d(i) = \begin{cases} 1 & \text{if there is any word with } i\text{th rank position in document } d \\ 0 & \text{in other case} \end{cases}$
   This factor is necessary because documents have a different number of feature words. If the number of states is too high, some documents may not have enough feature words to complete all position ranks.
   (e) $A_d(v_k) = \begin{cases} 1 & \text{if word } v_k \text{ appears at least one time in document } d \\ 0 & \text{in other case} \end{cases}$
   (f) $|V|$ is the number of feature words.

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

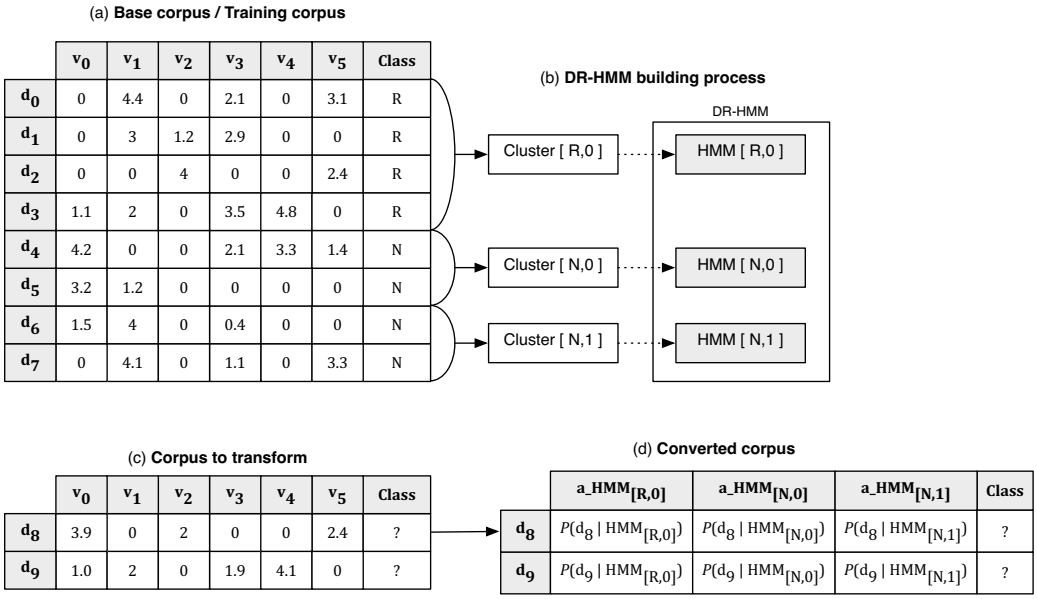
The number of states N and the generalization factor f are parameters that should be set depending on the corpus. The first part of formula (1) represents the relevance of the ranking order. The more weight this part has, the more restrictive the model is when classifying a new document, as it takes into account the exact order of words in relevance from the document set. Although it can increase the precision of the categorization process, this can lead to an overfitting if the f-value is too high.

The second part maintains the same value for all states and provides the model with a better generalization to classify documents. This is why the number of states and the f variable must be adapted to the corpus.

As an example, Fig. 3 shows the training process of an HMM with three states for a cluster consisting in four documents (Fig. 3(a)). Firstly, documents are formatted into a list of distinct words in descending order according to their ranking (Fig. 3(b)). The HMM is then created and a probability distribution matrix is assigned to each state following formula (1) (Fig. 3(c)).

2.4. Data transformation

The previous training process is applied for each cluster generated in the clustering step, since one HMM is created to represent each cluster. Once all the HMMs are trained, the DR-HMM model is considered to be finally built, and can be used to transform every single dataset into the final HMM-based similarity space.



	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

	v_0	v_1	v_2	v_3	v_4	v_5	Class
d_8	3.9	0	2	0	0	2.4	?
d_9	1.0	2	0	1.9	4.1	0	?

	$a\_HMM_{[R,0]}$	$a\_HMM_{[N,0]}$	$a\_HMM_{[N,1]}$	Class
d_8	$P(d_8 \mid HMM_{[R,0]})$	$P(d_8 \mid HMM_{[N,0]})$	$P(d_8 \mid HMM_{[N,1]})$	?
d_9	$P(d_9 \mid HMM_{[R,0]})$	$P(d_9 \mid HMM_{[N,0]})$	$P(d_9 \mid HMM_{[N,1]})$	?

Figure 4: Example of feature transformation using a DR-HMM model. (a) Training corpus to build the DR-HMM. (b) DR-HMM building process in which three final clusters are created. (c) Dataset to be transformed (d) Resultant transformed dataset. It has one synthetical attribute per cluster.

Figure 4 shows an example of the DR-HMM model building. In this case, the training corpus (Figure 4(a)) has two possible classes for the documents, Relevant or Non-relevant. The clustering process produces a total of three clusters: two for the non-relevant class and one for the relevant class. The clusters are taken as the training sets for each HMM created (Figure 4(b)). When a new document set (with the same feature space representation) arrives (Figure 4(c)), the DR-HMM consisting of the three HMMs is used to transform it into the new similarity space (Figure 4(d)).

The resultant dataset has a synthetical attribute $a\_HMM_{[x,y]}$ per trained HMM. In order to transform a document d, the similarities between this document and each cluster (represented by the HMMs) need to be computed:

- Initially, the document d must be formatted into an ordered wordlist L_d in the same way as in the training process for the HMMs.

- As words are considered observations in the HMMs, we calculate the probability (likelihood) of the word sequence L_d being produced by each HMM; that is, $P(L_d|\lambda_h)$ for each trained HMM λ_h in the DR-HMM model.

The likelihood measures are calculated by applying the forward-backward algorithm as proposed by Rabiner [9]. In this case, since there are no loops in the HMM model and the state transition is fixed, the likelihood $P(L_d|\lambda_h)$ for the HMM λ_h with n states is calculated as:

$$P(L_d|\lambda_h) = \prod_{i=1}^{\min(|L_d)|-1,n-1)} b_i(L_{d_i}) \tag{2}$$

Finally, the calculated likehood measures are taken as the similarity values between the document and the clusters. This can be seen in the elements of the vectors of Fig. 4(d).

3. Results and Discussion

The goal of the experiments is to test the performance of the proposed feature reduction technique in two document corpora, OHSUMED and TREC Genomics. The collections are described in the following sections, and are used in a classification process with the SVM and k-NN classifiers. In order to compare the effectiveness of the proposed feature reduction technique, the two corpora are classified without using any feature reduction algorithm. The same classifying process is applied with resultant collections from the DR-HMM system and the *Information Gain* feature selection process, commonly used in text-oriented classifications.

3.1. TREC Genomics Corpus

One of the tasks in TREC Genomics 2005 Track [18] 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). A different corpus is created for each category, where documents can be classified as relevant or non-relevant. In this paper, the collection of abstracts of the Allele corpus is used to test the performance of the proposed system.

3.2. OHSUMED Corpus

The OHSUMED test collection, initially compiled by Hersh *et al.* [19], is a subset of the MEDLINE database, which is a bibliographic database of important medical literature maintained by the National Library of Medicine. OHSUMED 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 with an average of 150 words 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 assigned two or more categories and one of them is considered relevant, then the document itself will also be considered relevant and will be 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, ending up with five distinct matrices.

3.3. Experiments description

Figure 5 shows the entire classification process with the application of a feature selection method. This workflow is used in our experiments to test the performance of the proposed system.

Initially, the document corpora need to be pre-processed. Following the bag-of-words approach, we format every document into a vector of feature words in which elements describe the word occurrence frequencies. 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 [20] is applied. Then, words occurring in fewer than ten documents of the entire training corpus are also removed.

Once the initial feature set is determined, a dataset matrix is created where rows correspond to documents and columns to feature words. The value 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 [3].

In the final step of the pre-processing phase, the initial dataset matrix is divided into two different matrices: the train matrix (50% of the documents) and the test matrix (50% of the documents). This division is performed randomly while maintaining the class proportion in the original corpus.

Once the pre-processing phase is finished, the feature reduction process is applied to further decrease the dimensionality of the corpus. Most feature reduction techniques, such as the proposed DR-HMM, are built around a training corpus. The train matrix is used as the base corpus in the following algorithms:

- The feature selection method based in Information Gain that is implemented in WEKA [21] is used as the base feature reduction algorithm, since it was previously employed in similar text classification tasks [4, 22]. 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 information gain is included in the resultant feature set.

- A DR-HMM system is built with the resultant corpus from the application of the Information Gain method. The feature reduction phase can be applied repeatedly so that different feature reduction techniques can be applied together. In this case, building the DR-HMM model with the feature word space resultant from the Information Gain algorithm leads to better results in accuracy and execution time. The clustering algorithm applied is the previously mentioned X-means technique, where the number of clusters does not need to be specified. Finally, the parameters of the HMMs that represent the generated clusters are:

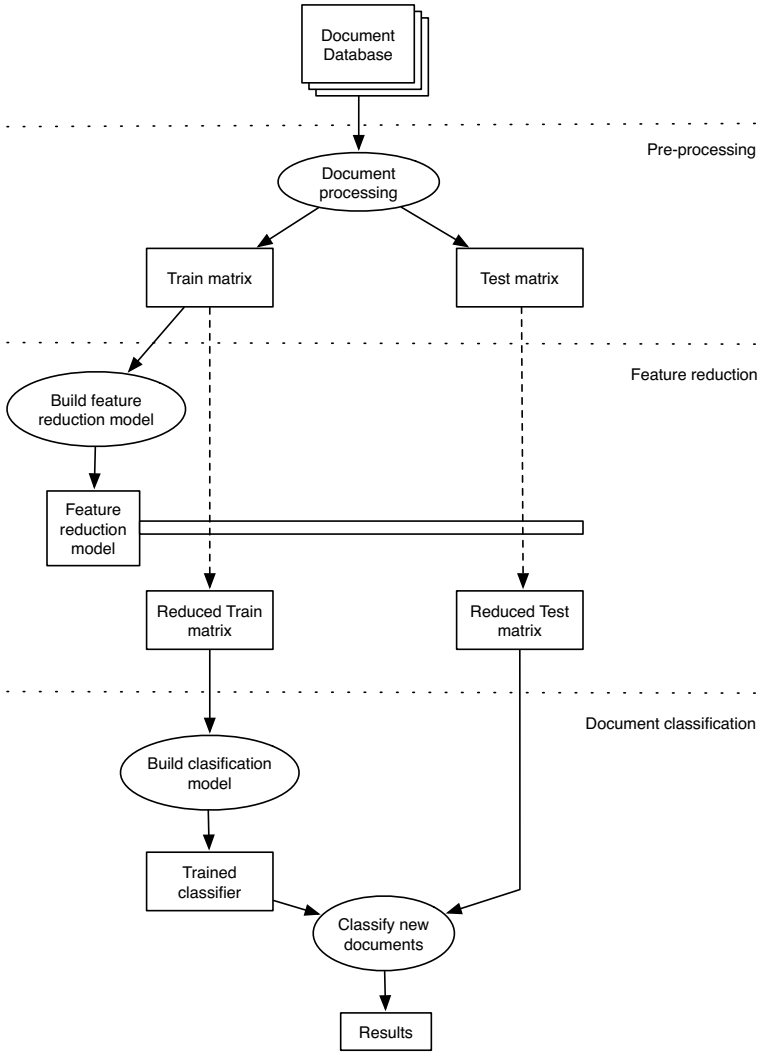


Figure 5: Classification process with the usage of a feature selection method

- The number of states N is set to the average number of feature words with a non-zero frequency value in the cluster documents.
- The f-factor is set to $f = 0.20$, since it provided the best results in the experiments (see Figure 6). For more information on the parametrization of the HMM, refer to [10].

The reduced train and test matrices obtained from the application of both methods are used in the next classification phase. In this case, the following classification algorithms are trained to perform the document classification task:

- k-NN: The number of neighbours is set to $k = 3$ since it leads to the best performance of the

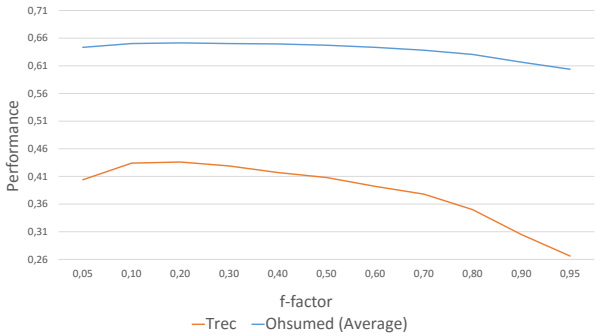


Figure 6: Results for the evaluation of the f-factor in TREC and OHSUMED Corpus. The results corresponds with the average Kappa Statistic achieved in 5 classification processes with a SVM for each tested f-factor value. For the OHSUMED corpus, the values achieved in each dataset (one for each category) are averaged and plotted in a single value.

algorithm in the tested corpus. In addition, the Euclidean distance is used to measure the distances between documents in the classifier.

- Support Vector Machine (SVM): The implementation of the SVM used in this case is LIBSVM [23] and the parameters are those utilized by default in the WEKA environment [24], applying a RBF kernel.

In order to evaluate the effectiveness of the DR-HMM system and compare it with the InfoGain based algorithm, the complete process consisting of the three phases of Fig. 5 is executed for each method. Specifically, for statistical comparison purposes, and since the corpus is randomly split in the first phase, the entire process is executed ten times per corpus, classifier and feature reduction method.

3.4. Experimental results

To evaluate the effectiveness of the models, F-measure and Kappa Statistic, evaluation measures commonly utilized in text classification and information retrieval, are used. F-measure is the weighted harmonic mean of recall and precision, and Kappa is a single metric that takes the output confusion matrix of an evaluation and reduces it into one value [25]. Kappa Statistic measures the agreement of prediction with the true class and it compares the accuracy of the system to the accuracy of a random system. Possible values of kappa range between 0 (random classification / no agreement) and 1 (perfect classification / complete agreement). Specifically, Kappa statistic is calculated as follows:

$$\text{Kappa} = 1 - \frac{(1-\text{observed accuracy})}{(1-\text{expected accuracy})}$$

Table 1 shows the results achieved. The values correspond to the average value achieved in that measure for the total of 10 executions with each method. In addition, the F-measure value is associated with the relevant class of the documents. This measure is selected since the relevant class is the minority class in every corpus, as shown in Table 2. Using this measure helps verify the differences between the models since this value is more sensitive to changes than its non-relevant counterpart.

Evidently, the new similarity representation offered by the DR-HMM leads to a large improvement of the classification performance with the k-NN algorithm. In the case of SVM, the values achieved with InfoGain and DR-HMM are much closer, and the DR-HMM system outperforms the InfoGain model in four of the six tested corpora. On the other hand, the application of InfoGain leads to better results in two of the six corpora with the SVM classifier.

In addition, the average final number of features for each technique can be seen in Table 2. The DR-HMM system clearly produces a smaller number of output features due to the application of the clustering method. This greatly helps classifiers like k-NN in which the classification process depends on the quality and quantity of the features to measure the distances between documents. It is important to note that the number of raw features can quickly increase as the number of documents and their content grows. Larger datasets can become non-tractable for classifiers like SVM, however, both InfoGain and DR-HMM can be combined to reduce their dimensionality. Based on the results, the usage of both techniques can alliviate the need for information extraction in some cases.

		k-NN			SVM		
Corpus	Measure	Raw Data	InfoGain	DR-HMM	Raw Data	InfoGain	DR-HMM
TREC	F-measure	0.000	0.111	**0.541**	0.108	0.148	**0.434**
Allele	Kappa	0.000	0.102	**0.516**	0.102	0.141	**0.415**
Ohsumed	F-measure	0.426	0.479	**0.789**	0.755	0.788	**0.807**
C04	Kappa	0.215	0.378	**0.715**	0.690	0.726	**0.745**
Ohsumed	F-measure	0.096	0.475	**0.624**	0.650	0.761	**0.776**
C06	Kappa	0.068	0.437	**0.551**	0.617	0.729	**0.745**
Ohsumed	F-measure	0.243	0.422	**0.800**	0.785	**0.824**	0.817
C14	Kappa	0.143	0.340	**0.735**	0.730	**0.771**	0.763
Ohsumed	F-measure	0.115	0.422	**0.620**	0.561	0.579	**0.696**
C20	Kappa	0.092	0.380	**0.548**	0.523	0.539	**0.656**
Ohsumed	F-measure	0.470	0.424	**0.554**	0.493	**0.580**	0.557
C23	Kappa	0.058	0.200	**0.274**	0.346	**0.393**	0.348

Table 1: The best values among classifiers for a specific measure and corpus are shown in bold

	Number of documents		Number of features		
Corpus	Relevant	Non-Relevant	Raw Data	InfoGain	DR-HMM
TREC Allele	591	10204	9590	592	12
Ohsumed C04	2630	7755	6597	1134	11
Ohsumed C06	1220	8430	6608	496	97
Ohsumed C14	2550	8030	6751	1206	26
Ohsumed C20	1220	8239	6487	575	49
Ohsumed C23	3952	6778	6302	597	71

Table 2: Description of the datasets. The number of documents corresponds with the total number of relevant and

non-relevant documents in each entire corpus before creating the train and test splits. The *number of features* column shows the resultant number of features for each corpus after applying the dimensionality reduction methods.

3.5. Statistical test

In order to demonstrate that the observed results are not just a chance effect in the estimation process, we must 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 the collection of Kappa measures achieved by InfoGain and DR-HMM methods in both k-NN and SVM classifiers in order to prove their differences. The results previously noted show that DR-HMM has generally a higher mean of Kappa values than InfoGain.

Table 3 shows the results for the executed t-tests. One test is performed for the collection results achieved in each corpus. The difference for a given confidence level is checked to determine if it exceeds the confidence limit. In that case, the *null-hypothesis* (the difference is due to chance) is rejected, proving that the model with a higher mean value is statistically better than the other one.

According to these results, DR-HMM clearly outperforms the InfoGain model in all the cases using a k-NN classifier. On the other hand, using a SVM classifier, the DR-HMM system is proven to be statistically better than InfoGain in four of the six tested corpus. In the two other cases, the application of the DR-HMM method does not improve the performance of the SVM classifier.

	k-NN with DR-HMM		k-NN with InfoGain		
Corpus	Av. Kappa	Sd Kappa	Avg Kappa	Sd Kappa	t-value
TREC Allele	0.516	0,026	0.102	0,026	32.462 •
Ohsumed C04	0.715	0,016	0.378	0,025	33.687 •
Ohsumed C06	0.551	0,044	0.437	0,033	5.913 •
Ohsumed C14	0.735	0,011	0.340	0,022	67.092 •
Ohsumed C20	0.548	0,037	0.380	0,017	12.513 •
Ohsumed C23	0.274	0,022	0.200	0,015	8.300 •
	SVM with DR-HMM		SVM with InfoGain		
Corpus	Avg Kappa	Sd Kappa	Avg Kappa	Sd Kappa	t-value
TREC Allele	0.415	0,028	0.141	0,031	29.890 •
Ohsumed C04	0.745	0,011	0.726	0,009	4.699 •
Ohsumed C06	0.745	0,013	0.729	0,013	2.352 •
Ohsumed C14	0.763	0,010	0.771	0,008	-2.980 ○
Ohsumed C20	0.656	0,012	0.539	0,017	19.622 •
Ohsumed C23	0.348	0,014	0.393	0,008	-9.571 ○

p-value (confidence level): 0.05

t-value confidence limits (two-tailed, 10 degrees of freedom): +/- 2.2282

null-hypothesis: There is no difference between models

• : InfoGain method is significantly worse than the DR-HMM method

○ : InfoGain method is significantly better than the DR-HMM method

Table 3: Results for corrected paired t-test between DR-HMM and InfoGain methods with both of the two used classifiers (k-NN and SVM). *Null-hypothesis* is rejected if t-value exceeds the confidence limits.

3.6. Execution time

The execution times for the experiments were also saved. This includes the time needed to build each feature reduction model, its application, and finally, the document classification.

Table 4 shows the results achieved using the most beneficial classifier from the application of the DR-HMM model: the k-NN algorithm. Even though the time spent building the DR-HMM system is greater than that needed for the InfoGain method, the classification time is highly decreased. This fact implies that multiple executions of a classification process (which is a common situation in machine learning domains) have a better performance in time using the DR-HMM method, as shown in Figure 7. The "raw" column refers to the execution of the classification process without using any feature reduction technique. As it can be seen, the classification step with all the initial features consumes much more time and becomes worse after multiple executions (see Figure 7).

The time needed to build the DR-HMM model is higher due to the clustering process. Larger datasets can lead to computational problems, however, a simpler clustering algorithm like K-means with a fixed number of clusters can be used to accelerate the building process.

CPU Time	Raw	InfoGain	DR-HMM
Build Model	-	50.920 s	101.644 s
Filter Test Corpus	-	0.152 s	14.174 s
Classify Test Corpus (k-NN)	70,804 s	30.310 s	2.467 s

Table 4: Results for user CPU time in seconds, representing the average execution time for building each model with the train corpus, filtering the test instances and classifying them with a k-NN classifier. The Allele corpus is taken as the base corpus.

4. Conclusions

This paper presents a novel technique for reducing the input space in order to improve the efficiency of the text classifiers. The technique utilizes a document clustering to separate data into groups, and introduces a similarity-based document representation based on a Text Hidden Markov Model. The approach is convenient for large datasets.

The proposed technique has a general purpose within the field of text classification. Experiments are performed using a biomedical corpus in order to test the applicability of the model to this scope. The experimental results show that the document classification accuracy obtained after the dimensionality reduction using the proposed technique is, in general, better than the original accuracy if a statistical filtering method such as InfoGain is applied.

Although the time spent building the system is greater, when comparing the performance of this proposal and other approximations, the classification time is highly decreased with multiple executions.

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 by the [12VI08] Contract-Programme from the University of Vigo.

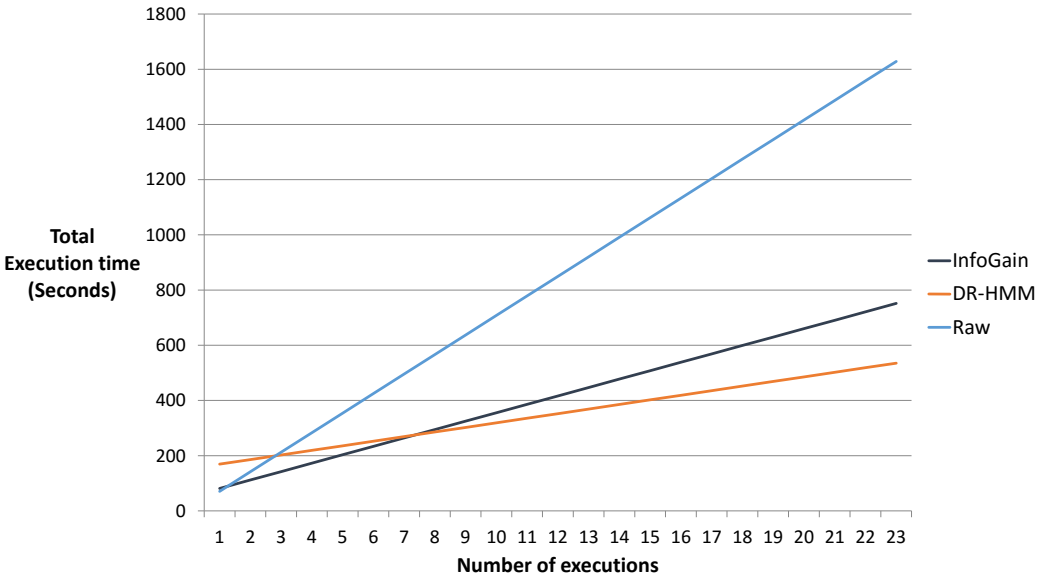


Figure 7: Total execution time for repeated tests with k-NN in the reduced corpus obtained with each method (InfoGain and DR-HMM) amd without using any feature reduction technique (Raw). The Allele corpus is used for this experiment.

[1] F. Sebastiani. Text categorization. In *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*, pages 109–129. WIT Press, 2005.

[2] N. Tsimboukakis and George Tambouratzis. Document classification system based on HMM word map. In Richard Chbeir; Youakim Badr ;Ajith Abraham ; Dominique Laurent ; Mario Koppen ;Fernando Ferri ;Lotfi A. Zadeh ; Yukio Ohsawa, editor, *In Proceedings of the CSTST-2008 Conference*, pages 7–12, Cergy-Pontoise, Paris, France, 27-31 October 2008.

[3] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman, 1999.

[4] A. G. Janecek, W. N. Gansterer, M. A. Demel, and G. F. Ecker. On the relationship between feature selection and classification accuracy. In *JMLR: Workshop and Conference Proceedings 4*, pages 90–105, 2008.

[5] Harun Uuz. A hybrid system based on information gain and principal component analysis for the classification of transcranial doppler signals. *Computer Methods and Programs in Biomedicine*, 107(3):598 – 609, 2012.

[6] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[7] E. Pekalska and R. P. W. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23:943–956, 2002.

[8] M. Bicego, V. Murino, and M. A. T. Figueiredo. Similarity-based classification of sequences using hidden markov models. *Pattern Recognition*, 37(12):2281–2291, 2004.

[9] L. R. Rabiner. Readings in speech recognition. chapter A tutorial on hidden Markov models and selected applications in speech recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[10] A.Seara Vieira, E.L. Iglesias, and L. Borrajo. T-hmm: A novel biomedical text classifier based on hidden markov models. In Julio Saez-Rodriguez, Miguel P. Rocha, Florentino Fdez-Riverola, and Juan F. De Paz Santana, editors, *8th International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB 2014)*, volume 294 of *Advances in Intelligent Systems and Computing*, pages 225–234. Springer International Publishing, 2014.

[11] Seara Vieira A. Iglesias E.L. Borrajo, L. Tcbr-hmm: An hmm-based text classifier with a cbr system. *Applied Soft Computing*, 26:463–473, 2015.

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

[13] Harun Uğuz, Gür Emre Güraksn, Uçman Ergün, and Rdvan Saraçoğlu. Biomedical system based on the discrete hidden markov model using the rocchio–genetic approach for the classification of internal carotid artery doppler signals. *Computer Methods and Programs in Biomedicine*, 103(1):51–60, 2015/06/29.

[14] K. Li, G. Chen, and J. Cheng. Research on hidden markov model-based text categorization process. *International Journal of Digital Content Technology and its Application*, 5(6):244–251, 2011.

[15] A. Huang. Similarity measures for text document clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[16] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD Workshop on Text Mining*, 2000.

[17] D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[18] W. Hersh, A. Cohen, J. Yang, R. T. Bhupatiraju, P. Roberts, and M. Hearst. Trec 2005 genomics track overview. In *TREC 2005 notebook*, pages 14–25, 2005.

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

[20] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[21] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Sys. Morgan Kaufmann, June 2005.

[22] Gregory J. Caporaso, William A. Baumgartner, Bretonnel K. Cohen, Helen L. Johnson, Jesse Paquette, and Lawrence Hunter. Concept recognition and the TREC Genomics tasks. In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*, 2005.

[23] C. Chang and C. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

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

[25] A. J. Viera and J. M. Garrett. Understanding interobserver agreement: the kappa statistic. *Family Medicine*, 37(5):360–363, 2005.