2009 Ninth International Conference on Intelligent Systems Design and Applications

Agglomeration and Elimination of Terms for Dimensionality Reduction

Patrick Marques Ciarelli
Department of Electrical Engineering
Universidade Federal do Espírito Santo, UFES
Vitória, Brazil
pciarelli@lcad.inf.ufes.br

Elias Oliveira
Department of Information Science
Universidade Federal do Espírito Santo, UFES
Vitória, Brazil
elias@lcad.inf.ufes.br

***Abstract*—The vector space model is the usual representation of texts database for computational treatment. However, in such representation synonyms and/or related terms are treated as independent. Furthermore, there are some terms that do not add any information at all to the set of text documents, on the contrary they even might harm the performance of the information retrieval techniques. In an attempt to reduce this problem, some techniques have been proposed in the literature. In this work we present a method to tackle this problem. In order to validate our approach, we carried out a serie of experiments on four databases and we compare the achieved results with other well known techniques. The evaluation results is such that our method obtained in all cases a better or equal performance compared to the other literature techniques.**

***Keywords*-dimensionality reduction; feature selection; agglomeration of terms; text classification**

I. Introduction

Text categorization is an activity that is rapidly growing in importance nowadays, due to the huge amount of information available and the great challenge of retrieving relevant information. These difficulties are tackled by the information retrieval (IR) communities, both in academic and industrial contexts. In order to be able to face this challenge of dealing with a huge amount of information at once it is necessary to consider the computer systems as a useful tool. However, in order to process a great quantities of available texts in a computer systems, it is necessary to model them before accordingly. A quite common model used is the vector space [1]. In this model each text is represented by a vector, where each dimension of this vector means a possibily weight of a term [1]. This weight can be the term's frequency, absence or presence of term (binary weight) or another form of weighting. In this way, the database can be represented by a $M \times N$ matrix, where M is the number of distinct terms and N the number of texts in the database. Such matrix is also called term-document matrix.

Nevertheless, this approach presents some problems. First, the word ordering information is lost. Second, the number of distinct terms in database is normally huge (many times of the magnitude of thousands of terms), as a consequence the computer cost is high. Third, each text is represented by just a small part of this terms, therefore, the matrices are much sparse. Fourth, synonimous and strong correlated terms are treated as independent terms, and this may lead to loss of information. Fifth, terms well known as stopwords, which have low semantic value [3] (such as articles, prepositions, etc), and terms with great frequency among the texts have low power of categories' discrimination.

Some methods to avoid, or minimize, these problems have been proposed in the literature. In order to reduce the number of terms with low semantic value is common to perform the procedure of removing out the stopwords. Other techniques use a procedure of weighting of the terms to select the terms with higher power of discrimination and eliminate the others, and thus to reduce the dimension of the matrix [3]. Moreover, there are techniques that perform the clustering of the terms to reduce the sparsity and the size of the matrices [2].

The last two set of techniques have obtained good results in the literature [4], [5], [6], [7]. Due to these results we expect the combination of these techniques may return better results than when they are separately applied. Thus, in this article we propose a novel approach that combines the caracteristics of these techniques to eliminate and merge terms. To show the performance of our proposed technique we carried out a series of experiments and compare them against others well known techniques in the literature.

We conducted experiments on four databases and we applied the statistical t-test to evaluate the performance of the algorithms. As result, our approach was statistically superior to some widespread methods (see Section IV for more explanation) and it presented similar performance to Latent Semantic Indexing (LSI). Furthermore it achieved 11 out of the 20 best results in the experiments.

This work is organized in the following structure: in Section II we present a revision of the techniques used in this work. We describe our proposed technique in Section III. In Section IV is defined how we performed our experiments and the results achieved. Finally, we present our conclusions and indicate some future paths for this research in Section V.

[1] Such representation is called of *bag of words*. There are other type of representation that do not use terms, but other extracted features of the documents. For more details we recomended to see [2].

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.9

IEEE computer society

II. Literature Revision

We start this section with a brief dicussion on some of the literature's techniques for the reduction of the dimension of the text-documents matrices.

The task of feature selection, or term selection, is to select a good subset of terms from database which helps to discriminate between classes, and eliminate terms that add noise. On the other hand, agglomeration of terms helps to reduce the level of stochastic dependence between terms, besides may merge terms which are highly correlated. Both methods can be employed to reduce the dimension of the data matrices.

Two main approaches can be employed to perform feature selection and groups of terms: the filter approach and the wrapper approach [6], [8]. In the first one the terms are selected or merged independetly from the learning method that will use them. In the second one the terms are chosen or grouped based on some learning algorithm that will identify the best combination of terms and then it will use them. The second approach has the disadvantage that is computational more costly than the filter approach, because it needs to call the learning algorithm for each combination. Due to this, the approach used in this work will be the filter.

Now we will do a brief general description of the methods of feature selection and agglomeration of terms.

A usual procedure in documents classification, or clustering, is the use of a document (formed by many terms) previously represented by a vector space and, by the use of a classifier, we identify one or more documents similar to the one at hand.

In the clustering of terms a similar process is done, however with a slight difference. In this case, each vector represents a term and its element is associated to a document. By this way, the measure of *distance* can be accomplished among the interest's point and the points assigned to a group, or it can be achieved between a point and a centroid of a group. The algorithms of clustering presented in this article refer to the second method. In addition, it is possible to devise two types of cluster: a soft cluster and a hard cluster. In the first case, the terms may belong more than one group and, in the second case, the terms are associated just to a unique group. With the exception of the Latent Semantic Indexing (see Section II-C), that perform a task similar to a soft clusters, all the clustering algorithms in this paper will form hard clusters.

For the feature selection methods the basic idea is to mathematically measure the importance level of each term for categorization. Terms with good scores are keeped in the database and those with bad scores are removed. All the algorithms presented in this article are unsupervised, including that proposed by us here.

A. Mutual Information (MI)

Mutual information (MI) measures the probability of observing two variables x and y together with the probability of observing x and y independently. High value of MI between x and y indicates that there is a high association between these two variables. For low value means that there is not significant association between them [9]. In this work we use MI to extract the words that minimize the loss of information. Words with high value of MI have high relationship with the texts contained in the database. Words with low value may have low information to categorization of texts, so they can be removed. Equations 1 and 2 will be used in our experiments, so that x is the word ($x \in W$), y is the document ($y \in ST$), W is the set of distinct words and ST is the set of texts. The only difference between these two equations is the fact that in Equation 2 it is not considered the probability of word x.

$$MI_x = p(x) \sum_{y \in ST} p(y \mid x) \log \left(\frac{p(x \mid y)}{p(y)} \right), \tag{1}$$

$$MI_x = \sum_{y \in ST} p(y \mid x) \log \left(\frac{p(x \mid y)}{p(y)} \right). \tag{2}$$

B. Inverse Document Frequency (IDF)

IDF is a classical term-weighting scheme that gives a high weight to words that appear in few documents and low weight to words that appear in many documents [3], [10]. Note that in the computation of IDF it is not considered the term's frequency. It is taken into account whether a term occurred or not. In our experiments, the words with small and the large values of IDF were removed. In so doing that, the rare and the very common terms in the database are eliminated, and only the terms with intermediary frequency are kept.

The IDF_x is then calculated as in Equation 3:

$$IDF_x = \log \left(\frac{N}{n_x} \right), \tag{3}$$

where x is the word, N is the number of documents in the database and n_x is the number of documents that the word x occurs.

C. Latent Semantic Indexing (LSI)

The core of this method is the mathematical procedure of singular value decomposition (SVD), which application decomposes a term-document matrix into a set of orthogonal factors from which the original matrix can be approximated by linear combination. A new term-document matrix can be formed from weighted combinations of terms [11].

Let P and Q be the training and test term-document matrix, respectively. Then applying SVD on P, we obtain:

$$P = TSD^T, \tag{4}$$

where S is a matrix whose the elements of its main diagonal are the singular values and the others elements are equal to zero. Let S_k be formed by k largest singular values and T_k and D_k their corresponding singular vectors from T and D, respectively, we can obtain a *k-reduced singular value decomposition (rank-k SVD)* from Equation 5.

$$P_k = T_k S_k D_k^T. \tag{5}$$

However, we cannot get any reduction of the dimensionality using this procedure. Therefore, instead of using the P_k, we will use another formulation. Knowing matrix D holds the coordinates of individual document vectors, we can obtain the new training and test matrices (with the dimensionality reduction) from Equations 6 and 7 [12], [13]:

$$\hat{\mathrm{P}} = D_k, \tag{6}$$

$$\hat{\mathrm{Q}} = S_k^{-1} T_k^T Q. \tag{7}$$

It is important to note that the matrices $\hat{\mathrm{P}}$ and $\hat{\mathrm{Q}}$ have smaller dimensions than the original ones, when k is smaller than the number of rows (terms).

D. Agglomerative Information Bottleneck (aIB)

In the aIB approach there are initially k clusters, where k is equal to number of words and each cluster contains exactly one word. At each step the algorithm merge a pair of clusters into a single new cluster in a way that locally minimizes the loss of mutual information. Thus, it is expected to perform *the best possible merge* [5]. This procedure is repeated until the number of wished clusters is met. This algorithm uses a greedy procedure to reduce the number of clusters.

The inputs of the algorithm are the probability matrix (term-document matrix), the number of wished clusters and the value of the parameter β. The β is a parameter of tradeoff between compression and precision and the *softness* of the classification. For $\beta \to \infty$, the clusters are induced to have hard partition, that is, each word just can belong to one cluster. This was the value chosen in our experiments.

E. Sequential Information Bottleneck (sIB)

In the sIB method the words are initially randomly particioned in the number of chosen clusters. At each step, one word is taken out of its current cluster and is represented as a new cluster *nc*. Using a greedy algorithm, we can merge this new cluster into a cluster t^{new}, such that $t^{new} = argmin_{t \in T} d_F(nc,t)$ to obtain a new partition T^{new}, where $d_F(.,.)$ is a measure of score. So, at each iteration it is obtained a new words' partition such that it is expected to increase the performance of the partition. These steps are performed until do not exist any more change. Since this method can fall into local optima, the previous procedure is repeated H times to obtain H different solutions. The solution that obtains the better perfomance is chosen to be the output of the algorithm [4].

Similar to aIB, the inputs of this algorithm are probability matrix (matrix of documents), the number of wished clusters, the value of the parameter β and the value of H. In our experiments, we use $\beta \to \infty$ and $H = 15$.

III. Iterative Agglomeration with Elimination (IAE)

In our proposed technique, as similarly one can find in aIB, we consider n groups at the beginning, where each group contains exactly one word. On the other hand, differently from the aIB, at each step of the algorithm it is perfomed a procedure of two parts. In the first part, the algorithm uses a method to compute the power discrimination of the groups and to determine when a group must be removed or not. In order to make this decision, it is used a threshold value. In the second part, the algorithm decides when to merge a number of groups which can vary from zero to a half of the total number of groups. In the latest case, the number of existent groups would fall down to the half of the initial number, depending on the pre-established threshold's value. Whether none group is merged at any step, the threshold's value is reduced. This procedure is repeated until the number of required groups k has been achieved. The algorithm pointing out this methodology is described in Table I.

According to what is shown in Table I, two measures are applied to determine the power of discrimination and relationship's level among groups. In this work, we selected the IDF to remove groups which happen with a certain level of frequency in the database. But, we apply the Equation 8, instead of Equation 3 shown in Section II, because in the first equation its value is within of range from 0 to 1.

$$IDF_x = \log\left(\frac{N}{n_x}\right) / \log(N) = 1 - \log_N(n_x). \tag{8}$$

The chosen measure to acquire the relationship's level among the groups was the Hamming distance. Such distance considers only the presence or absence of terms within the documents, it is thus not considered in this case the weight of the terms. If the same set of terms appears in two documents, then the distance between them is zero. But, we modified a little the original metric, as is shown in Equation 9:

$$Hamming(A,B) = 1 - \frac{1}{N}\sum_{i=1}^{N} xor(A_i, B_i), \tag{9}$$

where A and B are distinct terms, N is the number of documents in the database e the *xor* operator computes the logic operation exclusive–*or*. Thus, this metric measures the co-occorence of terms in the documents. The selected measurements have the advantages of that they are quick to compute and they do not need any normalization of the data [2], just one step of binarization. However, it is possible to

[2]Since the threshold of binarization considers the presence or absence of terms as one and zero, respectively.

Table I
PSEUDO CODE OF IAE.

```
Algorithm IAE
Input
        data: term-document matrix
        k: number of groups
        threshold_1: threshold to eliminate groups
        threshold_2: threshold to merge groups
        reduction: value of reduction of the threshold_2
Output
        output: relationship among terms - groups
  1  while k is not achieved do
  2        compute descrimination power of the groups
  3        if exist groups with descrimination power below of threshold_1 then
  4              these groups are eliminated
  5        compute the relationship among the terms
  6        if exist relationship among terms above of threshold_2 then
  7              merge these terms
  8        else
  9              threshold_2 := threshold_2 – reduction
```

apply any other set of measurements in such technique.

In the step of clustering of terms, when Equation 9 is used, the vectors are submitted at a logic operation of inclusive-*or*. Furthermore, it is not possible to merge more than two groups into only one group for step – for exemple, merge three groups in one. In the experiments we used the threshold to remove groups equal to 0.1, where groups with values below of this value are eliminated. The initial threshold to merge the groups and reduction's value were set to 0.95 and 0.05, respectively. Thus, every time that it is necessary to reduce the merge's threshold, it will be decreased 0.05 from the threshold's value. The low value of the first threshold is to avoid the elimination of many terms, but just a small set. Whereas the high value of the second threshold is to allow that the formation of the groups happens slowly, instead of suddenly. Both values must be in a range between 0 and 1.

IV. EXPERIMENTS

In ours experiments we used four distinct single-label of text database: WebKB, Reuters-52, Reuters-8 and CNAE-9. The first three databases were obtained from [14] and Cardoso-Cachopo has applied the following pre-processing in them: keep only letters (that is, numbers, punctuation and etc were removed). Turn all letters to lowercase. The title/subject of each document was added at the body of the document. Words with less than 3-characters length were removed. We also removed the stopwords of the documents [3]. Next step, the words were stemmed, that is, the words were reduced to their stem by applying Porter's Stemmer algorithm [15]. Finally, each document was represented as a vector, where each dimension of the vector is one stemmed word and its value is the frequency of the word in the document. Words which do not happen in the document have frequency zero. Besides theses procedures, we reduced the quantity of words to 3000 using mutual information (Equation 1) on the training set of each database (see Table II). This latter step was needed because of the aIB algorithm's limitation, that uses a great quantity of memory.

Table II
INFORMATION ABOUT THE DATABASES USED IN THE EXPERIMENTS.

	#C	#t	Training set	Test set
Reuters-52	52	3000	6532	2568
Reuters-8	8	3000	5485	2189
WebKb	4	3000	2803	1396
CNAE-9	9	856	900	180

For the CNAE-9 database the following steps were performed: initially, we also kept only letters and then we removed prepositions of the texts [3]. Next, the words were transformed to their canonical form. Finally, using a similar procedure applied to the other databases, each document was represented as a vector, where the weight of each word is its frequency in the document.

Information about each database are illustrate in Table II, where #C indicates the number of classes, #t the number of terms and "Training" and "Test" set indicate the number of documents used to training and test in the experiments, respectively. The division of the three first databases is equal to found in [14].

The chosen parameters' values for the aIB and sIB were those mentioned in the Section II, and they are the same applied in [4], [5]. For IAE's parameters we chosen the cited values in Section III. The other algorithms do not need to fix any parameter, just the number of terms/clusters. With exception of sIB, every algorithms used here are deterministic, therefore they were applied only once on each database. The sIB algorithim needed a long time to accomplish the experiments, so it was applied just once on the databases, but since it repeats H times its procedure, it is possible to obtain a good estimate of its performance.

All algorithms were applied to the set of training databases to reduce their dimension down to 50, 100, 150, 200 and 250 terms/clusters. In addition to the reduction, we applied to the vectors representing the documents the infinite norm normalization. The terms/clusters selected by each algorithm were used by Nearest Neighbor with cossine metric on the test databases to perform the classification of documents [16]. The choice of this classifier is motivated by its simplicity, because it does not need any fit of parameter and it is a deterministic classifier.

The performances achieved by techniques are shown in Table III and they are in percentage. Each row in the table corresponds to one technique and each column represents one dimension. The best obtained result for each dimension of each database is highlighted in boldface. The techniques MI_1 and MI_2 are two versions of mutual information, where the first employs Equation 1 and the second uses

[3]In previous unpublished works using similar databases we noted that this procedure had presented better results.

Equation 2.

Analyzing the performance of methods, we can see that the performance of the IDF and MI_2 were much inferior than the performance of other techniques. For example, for Reuters-52 database they almost ever obtained results below than 10%, ridiculous values if we compare with average result, that was around 80%. Some times such equations are used to reduce a dimension of the databases, because they are fast and do not require great amount of memory, however such approaches may be very poor, as it is illustrated here. But, the version MI_1 of mutual information has almost the same speed and consumption of memory than the MI_2, and it had a quite superior performance, becoming it in a approach more interesting.

Other observed details are the hit rate obtained by LSI and the proposed technique IAE, that achieved in many cases the best result for different databases and dimensions.

Table III
RESULTS OBTAINED BY TECHNIQUES ON THE DATABASES.

Technique	Dimensions				
	50	100	150	200	250
Reuters-52					
aIB	0.7998	0.8279	0.8396	0.8579	0.8489
sIB	0.7457	0.8361	0.8474	0.8645	0.8660
MI_1	0.7601	0.8033	0.8364	0.8497	0.8586
MI_2	0.0117	0.0218	0.0323	0.0460	0.0549
LSI	**0.8501**	**0.8645**	**0.8637**	0.8594	0.8567
IDF	0.0358	0.0553	0.0849	0.1336	0.1589
IAE	0.7948	0.8275	0.8633	**0.8738**	**0.8750**
Reuters-8					
aIB	0.8977	0.9077	0.9150	0.9278	0.9292
sIB	0.9032	0.9100	0.9105	0.9169	0.9173
MI_1	0.8789	0.9045	0.9173	0.9274	0.9315
MI_2	0.4966	0.4984	0.4998	0.5002	0.5039
LSI	**0.9260**	0.9274	0.9210	0.9114	0.8990
IDF	0.5162	0.5368	0.5413	0.5482	0.5582
IAE	0.8611	**0.9328**	**0.9283**	**0.9392**	**0.9360**
WebKb					
aIB	0.6461	0.6712	0.6605	0.6504	0.6669
sIB	0.6390	0.6748	0.6977	0.6948	0.6848
MI_1	0.6705	0.6662	0.6762	0.6891	0.7006
MI_2	0.3926	0.4011	0.4062	0.4176	0.4269
LSI	0.6905	0.6755	0.6812	0.6748	0.6726
IDF	0.4083	0.4327	0.4312	0.4513	0.4799
IAE	**0.7013**	**0.7256**	**0.7256**	**0.7135**	**0.7178**
CNAE-9					
aIB	0.7722	0.8500	0.8556	0.8889	0.9111
sIB	0.8611	0.8556	0.8722	0.9222	0.9111
MI_1	0.8444	0.8722	0.9000	**0.9278**	**0.9278**
MI_2	0.1278	0.1444	0.1667	0.1889	0.2000
LSI	**0.8778**	**0.9278**	**0.9222**	0.9167	0.9056
IDF	0.1667	0.1833	0.2056	0.2389	0.2778
IAE	0.8389	0.8722	0.8833	0.9056	0.9111

In order to accomplish an cleared evalution of the methods, we applied two-tailed paired t-test at 5% significance level. In order to perform this task, each method was represented by the 20 obtained results and we applied the t-test to evaluate the performance between each pair of methods.

For better comprehension a mark $>$ to indicate which algorithm is statistically superior. In that way, if the technique $A1$ has a better performance than $A2$, so we have $A1 > A2$. The results of this test are shown in Table IV. For example, in the first row is shown aIB performance: it was superior to MI_2 and IDF, and it was inferior to LSI and IAE.

Table IV
RELATIVE PERFORMANCE AMONG THE APPROACHES.

Technique	Statistical Performance
aIB	{LSI, IAE} $>$ **aIB** $>$ {MI_2, IDF}
sIB	{LSI, IAE} $>$ **sIB** $>$ {MI_2, IDF}
MI_1	IAE $>$ **MI_1** $>$ {MI_2, IDF}
MI_2	{aIB, sIB, MI_1, LSI, IDF, IAE} $>$ **MI_2**
LSI	**LSI** $>$ {aIB, sIB, MI_2, IDF}
IDF	{aIB, sIB, MI_1, LSI, IAE} $>$ **IDF** $>$ MI_2
IAE	**IAE** $>$ {aIB, sIB, MI_1, MI_2, IDF}

According Table IV both LSI and IAE were the best methods, whose approaches were not inferior to none. Furthermore the IAE was superior to 5 out of the 6 methods and it obtained 11 out of the 20 best results, whereas the LSI was superior to 4 methods and was the best in 7 out to the 20. In third appears MI_1, which was inferior to IAE. aIB and sIB come after with similar performance. In the last positions we have IDF and MI_2, where both were statistically weaker than the other ones.

Observing the outputs of IAE we can verify that happened a larger elimination of terms to 50 dimensions, around 500 terms, and the performance of the system was slightly altered. Nevertheless to 100 or more dimensions there were few terms removed, around 5, and for CNAE-9 database any term was removed.

Other comparisons among the algorithms are the time and consumption of memory. The IDF, MI_1 and MI_2 are the fastest and they needed little memory. The LSI was also quick, but it requires a reasonable amount of memory, both to storage and to generate the matrices. Such algorithms needed about a hour to perform the task. The IAE and aIB needed more time (around a day) to compute the outputs than the previous cited ones, in addition the aIB requests a great amount of memory, this turns prohibitive apply it to huge databases. The sIB and IAE demand considerable quantity of memory, nevertheless the sIB is very slow and it needed some days (around a week) to obtain the results of the experiments. The reason to this long wait is the fact that the sIB computes just one dimension at each run, whereas the other algorithms need only one step of run. All algorithms were implemented in the Matlab software, version 7, and they were executed on a PC with Athlon 64 Dual Core Processor 3800 with 1 GHz of RAM.

V. CONCLUSIONS

Although the vector space model has been shown to be a good representation for documents in Information Retrieval, it has some undesirable effects, specially in the bag of words representation. Some of them are the presence of terms with low semantic information, the fact that synonyms and/or

related terms are treated as independent, and the usual huge sparse matrix of documents' representation to be dealt with.

In order to minimize these drawbacks, we presented a technique that both eliminates and merges terms for dimensionality reduction of text database. To show the performance of our technique we carried out a series of experiments on four databases, and we compared their results against others well known techniques in the literature. Our approach was superior to 5 out of the 6 methods, and it has similar performance to LSI. Furthermore it achieved 11 out of the 20 best results in the experiments.

To the best knowledge of the authors, this approach was one of the first attempts to merge techniques of eliminations and clustering terms in an iterative way. Actually, there are many works in the literature which have applied such techniques together, however they are normally used in sequential steps, where each method is applied onde after another only once. The results which we have obtained from this work are promising. In future work we are planning on studing new methods to enhance even more the performance of our approach in order to get even better results.

ACKNOWLEDGEMENTS

This work is partially supported by the Internal Revenue Brazilian Service (Receita Federal do Brasil) and Fundação Espírito Santense de Tecnologia – FAPES-Brasil (grant 41936450/2008).

REFERENCES

[1] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," Ithaca, NY, USA, Tech. Rep., 1974.

[2] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. vol. 34, no. 1, pp. 1 – 47, 2002.

[3] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, first edition ed. New York: Addison-Wesley, 1998.

[4] N. Slonim, N. Friedman, and N. Tishby, "Unsupervised Document Classification using Sequential Information Maximization," *Proceeding of SIGIR'02, 25th ACM international Conference on Research and Development of Information Retireval*, pp. pp. 129–136, 2002.

[5] N. Slonim and N. Tishby, "Agglomerative Information Bottleneck," *Neural Information Processing Systems (NIPS)*, pp. pp. 617–623, 1999.

[6] G. Uchyigit and K. Clark, "A New Feature Selection Method for Text Classification," *International Journal of Pattern Recognition*, vol. 21, no. 2, pp. 423 – 438, March 2007.

[7] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 412–420.

[8] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem." Morgan Kaufmann, 1994, pp. 121–129.

[9] K. W. Church and P. Hanks, "Word Association Norms, Mutual Information, and Lexicography," *Computational Linguistics*, vol. 16, no. 1, pp. 22 – 29, 1990.

[10] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Ithaca, NY, USA, Tech. Rep., 1987.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, pp. 391 – 407, 1990.

[12] E. Garcia, "Latent Semantic Indexing (LSI) A Fast Track Tutorial," 2006, http://www.miislita.com/information-retrieval-tutorial/latent-semantic-indexing-fast-track-tutorial.pdf.

[13] P. Moravec, M. Kolovrat, and V. Snášel, "LSI vs. Wordnet Ontology in Dimension Reduction for Information Retrieval," p. 9, 2004.

[14] A. Cardoso-Cachopo, "Datasets for single-label text categorization. http://web.ist.utl.pt/ acardoso/datasets/," 2007.

[15] M. Porter, "The Porter Stemming Algorithm. http://tartarus.org/ martin/porterstemmer/," 2006.

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, second edition ed. New York: Wiley-Interscience, 2001.