Beyond TFIDF Weighting for Text Categorization in the Vector Space Model

Pascal Soucy
Coveo
Quebec, Canada
psoucy@coveo.com

Guy W. Mineau
Université Laval
Québec, Canada
guy.mineau@ift.ulaval.ca

Abstract

KNN and SVM are two machine learning approaches to Text Categorization (TC) based on the Vector Space Model. In this model, borrowed from Information Retrieval, documents are represented as a vector where each component is associated with a particular word from the vocabulary. Traditionally, each component value is assigned using the information retrieval TFIDF measure. While this weighting method seems very appropriate for IR, it is not clear that it is the best choice for TC problems. Actually, this weighting method does not leverage the information implicitly contained in the categorization task to represent documents. In this paper, we introduce a new weighting method based on statistical estimation of the importance of a word for a specific categorization problem. This method also has the benefit to make feature selection implicit, since useless features for the categorization problem considered get a very small weight. Extensive experiments reported in the paper shows that this new weighting method improves significantly the classification accuracy as measured on many categorization tasks.

1 Introduction

KNN and SVM are two machine learning approaches to Text Categorization (TC) based on the vector space model [Salton *et al*., 1975], a model borrowed from Information Retrieval (IR). Both approaches are known to be among the most accurate text categorizers [Joachims, 1998a; Yang and Liu, 1999]. In the vector space model, documents are represented as a vector where each component is associated with a particular word in the text collection vocabulary.

Generally, each vector component is assigned a value related to the estimated importance (some weight) of the word in the document. Traditionally, this weight was assigned using the TFIDF measure [Joachims, 1998a; Yang and Liu, 1999; Brank *et al*., 2002; Dumais *et al.*, 1998]. While this weighting method seems very appropriate for IR, it is not clear that it is the best choice for TC problems. Actually, this weighting method does not leverage the information implicitly contained in the categorization task to represent documents.

To illustrate this, let us suppose a text collection X and two categorization tasks *A* and *B*. Under the TFIDF weighting representation, each document in *X* is represented by the same vector for both *A* and *B*. Thus, the importance of a word in a document is seen as independent from the categorization task. However, we believe that this should not be the case in many situations. Suppose that *A* is the task that consists of classifying *X* in two categories: documents that pertain to *Computers* and documents that don't. Intuitively, words such as *computer*, *intel* and *keyboard* would be very relevant to this task, but not words such as *the* and *of*; for this reason, the former words should have a higher weight than the latter ones. Suppose, now that *B* consists of classifying *X* in two very different categories: documents written in English and documents written in other languages. It is arguable that in this particular task, words such has *the* (English stop word) and *les* (French stop word), are very relevant. However, under TFIDF, *the* would get a very small weight since its *IDF* (Inverse Document Frequency) would be low. In fact, it would get the same weight that was assigned for task *A*. While this example is somewhat an extreme case, we believe that a weighting approach could benefit from the knowledge about the categorization task at hand.

In this paper, we introduce a new weighting method based on statistical estimation of a word importance for a particular categorization problem. This weighting also has the benefit that it makes feature selection implicit since useless features for the categorization problem considered get a very small weight.

Section 2 presents both the TFIDF weighting function and the new weighting method introduced in this paper. Section 3 describes our evaluation test bed. In section 4, we report results that show significant improvements in terms of classification accuracy.

2 Weighting approaches in text categorization

2.1 TFIDF weighting

`TFIDF` is the most common weighting method used to describe documents in the Vector Space Model, particularly in IR problems. Regarding text categorization, this weighting function has been particularly related to two important machine learning methods: KNN and SVM. The `TFIDF` function weights each vector component (each of them relating to a word of the vocabulary) of each document on the following basis. First, it incorporates the word frequency in the document. Thus, the more a word appears in a document (e.g., its TF, term frequency is high) the more it is estimated to be significant in this document. In addition, IDF measures how infrequent a word is in the collection. This value is estimated using the whole training text collection at hand. Accordingly, if a word is very frequent in the text collection, it is not considered to be particularly representative of this document (since it occurs in most documents; for instance, stop words). In contrast, if the word is infrequent in the text collection, it is believed to be very relevant for the document. `TFIDF` is commonly used in IR to compare a query vector with a document vector using a similarity or distance function such as the cosine similarity function. There are many variants of `TFIDF`. The following common variant was used in our experiments, as found in [Yang and Liu, 1999].[1]

$$weight_{t,d} = \begin{cases} \log(tf_{t,d}+1)\log\dfrac{n}{x_t} & \text{if } tf_{t,d} \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $tf_{t,d}$ is the frequency of word t in document d, n is the number of documents in the text collection and x_t is the number of documents where word t occurs. Normalization to unit length is generally applied to the resulting vectors (unnecessary with KNN and the cosine similarity function).

2.2 Supervised weighting

[Debole and Sebastiani, 2003] have tested and compared some supervised weighting approaches that leverages on the training data. These approaches are variants of `TFIDF` weighting where the *idf* part is modified using common functions used to conduct feature selection. In this paper, their best finding is a variant of the Information Gain, the Gain Ratio. Respective to a category c_i, the Gain Ratio of the term t_k is:

$$GR(t_k, c_i) = \frac{IG(t_k, c_i)}{-\sum_{c \in \{c_i, \bar{c}_i\}} P(c)\log_2 P(c)} \quad (2)$$

Another approach is presented in [Han 1999]. In this study, vector components are weighted using an iterative approach involving the classifier at each step. For each iteration, the weights are slightly modified and the categorization accuracy is measured using an evaluation set (a split from the training set). Convergence of weights should provide an optimal set of weights. While appealing (and probably a near optimal solution if the training data is the only information available to the classifier), this method is generally much too slow to be used, particularly for broad problems (involving a large vocabulary).

2.3 A Weighting Methods based on Confidence

The weighting method (named `ConfWeight` in the rest of the text) introduced in this paper is based on statistical confidence intervals. Let x_t be the number of documents containing the word t in a text collection and n, the size of this text collection. We estimate the proportion of documents containing this term to be:

$$\tilde{p} = \frac{x_t + 0.5z^2_{\alpha/2}}{n + z^2_{\alpha/2}} \quad (3)$$

Where $\tilde{p}$ is the Wilson proportion estimate [Wilson, 1927] and $z^2_{\alpha/2}$ is a value such that $\Phi(z_{\alpha/2}) = \alpha/2$, Φ is the t-distribution (Student's law) function when $n < 30$ and the normal distribution one when $n \geq 30$. So when $n \geq 30$[2], $\tilde{p}$ is:

$$\tilde{p} = \frac{x_t + 1.96}{n + 3.84} \quad (4)$$

Thus, its confidence interval at 95% is:

$$\tilde{p} \pm 1.96\sqrt{\frac{\tilde{p}(1-\tilde{p})}{n+3.84}} \quad (5)$$

Most categorization tasks can be formulated in a way to use only binary classifiers (e.g. a classifier that decides whether a document belongs to a specific category or not). Thus, for a task with n categories, there will be n binary classifiers.

For a given category, let us name $\tilde{p}_+$ the equation (4) applied to the positive documents (those who are labeled as being related to the category) in the training set and $\tilde{p}_-$ to those in the negative class. Now, we use the label *MinPos* for the lower range of the confidence interval of $\tilde{p}_+$, and the label *MaxNeg* for the higher range of that of $\tilde{p}_-$ according to (5) measured on their respective training set. Let now *MinPosRelFreq* be:

$$MinPosRelFreq = \frac{MinPos}{MinPos + MaxNeg} \quad (6)$$

[1] In [Joachims, 1998a], a slight variant is used where the *tf* is used without the logarithm function, but [Yang and Liu, 1999] reports no significant difference in classification accuracy whether the log is applied or not).

[2] When $n < 30$ (which occurs for categories with few positive instances), the t-distribution was used instead of the normal law; thus, equations should be modified accordingly.

We now define the *strength* of term *t* for category +:

$$str_{t,+} = \begin{cases} \log_2(2 \cdot MinPosRelFreq) & \text{if } MinPos > MaxNeg \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Therefore, weight ≠ 0 iff the word appears proportionally more frequently in the + category than in the – category, even in the worst (measured by the confidence interval) estimation error scenario. There might be many categories where weight ≠ 0, since the categorization task is divided in *n* binary classifiers. We name the maximum strength of *t*:

$$\text{maxstr}(t) = \left(\max_{c \in Categories} (str_{t,c}) \right)^2 \quad (8)$$

Maxstr(t) is a global policy technique [Debole and Sebastiani, 2003], that is, the value is that of the best classifier and is thereafter used for all *n* binary classifiers. Using a global policy allows us to use the same document representation for all *n* binary classifiers. While local policies seem intuitively more appealing than global policies when the categorization task is divided in *n* binary problems, [Debole and Sebastiani, 2003] shown that global policies are at least as good as local policies. Note that a value of 0 for maxstr(*t*) is akin to a feature selection deciding to reject the feature.

Figure 1 presents an example to highlight the behavior of eq. (6) to (8). In this figure, *MinPos* is set to 0.5, which means that a hypothetic term occurs at least (recall that this value is the lower range of its relative document frequency confidence interval) in half the documents from the positive set. Then, the curves (labeled (6), (7) and (8) in the graph) consists of the resulting weights for different values of *MaxNeg*. Eq. (6) gives more weight to terms that occur more frequently (relative to the number of documents) in the positive category than in the negative one. Therefore, this weighting method favors features that are **proportionally** more frequent in the positive class. This weight decreases as *MaxNeg* increases. Eq. (7) scales the weight values linearly into the [0,1] range, so that the resulting weight is 0 when a term occurs at the same relative frequency in both classes or proportionally more frequently in the negative set. Finally, Eq. (8) makes the decrease faster, to reflect the rate at which features lose their "energy" as they are more evenly distributed among the positives and the negatives. As a consequence, very predictive features get a high weight, regardless of their absolute frequency (only proportion differences matter).

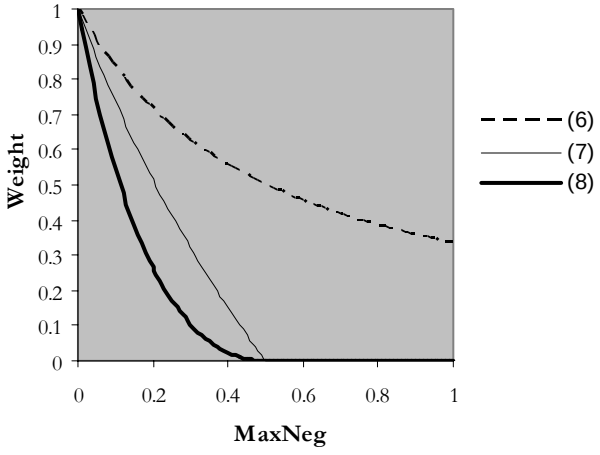

Figure 1: Weight when varying MaxNeg with a fixed MinPos = 0.5

As we are interested in weighting all training and testing documents components in the vector space model, we must use (8) with individual documents, taking the document term frequency into account. We define the `ConfWeight` of *t* in document *d* as:

$$ConfWeight_{t,d} = \log(tf_{t,d} + 1)\text{maxstr}(t) \quad (9)$$

Eq. (9) is quite similar to the `TFIDF` equation in (1): the first part weights the term according to its importance for the document while the second part weights the term globally. However, unlike `TFIDF`, `ConfWeight` uses the categorization problem to determine the weight of a particular term.

3 Methodology

3.1 Corpora

In this paper, three data sets previously studied in the literature have been selected. These datasets are: Reuters-21578, Ohsumed and the new Reuters Corpus Vol 1. Let us briefly describe these datasets.

Reuters-21578 [Lewis, 1997] is made of categories related to business news report. It is written using a limited vocabulary in a succinct manner. We used the ModApte [Lewis, 1997] split. There are 90 categories having at least one training and one testing document. These categories are highly unbalanced. Each document may be categorized in more than one category.

Ohsumed comes from a very large text collection (the MedLine Bibliographical Index) and is rarely used with all available categories and documents. We have chosen to split this text collection as done in [Lewis *et al*., 1996]. The result is a task comprising 49 closely related categories using a very technical vocabulary. Similarly to Reuters, a document may be classified in one or many categories.

Finally, Reuters Corpus Vol. 1 (RCV1) [Rose *et al*., 2001] is a newer text collection released by Reuters Corp. that consists of one full year of new stories. There are about 850,000 documents. 103 categories have documents assigned to them. This collection is very large, thus making

it a very challenging task for learning models such as SVM and KNN, which have polynomial complexity. Particularly, we were not able to use SVM with a large training set since SVM does not scale up very well to large text collections. Using our KNN implementation, we have limited the training set to the first 100,000 documents and the testing set to the next 100,000 documents[3]. An average of 3.15 categories is assigned to each testing document (over 315,000 total assignments).

3.2 Classifiers, feature selection and settings

The weighting method presented in this paper is intended to weight documents in the Vector Space Model. Thus, it can be used only with classifiers using this model. For this reason, we have evaluated our method using both KNN and SVM and compared the results obtained with `TFIDF` and `GainRatio` [Debole and Sebastiani, 2003] weighting.

We have used the SVM^{light} package [Joachims, 1998b] and the KNN classifier described in [Yang and Liu, 1999]. In our experiments with SVM, we divided each categorization task into *n* binary classification problems, as usual. In contrast, KNN is able to classify a document among the *n* categories using one multi-category classifier. To decide whether a document is classified or not in a particular category, thresholds were learned for each category [Yang and Liu, 1999]. `TFIDF` experiments were weighted using Eq. (1) and then normalized to unit length. `GainRatio` experiments were weighted as done by [Debole and Sebastiani, 2003].

To reach optimal classification accuracy, feature selection might be required. Thus, we have included feature selection in our tests. The Information Gain measure has been used to rank the features and many thresholds have been used to filter features out; with `ConfWeight`, in addition to the use of the Information Gain to select features, when *maxstr* (see Eq. 8) was 0, the feature was also rejected. Stop words were not removed and words were not stemmed.

4 Results and discussion

To assess classifier accuracy, a confusion matrix is created for each category:

	Classifier positive label	Classifier negative label
True positive label	A	B
True negative label	C	D

Table 1: Confusion matrix used to evaluate classifier accuracy

For instance, A (the true positives) is the number of documents labeled by the classifier to the category that are correct predictions. Similarly, B (the false negatives) is the number of documents that have not been labeled by the classifier to the category, but that should have.

For any category, the classifier *precision* is defined as A/(A+C) and the *recall* as A/(A+B). To combine these two measures in a single value, the F-measure is often used. The F-measure reflects the relative importance of recall versus precision. When as much importance is granted to precision as it is to recall we have the F1-measure:

$$F1 = \frac{(precision + recall)}{2 \cdot precision \cdot recall} \quad (10)$$

The F1-measure is an estimation of the breakeven point where precision and recall meets if classifier parameters are tuned to balance precision and recall. Since the F1 can be evaluated for each category, we get *n* different F1 values.

To compare two methods, it is needed to combine all the F1 values. In order to do that, two approaches are often used: the macro-F1 average and the micro-F1 average. The macro-F1 average is the simple average of all F1 values; thus each category gets the same weigh in the average. In counterpart, the micro-F1 average weighs large categories more than smaller ones. The micro-F1 is the F1 in (10) where A, B, C and D are global values instead of category-based ones. For instance, A in the micro-F1 is the total number of classifications made by the *n* classifiers that were good predictions. Micro-F1 has been widely used in Text Categorization [Lewis *et al*., 1996; Yang and Liu, 1999; Joachims, 1998a]. Table 2 includes micro-F1 results for SVM while Table 3 includes those of KNN. For each experiment, the best score (among `TFIDF`, `GainRatio` and `ConfWeight`) is bolded.

These results show that at low Information Gain thresholds, `ConfWeight` clearly outperforms both `TFIDF` and `GainRatio`. When more drastic term selection is conducted, overall scores tend to decrease for all three term weighting methods. Is it very interesting to note the very large difference between `ConfWeight` and `TFIDF` using KNN. This difference is particularly significant for a collection of the size of RCV1.

Figure 2, 3 and 4 show the curves resulting from the use of an increasing number of features (decreasing Information Gain thresholds) for each weighting method using KNN. Clearly, `ConfWeight` is the only weighting that doesn't suffer a decrease in accuracy as low-scored features are added. `TFIDF` results are less stable than `ConfWeight` and `GainRatio`, an observation that leads us to claim that `TFIDF` is very sensitive to the choice of feature selection settings.

While `GainRatio` is less sensitive to the presence of all terms (relevant or not) than `TFIDF`, `ConfWeight` seems not to need term selection at all, arguably due to its inherent term selection mechanism. We believe that `ConfWeight` can be used without feature selection and produce very good results.

[3] At the time these experiments were conducted, the LYRL2004 split was not yet released

IGain threshold	Weighting	Reuters 21578	Ohsumed
0	TFIDF	.848	.65
	GainRatio	.875	.702
	ConfWeight	**.877**	**.706**
0.001	TFIDF	.851	.679
	GainRatio	.875	.703
	ConfWeight	**.877**	**.707**
0.005	TFIDF	.875	.695
	GainRatio	.877	**.701**
	ConfWeight	**.882**	.697
0.01	TFIDF	.876	.692
	GainRatio	.877	.696
	ConfWeight	**.882**	**.697**
0.015	TFIDF	**.874**	.693
	GainRatio	.871	.694
	ConfWeight	**.874**	**.696**
0.03	TFIDF	.842	**.658**
	GainRatio	**.844**	**.658**
	ConfWeight	.836	**.658**

Table 2: SVM Micro-F1s by text collection and weighting method

IGain threshold	Weighting	Reuters 21578	Ohsumed	RCV1
0	TFIDF	.816	.588	.785
	GainRatio	.834	.659	.792
	ConfWeight	**.861**	**.683**	**.830**
.001	TFIDF	.819	.645	.812
	GainRatio	.833	.66	.812
	ConfWeight	**.862**	**.687**	**.833**
.005	TFIDF	.843	.621	.828
	GainRatio	**.**840	.661	.817
	ConfWeight	**.861**	**.683**	**.833**
.01	TFIDF	.856	.640	.823
	GainRatio	.834	.659	.822
	ConfWeight	**.864**	**.681**	**.824**
.015	TFIDF	.85	.655	.811
	GainRatio	.832	.654	**.812**
	ConfWeight	**.856**	**.675**	.811
.03	TFIDF	**.832**	.640	.757
	GainRatio	.799	.639	**.765**
	ConfWeight	.824	**.646**	.749

Table 3: KNN Micro-F1s by text collection and weighting method

Another interesting remark is that the best overall scores on each corpora, both using KNN and SVM, are obtained by `ConfWeight` (Reuters-21578: .882 with SVM and .864 with KNN; Ohsumed: .707 with SVM, .687 with KNN; RCV1 .833 with KNN).

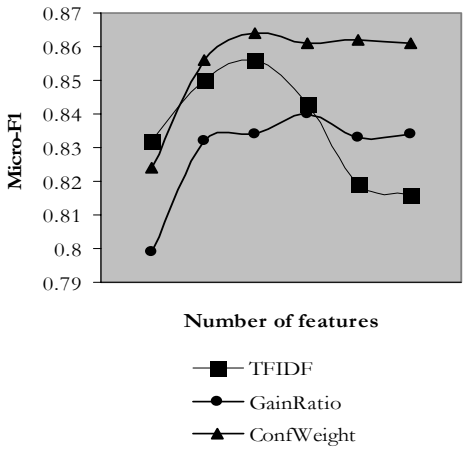

Figure 2: KNN Micro-F1s on Reuters-21578 as the number of feature increases

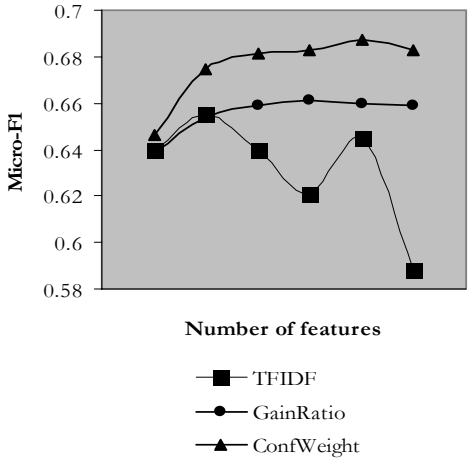

Figure 3: KNN Micro-F1s on Ohsumed as the number of feature increases

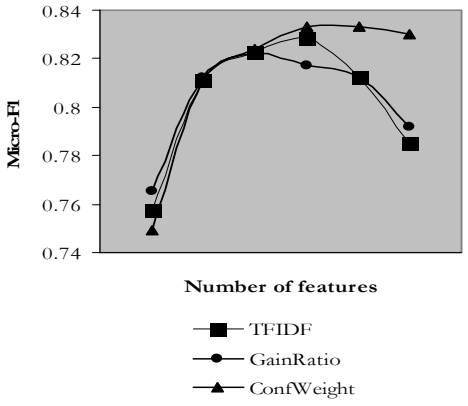

Figure 4: KNN Micro-F1s on RCV1 as the number of feature increases

Finally, we believe that ConfWeight is able to leverage the many features that get a low Information Gain score, which is not always the case with TFIDF and GainRatio. Let us take as an example the TFIDF behavior with SVM in table 2. At .005, there is much less features in the feature space than at .001. Adding features scored between .001 and .005 decreases the Micro-F1 for Reuters-21578 and Ohsumed. On the other hand, the accuracy with ConfWeight increases on Ohsumed if these same low-score features are added to the feature space, while results on Reuters-21578 stay about the same. Using only TFIDF, we might have concluded that features which have an Information Gain lower than 0.005 are harmful for most categorization tasks. Conversely, results so far using ConfWeight tend to show the relevancy and usefulness of low-score features in some settings.

5 Conclusions

In this paper, we have presented a new method (ConfWeight) to weight features in the vector-space model for text categorization by leveraging the categorization task. So far, the most commonly used method is TFIDF, which is unsupervised. To assess our new method, tests have been conducted using three well known text collections: Reuters-21578, Ohsumed and Reuters Corpus Vol. 1. As ConfWeight generally outperformed TFIDF and GainRatio on these text collections, our conclusion is that ConfWeight could be used as a replacement to TFIDF with significant accuracy improvements on the average, as shown in Tables 2 and 3. Moreover, ConfWeight has the ability to perform very well even if no feature selection is conducted, something depicted in the results presented in this paper. Actually, when a feature is irrelevant to the classification task, the weight it gets from ConfWeight is so low that this is merely equivalent to the feature rejection by a feature selection process. TFIDF, on the other hand, always yields a score higher than 0 (if the term occurs in the document for which TFIDF is computed) and this score is not related to the categorization problem, but only to the text collection as a whole. Since feature selection is not inherent to TFIDF, many additional parameters (for instance, the feature selection function to use and thresholds) need to be tuned to achieve optimal results.

[Debole and Sebastiani, 2003] argue for the use of supervised methods to weight features (GainRatio and ConfWeight are two such methods). Despite positive results in some settings, GainRatio failed to show that supervised weighting methods are generally higher than unsupervised ones. We believe that ConfWeight is a promising supervised weighting technique that behaves gracefully both with and without feature selection. Therefore, we advocate its use in further experiments.

References

[Brank *et al.*, 2002] J. Brank, M. Grobelnik, N. Frayling and D. Mladenic. Interaction of Feature Selection Methods and Linear Classification Models, In *Proc. of 19th Conf. on Machine Learning (ICML-02)*, Workshop on Text Learning.

[Debole and Sebastiani, 2003] F. Debole and F. Sebastiani. Supervised term weighting for automated text categorization. In *Proc. of SAC-03, 18th ACM Symposium on Applied Computing*, Melbourne, US, 2003, pp. 784-788.

[Dumais *et al*., 1998] S. Dumais, J. Platt, D. Heckerman and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. of the 1998 ACM 7th International Conference on Information and Knowledge Management*, 148-155, 1998.

[Han 1999] E.H. Han. Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. PhD thesis, University of Minnesota, Oct.1999.

[Joachims, 1998a] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of the European Conference on Machine Learning,* Springer, 1998.

[Joachims, 1998b] T. Joachims, Making Large-Scale SVM Learning Practical. *LS8-Report*, 24, Universität Dortmund, 1998.

[Lewis *et al.*, 1996] D.D. Lewis, R. Schapire, J. Callan, and R. Papka. Training Algorithms for Linear Text Classifiers, In *Proc. of ACM SIGIR*, 298-306, 1996.

[Lewis, 1997] D.D. Lewis. *Reuters-21578 text categorization test collection*, Distrib. 1.0, Sept 26. 1997.

[Rose *et al.*, 2001] T.G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 - from yesterday's news to tomorrow's language resources. In *Proc. of the Third International Conference on Language Resources and Evaluation*, Spain, 29-31 May. 2001.

[Salton *et al.*, 1975] G. Salton, A. Wong, and C.S. Yang. *A vector space model for information retrieval*. Journal of the American Society for Information Science, 18(11):613-620, Nov. 1975.

[Yang and Liu, 1999] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR-99*, 1999.

[Wilson, 1927] E.B. Wilson. Probable Inference, the Law of Succession, and Statistical Inference. *Journal of the American Statistical Association*, 22, 209, 212. 1927.