Random-Walk Term Weighting for Improved Text Classification

Samer Hassan and **Rada Mihalcea** and **Carmen Banea**
Department of Computer Science
University of North Texas
samer@unt.edu, rada@cs.unt.edu, carmenb@unt.edu

Abstract

This paper describes a new approach for estimating term weights in a document, and shows how the new weighting scheme can be used to improve the accuracy of a text classifier. The method uses term co-occurrence as a measure of dependency between word features. A random-walk model is applied on a graph encoding words and co-occurrence dependencies, resulting in scores that represent a quantification of how a particular word feature contributes to a given context. Experiments performed on three standard classification datasets show that the new random-walk based approach outperforms the traditional term frequency approach of feature weighting.

1 Introduction

Term frequency has long been used as a major factor for estimating the probabilistic distribution of features in a document, and it has been employed in a broad spectrum of tasks including language modeling [18], feature selection [29, 24], and term weighting [13, 20]. The main drawback associated with the term frequency method is the fact that it relies on a bag-of-words approach. It implies feature independence, and disregards any dependencies that may exist between words in the text. In other words, it defines a "random choice," where the weight of the term is proportional to the probability of choosing the term randomly from the set of terms that constitute the text. Such an approach might be effective for capturing the relevance of a term in a local context, but it fails to account for the global effect that the term's existence exerts on the entire text segment.

We argue that the bag-of-words model may not be the best technique to capture term importance. Instead, given that relations in the text could be preserved by maintaining the structural representation of the text, a method that takes into account the structural properties of the context could lead to a better term weighting scheme. Previous work has shown that a higher but costly performance can be achieved by incorporating such dependencies [22].

In this paper we introduce a system that models the weighting problem as a "random-walk" rather than "random-choice." We assume an imaginary reader (or "walker") who steps through the text on a term by term basis. In this setting, the importance of the term is determined by the probability of the random-walker to encounter the target term in the text during the walk.

The new measure of term weighting integrates both the locality of a term and its relation to the surrounding context. We model this local contribution using a co-occurrence relation in which terms that co-occur in a certain context are likely to share between them some of their importance (or significance). Note that in this model the relation between a given term and its context is not linear. A given term relates to a context, and the context, in turn, relates to a collection of terms. In order to model this recursive relation, we use a graph-based ranking algorithm, namely the PageRank random-walk algorithm [2], and its TextRank adaptation to text processing [15]. In this paper, we show how TextRank can be used to model the probabilistic distribution of word features in a document. Through experiments performed on a text classification task, we show that the random-walk scores outperform the traditional term frequencies, typically used to model feature weights for this task.

In the following, we first overview the basic principles behind random-walk algorithms, and briefly describe the TextRank application for text processing. We then show how these random-walk models can be adapted to term weighting, and demonstrate that the new weighting scheme can be used to significantly improve the accuracy of a text classification system, as compared to the traditional term frequency weighting scheme. Finally, we conclude with a discussion and directions for future work.

2 Random-Walk Algorithms

The basic idea implemented by a random-walk algorithm is that of "voting" or "recommendation." When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting a vote determines how important the vote itself is, and this information is also taken into account by the ranking algorithm. While there are several random-walk algorithms that have been proposed in the past, we focus on only one such algorithm, namely PageRank [2], as it was previously found successful in a number of applications, including Web link analysis [2], social networks [8], citation analysis, and more recently in several text processing applications [15, 9].

Given a graph $G = (V, E)$, let $In(V_a)$ be the set of vertices that point to vertex V_a (predecessors), and $Out(V_a)$ be the set of vertices that vertex V_a points to (successors). The PageRank score associated with the vertex V_a is defined using a recursive function that integrates the scores of its predecessors:

$$S(V_a) = (1 - d) + d * \sum_{V_b \in In(V_a)} \frac{S(V_b)}{|Out(V_b)|} \quad (1)$$

where d is a parameter that is set between 0 and 1[1].

The score of each vertex is recalculated upon each iteration based on the new weights that the neighboring vertices have accumulated. The algorithm terminates when the convergence point is reached for all the vertices, meaning that the error rate for each vertex falls below a pre-defined threshold.

This vertex scoring scheme is based on a random-walk model, where a walker takes random steps on the graph, with the walk being modeled as a Markov process. Under certain conditions (the graph is aperiodic and irreducible), the model is guaranteed to converge to a stationary distribution of probabilities associated with the vertices in the graph [10]. Intuitively, the stationary probability associated with a vertex represents the probability of finding the walker at that vertex during the random-walk, and thus it represents the importance of the vertex within the graph.

Particularly relevant for our work is the application of random-walks to text processing, as done in the TextRank system [15]. TextRank has been successfully applied to three natural language processing tasks: document summarization, word sense disambiguation, and keyword extraction, with results competitive with those of state-of-the-art systems. The strength of the model lies in the global representation of the context and its ability to model how the co-occurrence between features might propagate across the context and affect other distant features. Our approach follows similar steps as used in the TextRank keyword extraction application, which derives term weights using a graph representation that accounts for the co-occurrence dependencies between words in the text. We are however incorporating a larger number of lexical units, and use different window sizes, as we will show in the following section.

[1] The typical value for d is 0.85 [2], and this is the value we are using in our implementation.

3 Random-Walks for Term Weighting

Starting with a given document, we determine a ranking over the words in the document by using the following models.

3.1 Random-walk Models

In our work, we experimented with several variations of PageRank that incorporate additional information and variables into the traditional version shown in (Equation 1). We summarize the best PageRank-based term ranking models as follows:

$\overleftrightarrow{rw}_o$: It represents the basic or original model, as described in (Equation 1) in which we use an undirected graph with a constant damping factor that adheres strictly to the traditional formula of PageRank.

$\overleftrightarrow{rw}_{e.idf}$: This model represents an undirected graph approach that uses the weighted edge version of PageRank with a variable damping factor. The edge weight is calculated by the following formula:

$$E_{V_1,V_2} = tf.idf_{V_1} * tf.idf_{V_2} \quad (2)$$

where E_{V_1,V_2} is the edge connecting V_1 to V_2, and $tf.idf$ represents the term frequency multiplied by the inverse document frequency.

The damping factor is expressed as a function of the incoming edges' weight, calculated as follows:

$$d_{E_{V_1,V_2}} = E_{V_1,V_2}/E_{max} \quad (3)$$

where $d_{E_{V_1,V_2}}$ is the damping function and E_{max} represents the highest weight for an edge in the graph. The resulting node ranking formula is:

$$S'(V_a) = \frac{(1-d)}{|N|} + \sum_{V_b \in In(V_a)} C * \frac{d_{E_{V_b,V_a}} * S(V_b)}{|Out(V_b)|} \quad (4)$$

where N represents the total number of nodes in the graph and d is the damping constant, C is a scaling constant[2], In order to address the cases where there are no incoming edges, we set the vertex scores in our experiments to $V_{min} = (1 - d)/N$.

The model biases the random walker toward nodes with stronger edges compared to nodes with weaker edges.

$\overleftrightarrow{rw}_{e.oc}$: This model is similar to the above approach however the damping factor for an edge is estimated in terms of the bigram co-occurrence frequency of the two nodes connected by the edge (equation 5). For example, if the bigram "'free software"' occurred four times in a document then the weight of the edge connecting "'free"' and "'software"' is four.

[2] C is a scaling constant which is set to 0.95

$$E_{V_1,V_2} = tf(_{V_1 V_2}) \tag{5}$$

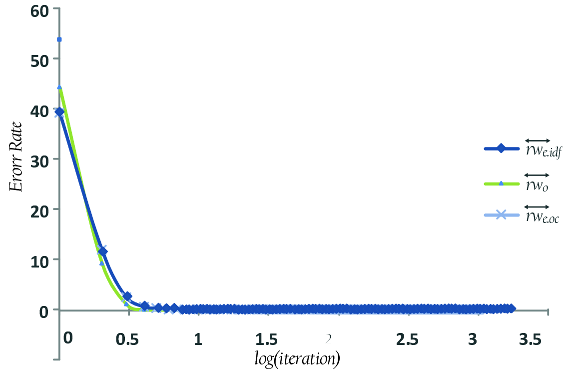

Figure 1. Convergence graphs for the random-walk models

3.2 Term Weighting

Given a document, the following steps are applied to derive a weight associated with the words in the text:

First, the document is tokenized for punctuation, special symbols, and word abbreviations. Common words are also removed, using a list of approximately 500 frequently used words.[3]

Next, the resulting text is processed to extract both term frequency (tf) and random-walk (rw) weights for each term in the document. Note that we do not apply any syntactic filters, as it was previously done in applications of TextRank. Instead, we consider each word as a potential feature. To determine tf, we simply count the frequencies of each word in the document. To determine rw, all the terms are added as vertices in a graph representing the document. A co-occurrence scanner is then applied to the text to relate the terms that co-occur within a given window size. For a given term, all the terms that fall in the vicinity of this term are considered dependent terms. This is represented by a set of edges that connect the term to all the other terms in the window. Experiments are performed for window sizes of 2, 4, 6, and 8. Once the graph is constructed and the edges are in place, the random-walk algorithm is applied.[4] The result of the ranking is a list of all the input terms and their corresponding rw scores.

[3]We use the list of common words distributed with the Smart system ftp://ftp.cs.cornell.edu/pub/smart.

[4]Unless otherwise stated, throughout this paper we refer to a random-walk implementation where the damping factor is set to 0.85, and the convergence threshold to 0.0001. Each graph node is assigned an initial weight of 0.25.

3.3 An Example

To understand why the rw weights might be a good replacement for the traditional tf weights, consider the example in Figure 2, which models a sample document. Starting with this text, a graph is constructed as follows. If a term has not been previously seen, then a node is added to the graph to represent this term. A term can only be represented by one node in the graph. An undirected edge is drawn between two nodes if they co-occur within a certain window size. Figure 3 shows the graph constructed for this text, assuming a window size of 2, corresponding to two consecutive terms in the text (e.g. *London* is linked to *based*).

London-based sugar operator Kaines Ltd confirmed it sold two cargoes of white sugar to India out of an estimated overall sales total of four or five cargoes in which other brokers participated. The sugar, for April/May and April/June shipment, was sold at between 214 and 218 dlrs a tonne cif, it said.

Figure 2. Sample document

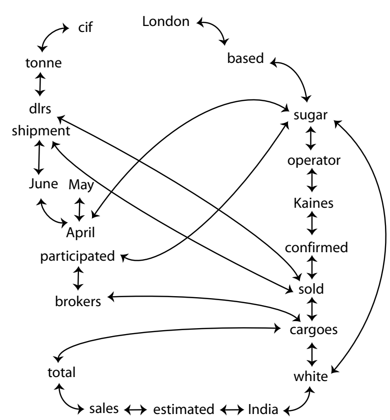

Figure 3. Sample graph

Term	rw	tf	Term	rw	tf
sugar	16.88	3	participated	3.87	1
sold	14.15	2	april	3.87	2
based	7.39	1	india	1.00	1
confirmed	6.90	1	estimated	1.00	1
Kaines	6.81	1	sales	1.00	1
operator	6.76	1	total	1.00	1
London	4.14	1	brokers	1.00	1
cargoes	4.01	2	may	1.00	1
shipment	4.01	1	june	1.00	1
dlrs	4.01	1	tonne	1.00	1
white	3.87	1	cif	1.00	1

Table 1. tf & rw scores for a sample text

After the graph is constructed, the random-walk model is applied on the graph, resulting in a set of scores associated with the vertices (words). Table 1 shows the tf and rw

weights. By analyzing the rw weights, we can observe a non-linear correlation with the tf weights, with an emphasis given to terms surrounding important key-terms such as e.g. *sugar* or *cargoes.* This spatial locality has resulted in higher ranks for terms like *operator* compared to other terms like *london*.[5]

4 Experimental Setup

To evaluate the random-walk based approach to feature weighting, we integrate it in a text classification algorithm, and evaluate its performance on several standard text classification datasets.

Text classification is a problem typically formulated as a machine learning task, where a classifier learns how to distinguish between categories in a given set by using features automatically extracted from a collection of training documents.

We use the tf and rw feature weights (and their alternatives, as described below) to create feature vectors for the Support Vector Machines (SVM) and the Naïve Bayes classifiers. Following standard practice in term weighting for a Rocchio classifier, we use the $tf.idf$ and $rw.idf$ feature weights in our initial evaluation of the models (Tables 2, 3, 5, 6).[6] In the final experiments (Tables 7 and 8) we report tf and $tf.idf$ results separately. The results obtained using tf will act as a baseline for all the evaluations.

4.1 Text Classifiers

We compare the results obtained with three frequently used text classifiers – Rocchio, Naïve Bayes, and SVM, selected based on their performance and diversity of learning methodologies.

Naïve Bayes. The basic idea in a Naïve Bayes text classifier is to estimate the probability of a category given a document using joint probabilities of words and documents. Naïve Bayes text classifiers assume word independence, but despite this simplification, they were shown to perform surprisingly well [11, 23].

Rocchio. The Rocchio text classification method uses standard term weighted vectors to represent documents, and builds a prototype vector for each category by summing up the vectors of the training documents in each category. Test documents are then assigned to the category with the closest prototype vector, based on cosine similarity. Classification experiments with different versions of the algorithm showed competitive results on standard benchmarks [11, 16].

SVM. SVM [27] is a state-of-the-art machine learning approach based on decision plans. The algorithm defines the best hyper-plan that separates the set of points associated with different class labels with a maximum-margin. The unlabeled examples are then classified by deciding on which side of the hyper-surface they reside.

In our evaluations we use $SVMTorch$ [5] with a linear kernel, since it was proved to be as powerful as other kernels in text classification experiments [28]. This SVM implementation is also observed to be the fastest when compared to $SVMlib$ and $Weka's\ SMO$ [12].

4.2 Datasets

We use three standard datasets: $WebKB$, $LingSpam$, and $20Newsgroups$ – commonly used in text classification evaluations [26, 1, 23].

WebKB[7] is a data set collected from computer science departments of various universities. The dataset contains seven class labels: Project, Student, Department, Faculty, Staff, Course, and Other. The Other label was removed from the dataset for evaluation purposes. Most of the evaluations in the literature have been performed on only four of the categories (Project, Student, Faculty, and Course) since they represent the largest categories. However, since we wanted to see how our system behaves when only a few training examples are available, we also considered the Staff and the Department classes which have only a few training documents available. We performed our evaluations on two versions of $WebKB$: one with the four categories version ($WebKB_4$) and one with the six categories ($WebKB_6$).

20**Newsgroups**[8] is a collection of 20,000 messages from 20 newsgroups, corresponding to different topics or subjects. Each newsgroup has about 1000 message split into 400 test and 600 train documents.

LingSpam[9] is a spam corpus [1], consisting of email messages organized in 10 sets to allow for 10-fold cross validation. Each collection has roughly 300 spam and legitimate messages. There are four versions of the corpus standing for bare, stop-word filtered, lemmatized, and stop-word and lemmatized. We use the bare collection with a standard 10-fold cross validation.

5 Evaluation and Discussion

As a first step, we evaluate each of the random-walk models presented in Section 3 ($\overleftrightarrow{rw}_o$, $\overleftrightarrow{rw}_{e.idf}$, and $\overleftrightarrow{rw}_{e.oc}$). Tables 2 and 3 show the micro-average and macro-average accuracy figures for each model, classifier, and dataset for a window size of 2. The tf column shows the results ob-

[5] All the missing words e.g. *Ltd, it,* not shown in the graph are common words that were eliminated during pre-processing.

[6] We refer to this results as $Rocchio^{idf}$

[7] http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

[8] http://people.csail.mit.edu/jrennie/20Newsgroups

[9] http://boole.cs.iastate.edu/book/acad/bag/data/lingspam

[9] $*$ indicates a statistically significant result where $0.05 > \rho > 0.001$. The result is marked by $**$ when $\rho \leq 0.001$.

dataset	tf	$\overleftrightarrow{rw}_{e.oc}$	$\overleftrightarrow{rw}_{e.idf}$	$\overleftrightarrow{rw}_{o}$
Naïve Bayes				
$WebKB_4$	84.2	**86.1***	85.8*	86.1**
$WebKB_6$	81.3	82.4	81.9	**83.3***
$LSpam$	99.2	**99.3**	99.2	99.3
$20NG$	89.3	91.5**	**91.7****	90.6**
$Rocchio^{idf}$				
$WebKB_4$	84.3	87.5**	**87.7****	86.9**
$WebKB_6$	80.1	**84.3****	84.2**	83.4**
$LSpam$	98.1	**98.5**	98.3	98.2
$20NG$	91.5	**94.3****	93.8**	93.0**
SVM				
$WebKB_4$	81.3	**94.1****	93.2**	80.3
$WebKB_6$	79.3	**90.7****	90.4**	78.9
$LSpam$	93.6	98.9**	**99****	92.1
$20NG$	90.1	**94.4****	94.4**	86.9

Table 2. Micro-Average results for different random-walk models

dataset	tf	$\overleftrightarrow{rw}_{e.oc}$	$\overleftrightarrow{rw}_{e.idf}$	$\overleftrightarrow{rw}_{o}$
Naïve Bayes				
$WebKB_4$	82.5	83.5*	82.9*	**84.2****
$WebKB_6$	68.1	66.5	65.5	**70.0***
$LSpam$	98.6	**98.8**	98.6	98.8
$20NG$	89.3	**91.4****	90.6**	90.5**
$Rocchio^{idf}$				
$WebKB_4$	84.3	**86.7****	86.7**	86.1**
$WebKB_6$	71.6	**75.8****	75.3**	75.1**
$LSpam$	96.7	**97.3**	97.1	96.9
$20NG$	91.5	**94.3****	93.7**	93.0**
SVM				
$WebKB_4$	77.4	**93.2****	93.2**	80.3
$WebKB_6$	68.2	81.1**	**81.5****	65.2
$LSpam$	86.3	97.9**	**98.1****	81.2
$20NG$	90.6	**94.4****	94.4**	88.1

Table 3. Macro-average results for different random-walk models

tained using the term frequency weighting scheme, which, as stated before, acts as a baseline throughout all our experiments.

As seen in the tables, most of the models presented perform better than the tf baseline. Both the $\overleftrightarrow{rw}_{e.oc}$ and $\overleftrightarrow{rw}_{e.idf}$ models stand out as the best performing models with noticeable improvements, especially for the SVM and the Rocchio classifiers.

The $\overleftrightarrow{rw}_{e.idf}$ and $\overleftrightarrow{rw}_{e.oc}$ models redefine the random jump component of PageRank, by considering the damping factor as a function that can be estimated per edge. A highly connected node with relatively strong edges would tend to encourage the random-walker to following its outgoing links rather than randomly jumping out. The consideration given to the relative weight of the edges signifies the encapsulation of global information in the biasing factor. This allows us, in a sense, to steer the random-walker toward useful nodes more effectively, which is valuable in emphasizing the discriminative power of central features.

In addition to accuracy, we also evaluated the efficiency of the new models. By comparing the processing time for 1000 $WebKB4$ documents using the proposed models, we notice a small overhead of 53 seconds for the $\overleftrightarrow{rw}_{o}$ model and 26 seconds for the $\overleftrightarrow{rw}_{e.oc}$ and $\overleftrightarrow{rw}_{e.idf}$, as compared to the tf baseline.[10] This is due to the fast convergence of these models in approximately 15 iterations (Figure 1). We believe that this small increase in processing time is a reasonable cost for achieving significantly higher accuracies.

tf	$\overleftrightarrow{rw}_{e.oc}$	$\overleftrightarrow{rw}_{e.idf}$	$\overleftrightarrow{rw}_{o}$
112	138	138	165

Table 4. Running time in seconds for the processing of 1000 documents from $WebKB_4$, on a Pentium-IV machine with 2Gb RAM.

5.1 Different Window Sizes

Among the various models, the $\overleftrightarrow{rw}_{e.oc}$ model seems to consistently outperform the other models. To take a closer look at this model, we further analyze it under different window sizes. Table 5 and 6 show the $\overleftrightarrow{rw}_{e.oc}$ classification results for $WebKB_4$, $WebKB_6$, $LingSpam$, $20Newsgroups$ respectively. The rw_2, rw_4, rw_6, and rw_8 represent the accuracies achieved using the $\overleftrightarrow{rw}_{e.oc}$ weighting scheme under window sizes of 2, 4, 6, and 8 respectively.

dataset	tf	rw_2	rw_4	rw_6	rw_8
Naïve Bayes					
$WebKB_4$	84.2	[11]**86.1***	85.8*	85.8*	85.7*
$WebKB_6$	81.3	**82.4***	81.9*	81.8	76.6
$LSpam$	99.2	99.3	**99.3**	99.3	99.3
$20NG$	89.3	**91.5**	91.2	91.2	91.2
$Rocchio^{idf}$					
$WebKB_4$	84.3	87.5**	87.5**	87.4**	**87.6****
$WebKB_6$	80.1	84.3**	84.0**	84.3**	**84.4****
$LSpam$	98.1	**98.5**	98.3	98.4	98.3
$20NG$	91.5	94.3**	**94.3****	94.2**	94.2**
SVM					
$WebKB_4$	81.3	**94.1****	93.6**	93.5**	93.7**
$WebKB_6$	79.3	**90.7****	90.6**	90.7**	90.6**
$LSpam$	93.6	98.9**	**99.1****	99.1**	99.0**
$20NG$	90.1	94.4**	94.5**	94.5**	**94.6****

Table 5. Micro-average results for the $\overleftrightarrow{rw}_{e.oc}$ random-walk model for different window sizes

[10]This time includes tokenization and stopword removal.

dataset	tf	rw_2	rw_4	rw_6	rw_8
Naïve Bayes					
$WebKB_4$	82.5	[12]**83.5***	83.0*	82.9*	82.6*
$WebKB_6$	68.1	**66.5***	66.0*	65.6	62.1
$LSpam$	98.6	98.8	**98.8**	98.8	98.8
$20NG$	89.3	**91.4**	91.2	91.1	91.1
$Rocchio^{idf}$					
$WebKB_4$	83.4	**86.7****	86.6**	86.5**	86.7**
$WebKB_6$	71.6	75.8**	75.5**	75.7**	**76.2****
$LSpam$	96.7	**97.3**	97.1	97.3	97.1
$20NG$	91.5	**94.3****	94.2**	94.2**	94.2**
SVM					
$WebKB_4$	77.4	**93.2****	92.9**	92.7**	92.9**
$WebKB_6$	68.2	81.1**	**81.5****	81.3**	81.3**
$LSpam$	86.3	97.9**	98.3**	**98.4****	98.2**
$20NG$	90.6	94.4**	94.5**	94.5**	**94.6****

Table 6. Macro-average results for the $\overleftrightarrow{rw}_{e.oc}$ random-walk model for different window sizes

The system displays consistent performance across different window sizes. By further analyzing the results using statistical t-tests we notice that windows of size 2 and 4 supply the most significant results across all the classifiers and the datasets.

Comparing the tf and rw weighting schemes for the $WebKB_6$ dataset, we found that both schemes failed to predict the class Staff. However, a significant improvement was obtained over the class Department, in which our rw model scores an accuracy of 47% compared to 4% when using tf. This could be due to the ability of the model to extract more realistic and smoother distribution of terms, hence reducing the feature bias imposed by the limited number of training examples.

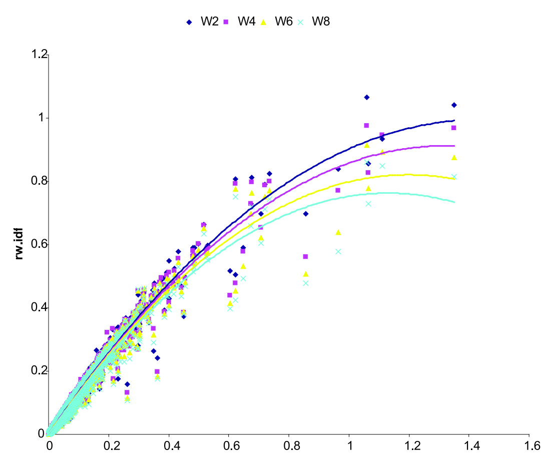

Figure 4. Correlations of the tf and rw models for the WebKB4 collection

We also notice a positive impact of using our model with the Rocchio classifier. In order to reach a deeper understanding of the correlation between the two models in the context of the Rocchio classifier, we macro-averaged the $tf.idf$ and the $rw.idf$ term scores over all the $WebKB4$ documents, in a manner similar to the construction of prototype classification vectors but taking into account the entire corpus rather than individual classes. The resulting term scores are plotted in Figure 4. Polynomial approximations are used to visualize the trends of the plotted points, where each trendline represents a different window size.

By analyzing the graph we can distinguish three interesting properties:

1. A clear non-linear correlation between the tf and rw models.

2. An increasing drift from the tf model as we increase the window size.

3. A smoothing effect associated with the rw model, with a growth rate of the tf values clearly faster than rw.

5.2 Other Weighting Scheme Policies

We also compared our models to other reported state-of-the-art weighting schemes:

$tf.idf$**:** since its introduction [25], $tf.idf$ has been one of the most extensively studied weighting schemes [21, 17].It served as a standard baseline in term weighting studies [4, 13] and proved hard to beat in [7].

itf**:** defined as one minus the inverse term frequency, was found to have an excellent performance when used with an SVM linear kernel [14, 6].

$log(tf)$**:** first introduced in [3], it was recently used with the purpose of smoothing term frequencies and hence minimizing the feature bias [19, 4].

$log(tf).idf$**:** this scheme was suggested in [3], and it showed superior performance in [4]. In this scheme, the smoothed term frequency is scaled by its idf to confer higher weights to domain relevant features.

$\sqrt{tf}$**:** due to the interesting trends observed in figure 4, we introduced an approximation of the rw/tf correlation using the square root $\sqrt{tf}$ function, which exhibits the general behavior of the plotted curves.

For each of the schemes, we also introduce the rw alternative, by replacing tf in all of the presented schemes with the rw values calculated using our random-walk model. For instance, for the $tf.idf$ scheme, we introduce a $rw.idf$ scheme; for $log(tf)$, we introduce $log(rw)$; and so forth.

dataset	tf	rw	$tf.idf$	$rw.idf$	$\sqrt{tf}$	$\sqrt{rw}$	$log(tf)$	$log(rw)$	itf	irw	$log(tf).idf$	$log(rw).idf$
Naïve Bayes												
$WebKB_4$	84.2	**86.1***	81.6	**83.8***	85.2	**85.9**	85.1	**85.9**	83.3	83.3	83.2	**83.8**
$WebKB_6$	81.3	**82.4**	78.8	**81.5***	81.5	**81.9**	81.6	**81.9**	77.3	**77.7**	80.4	**81.7**
$LSpam$	99.2	**99.3**	99.1	99.1	99.3	99.3	99.3	**99.4**	99.3	**99.4**	99.0	99.1
$20NG$	89.3	**91.5****	86.9	**88.6****	90.8	**91.4**	77.3	**91.4***	91.2	**91.5**	76.7	**88.4****
Rocchio												
$WebKB_4$	74.9	**83.5****	84.3	**87.5****	81.2	**84.5****	80.6	**84.2****	82.9	**84.7***	86.5	**87.8**
$WebKB_6$	70.1	**78.6****	80.1	**84.3****	76.3	**79.9****	75.3	**79.5****	78.1	**80.1***	82.9	**84.3**
$LSpam$	96.5	**97.8****	98.1	**98.5**	97.4	**98.1**	97.3	**97.9**	97.6	**98.1**	98.2	**98.5**
$20NG$	90.5	**94.8****	91.5	**94.3****	94.1	**94.8***	67.0	**94.9****	94.6	**94.7**	76.1	**94.4****
SVM												
$WebKB_4$	81.3	**94.1****	88.1	**90.4**	92.0	**93.8***	91.4	**94.0***	92.6	**92.9**	89.8	**90.6**
$WebKB_6$	79.3	**90.7****	87.1	**88.7***	89.7	**90.8**	89.9	**90.9***	**89.9**	89.6	88.1	**88.6**
$LSpam$	93.6	**98.9****	93.6	**93.9**	98.2	**99.1***	97.6	**99.1****	98.6	**98.7**	94.5	**94.7**
$20NG$	90.1	**94.4****	92.5	**93.2***	92.0	**93.9**	77.6	**94.5****	94.6	**94.7**	93.1	**93.3**

Table 7. Micro-average Results for different weighting schemes ($\overleftrightarrow{rw}_{e.oc}$)

dataset	tf	rw	$tf.idf$	$rw.idf$	$\sqrt{tf}$	$\sqrt{rw}$	$log(tf)$	$log(rw)$	itf	irw	$log(tf).idf$	$log(rw).idf$
Naïve Bayes												
$WebKB_4$	82.5	**83.5***	78.9	**81.2***	82.4	**83.1**	82.4	**83.2**	**78.1**	77.5	80.6	**81.1**
$WebKB_6$	**68.1**	66.5	68.5	**69.8***	**66.3**	65.6	**66.9**	65.7	**52.6**	52.3	69.4	**70.2**
$LSpam$	98.6	**98.8**	98.4	98.4	98.7	**98.8**	98.8	**98.9**	98.9	98.9	98.3	**98.4**
$20NG$	89.3	**91.4****	86.8	**88.5****	90.7	**91.3**	77.1	**91.3***	91.1	**91.4**	76.5	**88.4****
Rocchio												
$WebKB_4$	74.4	**82.6****	83.4	**86.7****	80.3	**83.5****	79.7	**83.2****	82.0	**83.7***	85.7	**86.9**
$WebKB_6$	62.7	**71.9****	71.6	**75.8****	69.6	**73.2****	68.6	**72.7****	71.5	**73.1***	74.7	**75.9**
$LSpam$	94.1	**96.2****	96.7	**97.3**	95.5	**96.7**	95.5	**96.5**	96.0	**96.7**	97.0	**97.4**
$20NG$	90.5	**94.8****	91.5	**94.3****	94.2	**94.8***	68.1	**94.9****	94.6	**94.8**	76.1	**94.4****
SVM												
$WebKB_4$	77.4	**93.2****	85.8	**88.6****	90.8	**93.0***	90.1	**93.2***	91.7	**91.9**	87.9	**88.8**
$WebKB_6$	68.2	**81.1****	75.9	**77.3***	81.0	**82.0**	80.0	**81.7***	**82.1**	81.3	77.1	**77.2**
$LSpam$	86.3	**97.9****	86.1	**86.8**	96.6	**98.3***	95.4	**98.3****	97.6	**97.7**	88.4	**88.8**
$20NG$	90.6	**94.4****	92.6	**93.1***	90.8	**94.0**	77.6	**94.5****	94.6	**94.7**	93.1	**93.3**

Table 8. Macro-average results for different weighting schemes ($\overleftrightarrow{rw}_{e.oc}$)

The classification results obtained for the different weighting schemes on the three datasets are shown in Tables 7 and 8. Statistical significance tests were run to compare the performance of the rw and tf alternatives for each of the weighting schemes.

As seen in the tables, our random-walk models clearly outperform the term frequency alternative in both micro and macro averages under all datasets and classifiers. In the worst case, the system performs as good as the baseline model. The superiority of our rw models indicate that the use of dependencies between features can lead to significant improvements, and these improvements are consistent for different weighting schemes.

6 Conclusions and Future Work

In this paper, we introduced a random-walk approach for term weighting that has the ability to capture term dependencies in a text by accounting for the structural properties of the text. Through experiments performed on a text classification task, we showed that the random-walk model can achieve relative error rate reductions of 3.2–84.3%, as compared to the traditional term frequency based approach. The evaluation results have shown that the system's performance is consistent for various window sizes, and its running time is comparable to the $tf.idf$ model.

Additionally, in experiments carried out using a variety of weighting scheme policies, the random-walk term weighting was consistently found superior as compared to the traditional term frequency weighting scheme.

We believe these results support our claim that random-walk models can accurately estimate term weights by accounting for term dependencies, and can be used as a technique to model the probabilistic distribution of features in a document.

In future work we plan to extend the model and use it to define a formal language model, in which we can estimate the probability of longer n-gram sequences of words.

Acknowledgments

This work was supported in part by a research grant from the Texas Advanced Research Program (#003594).

References

[1] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the workshop on Machine Learning in the New Information Age*, 2000.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7), 1998.

[3] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *Proceedings of the Text Retrieval Conference*, 1994.

[4] P. D. Ciya Liao, Shamim Alpha. Feature preparation in text categorization. In *Oracle Corporation*, 2002.

[5] R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[6] P. Dai, U. Iurgel, and G. Rigoll. A novel feature combination approach for spoken document classification with support vector machines, 2003.

[7] F. Debole and F. Sebastiani. Supervised term weighting for automated text categorization. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 784–788, New York, NY, USA, 2003. ACM Press.

[8] B. Dom, I. Eiron, A. Cozzi, and Y. Shang. Graph-based ranking algorithms for e-mail expertise analysis. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, San Diego, California, 2003.

[9] G. Erkan and D. Radev. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, December 2004.

[10] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 1989.

[11] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, Nashville, US, 1997.

[12] A. Klautau. Speech recognition based on discriminative classifiers. In *Proceedings of the Simposio Brasileiro de Telecomunicacion-SBT*, Rio de Janeiro, Brazil, 2003.

[13] M. Lan, C. Tan, H. Low, and S. Sungy. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *Proceedings of the 14th international conference on World Wide Web*, pages 1032–1033, 2005.

[14] E. Leopold and J. Kindermann. Text categorization with support vector machines. how to represent texts in input space? In *Machine Learning*, volume 46, pages 423–444, Hingham, MA, USA, 2002. Kluwer Academic Publishers.

[15] R. Mihalcea and P. Tarau. TextRank – bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain, 2004.

[16] A. Moschitti. A study on optimal paramter tuning for Rocchio text classifier. In *Proceedings of the European Conference on Information Retrieval*, Pisa, Italy, 2003.

[17] K. Papineni. Why inverse document frequency? In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[18] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, pages 275–281, 1998.

[19] M. Radovanovic and M. Ivanovic. Document representations for classification of short web-page descriptions. In *DaWaK*, pages 544–553, 2006.

[20] R. Robertson and K. Sparck-Jones. Simple, proven approaches to text retrieval. Technical report, 1997.

[21] S. Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 5:503–520, 2004.

[22] M. Sahami. Learning limited dependence bayesian classifiers. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

[23] K. Schneider. A new feature selection score for multinomial naive bayes text classification based on kl-divergence. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, July 2004.

[24] H. Schutze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, Seattle, Washington, 1995.

[25] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[26] S. Tan, X. Cheng, M. M. Ghanem, B. Wang, and H. Xu. A novel refinement approach for text categorization. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 469–476, Bremen, Germany, 2005.

[27] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[28] Y. Yang and X. Liu. A reexamination of text categorization methods. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[29] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, Nashville, US, 1997.