

# Text Categorization Using Word Similarities Based on Higher Order Co-occurrences<sup>\*†</sup>

Syed Fawad Hussain<sup>1</sup>

Gilles Bisson<sup>1</sup>

## Abstract

In this paper, we propose an extension of the  $\chi$ -Sim clustering algorithm to deal with the text categorization task. The idea behind  $\chi$ -Sim method [1] is to iteratively learn the similarity matrix between documents using similarity matrix between words and vice-versa. Thus, two documents are said to be similar if they share similar (but not necessary identical) words and two words are similar if they occur in similar documents. The algorithm has been shown to work well for unsupervised document clustering. By introducing some “a priori” knowledge about the class labels of documents in the initialization step of  $\chi$ -Sim, we are able to extend the method to deal for the supervised task. The proposed approach is tested on different classical textual datasets and our experiments show that the proposed algorithm compares favorably or surpass both traditional and state-of-the-art algorithms like  $k$ -NN, supervised LSI and SVM.

**Keywords:** Text categorization, clustering, Higher-order co-occurrences, supervised learning.

## 1. Introduction

Text categorization is defined as the task of assigning predefined categories to new unlabeled documents. Text corpora are usually represented with the Vector Space Model (VSM) popularized by [2]. In this model, documents are represented as vectors over a set of dimensions, each representing a word in the corpus dictionary. Thus, the corpus is represented as a matrix where each row is a document and each column is a word. The entries in the matrix signify the number of occurrences of a word in a given document. Despite the widespread usage of the VSM, it has been argued that

traditional classification algorithms such as  $k$ -nearest neighbors ( $k$ -NN), Naïve Bayesian and Support Vector Machines (SVM) do not scale up for the task of text categorization with the increase of dimensionality mainly because of their inability to handle synonymy and polysemy [3], [4]. Moreover, it is a well known problem that when represented in such a high-dimensional form, document vectors tend to be highly sparse and suffer from the *curse of dimensionality* [4]. In such a scenario, traditional distance measures such as Euclidean or Cosine do not always make much sense [5]. As a result, several other approaches have been proposed to overcome these limitations by exploiting the dual relationship between documents and words to extract semantic knowledge from the data.

The concept of ‘higher-order’ co-occurrences has been investigated [6], [7], among others, as a measure of semantic relationship between words. The underlying analogy is that humans do not necessarily use the same vocabulary when writing about the same topic. For instance, Lemaire and Denhière [7] report finding 131 occurrences of the words “*internet*” and 94 occurrences of the word “*web*” but no co-occurrence at all in a corpus of 24-million words collected from the French newspaper *Le Monde*. It is evident, however, that these two words have a strong relationship. This relationship can be brought to light if the two words co-occur with other words in the corpus. For example, consider a document set containing significant number of co-occurrences between the words “*sea*” and “*waves*” and another document set in which the words “*ocean*” and “*waves*” co-occur. We could infer that the worlds “*ocean*” and “*sea*” are conceptually related even if they do not directly co-occur in any document. Such a relationship between “*waves*” and “*ocean*” (or “*sea*” and “*waves*”) is termed as a first-order co-occurrence. This conceptual association between “*sea*” and “*ocean*” is called a second-order relationship. The concept can be generalized to higher (3rd, 4th, 5th, etc) order co-occurrences.

The relationship between documents and words also allows for exploitation of the dual nature of the problem i.e. the relationship between groups of words that occur mostly in a group of documents. Many algorithms have

---

<sup>1</sup> Laboratoire TIMC-IMAG, CNRS / UJF 5525, University of Grenoble, France. {fawad.hussain, gilles.bisson}@imag.fr

\* This work is part of a PhD thesis funded by the Higher Education Commission, Government of Pakistan.

† This work is partially supported by the French ANR project FRAGRANCES under grant 2008-CORD 00801.

been proposed mostly in the unsupervised domain that exploits this dual nature of this relationship such as the SVD based LSI and information theoretic approaches [1], [8], [9]. The recently introduced co-similarity based algorithm (called  $\chi$ -Sim) [1] exploits both the duality between words and documents in a document-term matrix as well as their respective higher-order co-occurrences in the corpus.

The  $\chi$ -Sim algorithm however, like clustering algorithms in general, have an inherent limitation when applied to the categorization task in that it does not exploit the additional information regarding the document's category label in the training data. Several works in the LSI framework have shown that by incorporating class knowledge, we can, without modifying the basic algorithm, obtain good results on classification problems [10], [11], [12]. Moreover, using word clustering has also been shown to improve the result of the text categorization [13], [14]. Motivated by these results, in this paper we expand the unsupervised  $\chi$ -Sim algorithm to the supervised classification task by exploiting information about category labels from the training dataset.

We propose a two-step approach to incorporate class knowledge into the learned similarity matrices used by  $\chi$ -Sim. Firstly, we introduce (as in [11]) class specific columns into the training dataset that incorporates the category information about the class. The basic concept behind this is that since the  $\chi$ -Sim algorithm explicitly captures higher-order similarities, adding category specific columns will force higher-order co-occurrences. Secondly, we introduce a mechanism whereby the similarity values of documents belonging to different categories are reduced. This helps to promote the influence of higher order co-occurrences between documents within category to be higher. Thus documents belonging to the same category are brought closer together and documents belonging to different categories are blown farther apart. In the same way, due to the iterative process used in  $\chi$ -Sim (as explained in section 2), words representative of a document category are brought relatively closer together than words representative of different document categories.

The rest of this paper is organized as follows. In section 2, we briefly explain the unsupervised  $\chi$ -Sim algorithm. Section 3 analyses the theoretical interpretation of the algorithm using random walks in a bi-partite graph model. In section 4, we propose a natural extension of the algorithm to incorporate category knowledge into the algorithm. Section 5 reviews some of the related work in the literature and section 6 provides some empirical results and analysis of our experimentations.

## 2. The Unsupervised $\chi$ -Sim Algorithm

We will use the classical notation: matrices (in capital letters) and vectors (in small letters) are in bold and all variables are in italic.

*Data matrix:* let  $\mathbf{M}$  be the data matrix representing a corpus having  $r$  rows (documents) and  $c$  columns (words);  $m_{ij}$  corresponds to the number of occurrences of the  $j$ th word in the  $i$ th document;  $\mathbf{m}_i = [m_{i1} \dots m_{ic}]$  is the row vector representing the document  $i$  and  $\mathbf{m}^j = [m_{1j} \dots m_{rj}]$  is the column vector corresponding to word  $j$ . We will refer to a document as  $d_1, d_2, \dots$  when talking about documents casually and refer to it as  $\mathbf{m}_1, \mathbf{m}_2, \dots$  when specifying its (row) vector in the matrix  $\mathbf{M}$ . Similarly, we will refer to a word as  $w_1, w_2, \dots$  when talking about words 1, 2, ... and use the notation  $\mathbf{m}^1, \mathbf{m}^2, \dots$  when emphasizing the word vector.

*Similarity matrices:*  $\mathbf{SR}$  and  $\mathbf{SC}$  represent the square and symmetrical Row Similarity and Column Similarity matrices of size  $r \times r$  and  $c \times c$  respectively with  $sr_{ij} \in [0, 1]$ ,  $1 \leq i, j \leq r$  and  $sc_{ij} \in [0, 1]$ ,  $1 \leq i, j \leq c$ ;  $\mathbf{SC}_i = [sc_{i1} \dots sc_{ic}]$  (respectively  $\mathbf{SR}_i = [sr_{j1} \dots sr_{jc}]$ ) is the similarity vector between the word  $i$  and all the other words (respectively between the document  $j$  and the other documents).

*Similarity function:* function  $F_s(\dots)$  is a generic similarity function that takes two elements  $m_{ij}$  and  $m_{kl}$  of  $\mathbf{M}$  as input and returns a measure of the similarity  $F_s(m_{ij}, m_{kl})$  between these two elements. For the sake of brevity, we will also use the shorthand notation  $\vec{F}(m_{ij}, \mathbf{m}_k)$  to represent a vector that contains the pair-wise similarity  $F_s(\dots)$  between the scalar  $m_{ij}$  and each element of the vector  $\mathbf{m}_k$ .

**2.1 Calculating the Co-Similarities.** The  $\chi$ -Sim algorithm is a *co-similarity* based approach which builds on the idea of iteratively generating the similarity matrices  $\mathbf{SR}$  (between documents) and  $\mathbf{SC}$  (between words), each of them built on the basis of the other.

First, we present an intuitive idea of how to compute the co-similarity matrix  $\mathbf{SR}$  between rows. Usually, the similarity (or distance) measure between two documents  $\mathbf{m}_i$  and  $\mathbf{m}_j$  is defined as a function - denoted here as  $Sim(\mathbf{m}_i, \mathbf{m}_j)$  - that is the sum of the similarities between words occurring in both  $\mathbf{m}_i$  and  $\mathbf{m}_j$  as

$$(2.1) \quad Sim(\mathbf{m}_i, \mathbf{m}_j) = F_s(m_{i1}, m_{j1}) + \dots + F_s(m_{ic}, m_{jc})$$

Other factors may be introduced, for instance to normalize the similarity in the interval  $[0, 1]$ , but the main idea is always the same: we consider the values between columns having the same indices. Now let's suppose we have a matrix  $\mathbf{SC}$  whose entries provide a measure of similarity between the columns (words) of

the corpus. Equation (2.1) can be re-written as follows without changing its meaning if  $sc_{i,i}=1$ :

$$(2.2) \quad Sim(\mathbf{m}_i, \mathbf{m}_j) = F_s(m_{i1}, m_{j1}) \cdot sc_{11} + \dots + F_s(m_{ic}, m_{jc}) \cdot sc_{cc}$$

Here our idea is to generalize equation (2.2) in order to take into account all the possible pairs of *features* (words) occurring in documents  $\mathbf{m}_i$  and  $\mathbf{m}_j$ . In this way, not only do we “capture” the similarity of their common words but also the similarity coming from words that are not directly shared by the two documents but that are considered to be *similar*. For each pair of words not directly shared by the documents, we take into account their similarity as provided by the **SC** matrix. Thus the overall similarity between documents  $\mathbf{m}_i$  and  $\mathbf{m}_j$  is defined in equation (2.3) in which the terms in the boxes are those occurring in Equation(2.2):

$$(2.3) \quad Sim(\mathbf{m}_i, \mathbf{m}_j) = \boxed{F_s(m_{i1}, m_{j1}) \cdot sc_{11}} + F_s(m_{i1}, m_{j2}) \cdot sc_{12} + \dots + F_s(m_{i1}, m_{jc}) \cdot sc_{1c} + F_s(m_{i2}, m_{j1}) \cdot sc_{21} + \boxed{F_s(m_{i2}, m_{j2}) \cdot sc_{22}} + \dots + F_s(m_{i2}, m_{jc}) \cdot sc_{2c} + \dots + F_s(m_{ic}, m_{j1}) \cdot sc_{c1} + F_s(m_{ic}, m_{j2}) \cdot sc_{c2} + \dots + \boxed{F_s(m_{ic}, m_{jc}) \cdot sc_{cc}}$$

By using our shorthand notation for representing a row vector, we may re-write the above formula as follows

$$(2.4) \quad Sim(\mathbf{m}_i, \mathbf{m}_j) = \bar{F}(m_{i1}, \mathbf{m}_j) \bullet \mathbf{sc}_1 + \dots + \bar{F}(m_{ic}, \mathbf{m}_j) \bullet \mathbf{sc}_c$$

where “ $\bullet$ ” represents a dot product.

Conversely, when to express the similarity between two words (columns)  $\mathbf{m}^i$  and  $\mathbf{m}^j$  of **M**, we use the same approach. Hence:

$$(2.5) \quad Sim(\mathbf{m}^i, \mathbf{m}^j) = \bar{F}(\mathbf{m}_{i1}, \mathbf{m}^j) \bullet \mathbf{sr}_1 + \dots + \bar{F}(\mathbf{m}_{ir}, \mathbf{m}^j) \bullet \mathbf{sr}_r$$

The combination of equations (2.4) and (2.5) exploit the dual relationship between documents and words in a corpus i.e. documents are similar when they are expressed using similar words. Words, in turn are similar when they occur in similar documents.

As stated previously, each element of the matrices **SR** and **SC** must belong to the interval [0, 1], since we define the maximum similarity between two documents (or words) to be unity, but neither  $Sim(\mathbf{m}_i, \mathbf{m}_j)$  nor  $Sim(\mathbf{m}^i, \mathbf{m}^j)$  verify this property. We now re-introduce the normalization factor for the  $\chi$ -Sim to verify this property. Since the maximum value defined by the function  $sc_{ij}$  is 1, it follows from equation (2.4) that the upper bound of  $Sim(\mathbf{m}_i, \mathbf{m}_j)$  ( $1 \leq i, j \leq r$ ) is given by the product of the sum of elements of  $\mathbf{m}_i$  and  $\mathbf{m}_j$  denoted by

$\mu_i$  and  $\mu_j$  i.e.  $\mu_i = \sum_k m_{ik}$  and  $\mu_j = \sum_k m_{jk}$  respectively.

The normalization function when comparing two documents is therefore defined as  $\mu_i \mu_j$  and when comparing two words as  $\mu^i \mu^j$ . This normalization is particularly suited for textual datasets since it allows us to take into consideration the actual length of the document and word vectors when dealing between pairs of documents or words of uneven length, which is typically the case.

**2.2 Generalization of the approach.** We describe here a generic formulation of several *classical* similarity measures between two documents. Given two documents  $\mathbf{m}_i$  and  $\mathbf{m}_j$ , the similarity measure can be expressed as a product of matrices given by

$$(2.6) \quad Sim(\mathbf{m}_i, \mathbf{m}_j) = \frac{\mathbf{m}_i (\mathbf{SC}) \mathbf{m}_j^T}{N(\mathbf{m}_i, \mathbf{m}_j)}$$

where  $\mathbf{m}^T$  denotes the transpose of  $\mathbf{m}$  and  $N(\mathbf{m}_i, \mathbf{m}_j)$  is a normalization function that depends on  $\mathbf{m}_i$  and  $\mathbf{m}_j$  used to map the similarity function to a particular interval, such as [0,1]. Equation (2.6) can be seen as a generalization of several similarity measures. For example, the Cosine similarity measure can be written using the above equation, where **SC** is set to the identity matrix, as

$$(2.7) \quad \text{Cosine}(\mathbf{m}_i, \mathbf{m}_j) = \frac{\mathbf{m}_i (\mathbf{I}) \mathbf{m}_j^T}{\|\mathbf{m}_i\| \|\mathbf{m}_j\|}$$

where  $\|\mathbf{m}_i\|$  represents the  $L_2$  norm of the vector  $\mathbf{m}_i$ . Similarly, the Jaccard index can be obtained by setting **SC** to **I** and  $N(\mathbf{m}_i, \mathbf{m}_j)$  to  $|\mathbf{m}_i| + |\mathbf{m}_j| - \mathbf{m}_i \mathbf{m}_j^T$  ( $|\mathbf{m}_i|$  denotes  $L_1$  norm), while the Dice Coefficient can be obtained by setting the **SC** to  $2\mathbf{I}$  and  $N(\mathbf{m}_i, \mathbf{m}_j)$  to  $|\mathbf{m}_i| + |\mathbf{m}_j|$ . Note that by setting the **SC** matrix to identity, we define the similarity between a word and itself to be 1 (maximal) and between every non-identical pair of word to be 0.

Now if we define our function  $F_s$  as a product of the elements  $m_{ij}$  and  $m_{kl}$  i.e.  $F_s(m_{ij}, m_{kl}) = m_{ij} m_{kl}$  (similar to the cosine similarity measure), then the similarity value between documents  $\mathbf{m}_i$  and  $\mathbf{m}_j$ , as expressed in equation (2.4), can be expressed in the form of equation (2.6). Here the numerator from equation (2.6) corresponds to the terms in equation (2.4) and the denominator corresponds to the normalization factor  $\mu_i \mu_j$  as described previously. We can now re-write how to compute the elements  $sr_{ij}$  ( $1 \leq i, j \leq r$ ) of **SR** as,

$$(2.8) \quad \forall i, j \in 1..r, sr_{ij} = Sim(\mathbf{m}_i, \mathbf{m}_j) = \frac{\mathbf{m}_i (\mathbf{SC}) \mathbf{m}_j^T}{\mu_i \mu_j}$$

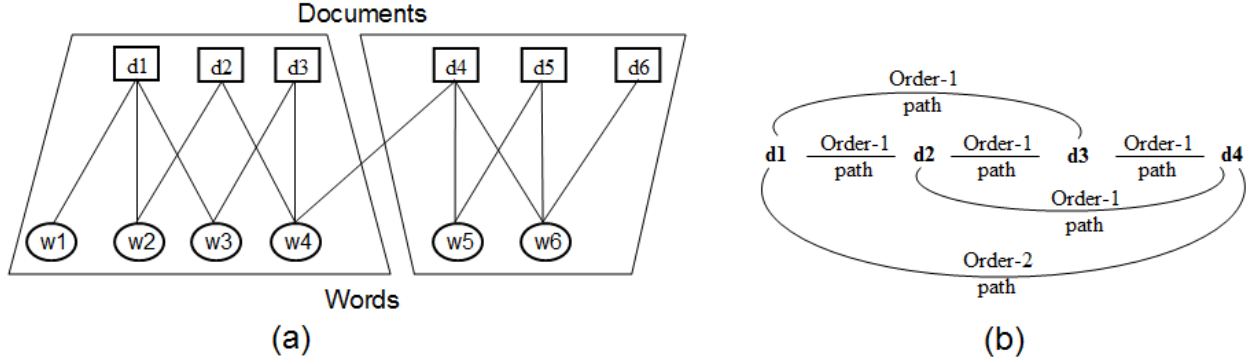


Figure 1: (a) A bi-partite graph view of the matrix  $D$ . The square vertices represent documents and the rounded vertices represent words, and (b) some of the higher order co-occurrences between documents in the bi-partite graph.

Similarly, the elements  $sc_{ij}$  ( $1 \leq i, j \leq c$ ) of the word similarity matrix  $\mathbf{SC}$  can be computed as

$$(2.9) \quad \forall i, j \in 1..c, sc_{ij} = Sim(\mathbf{m}^i, \mathbf{m}^j) = \frac{\mathbf{m}^i (\mathbf{SR}) \mathbf{m}^j}{\mu^i \mu^j}$$

As opposed to the other similarity measures mentioned above like the cosine measure in equation (2.7), the similarity values in (2.8) and (2.9) differ in two aspects: firstly, the non-diagonal elements of the  $\mathbf{SC}$  matrix are not zero; and secondly the values in the  $\mathbf{SC}$  matrix are defined as a function of another matrix  $\mathbf{SR}$  and the two are iteratively computed.

**2.3 The  $\chi$ -Sim Algorithm.** Equations (2.8) and (2.9) allows us to compute the similarities between two documents and two words. The extension over all pair of documents and all pair of words can be generalized as a matrix multiplication. The algorithm follows:

1. We initialize the similarity matrices  $\mathbf{SR}$  (documents) and  $\mathbf{SC}$  (words) with the identity matrix  $\mathbf{I}$ , since, at the first iteration, only the similarity between a document (resp. word) and itself equals 1 and zero for all other documents (resp. words). We denote these matrices as  $\mathbf{SC}^{(0)}$  and  $\mathbf{SR}^{(0)}$  where the superscript denotes the iteration.
2. At each iteration  $t$ , we calculate the new similarity matrix between documents  $\mathbf{SR}^{(t)}$  by using the similarity matrix between words  $\mathbf{SC}^{(t-1)}$ . To normalize the values, we take the Hamard<sup>3</sup> product between the matrix  $\mathbf{SR}^{(t)}$  and a pre-calculated matrix  $\mathbf{NR}$  defined as  $\forall i, j \in [1, r], nr_{ij} = 1/N(\mathbf{m}_i, \mathbf{m}_j)$ . We do

the same thing for the similarity matrix  $\mathbf{SC}^{(t)}$  and normalize it using  $\mathbf{NC}$  given by  $\forall i, j \in [1, c], nc_{ij} = 1/N(\mathbf{m}^i, \mathbf{m}^j)$ . The two relations are as follows:

$$(2.10) \quad \mathbf{SR}^{(t)} = (\mathbf{M} \bullet \mathbf{SC}^{(t-1)} \bullet \mathbf{M}^T) \otimes \mathbf{NR}, nr_{ij} = \frac{1}{\mu_i \mu_j}$$

$$(2.11) \quad \mathbf{SC}^{(t)} = (\mathbf{M}^T \bullet \mathbf{SR}^{(t-1)} \bullet \mathbf{M}) \otimes \mathbf{NC}, nc_{ij} = \frac{1}{\mu^i \mu^j}$$

where ‘ $\bullet$ ’ and ‘ $\otimes$ ’ denotes the matrix and Hadamard multiplication respectively.

3. Set the diagonal of  $\mathbf{SR}^{(t)}$  and  $\mathbf{SC}^{(t)}$  to 1 to force the fact that the similarity between a document and itself must be maximal ( the same for the words).
4. Step 2-4 are repeated  $t$  times to iteratively update  $\mathbf{SR}^{(t)}$  and  $\mathbf{SC}^{(t)}$ .

It is worth noting here that even though  $\chi$ -Sim computes the similarity between each pair of documents using all pair of words, the complexity of the algorithm remains comparable to classical similarity measures like cosine or Euclidean distances. To update  $\mathbf{SR}$  (or  $\mathbf{SC}$ ) we just have to multiply three matrices using equations (2.10) and (2.11) respectively. Given that the complexity of matrix multiplication<sup>4</sup> is in  $O(n^3)$  (for a generalized matrix of size  $n$  by  $n$ ), the overall complexity of  $\chi$ -Sim is given by  $O(tn^3)$  where  $t$  is the number of iterations.

The natural question that arises is the number of iterations to be performed. Before we can answer this question, we need to understand the meaning of an iteration, which we present in the next section.

<sup>3</sup> In a Hadamard product  $\mathbf{A}=\mathbf{B}\otimes\mathbf{C}$ , the elements  $a_{i,k}$  of matrix  $\mathbf{A}$  are defined as:  $a_{i,k}=b_{i,k} \cdot c_{i,k}$

<sup>4</sup> The complexity of the Hadamard product is  $O(n^2)$

### 3. Theoretical Interpretation

We now present a graph theoretical interpretation of the algorithm which would enable us to better understand the working of the algorithm and give us some intuition on how to exploit the category labels from the training dataset. Consider the bi-partite graph representation of a sample data matrix in Fig. 1(a) having 6 documents  $d_1$ - $d_6$  and 6 words  $w_1$ - $w_6$ . The documents and words are represented by rectangular and oval nodes respectively and an edge between a document  $d_i$  and a word  $w_j$  in the graph corresponds to the entry  $m_{ij}$  in the document-term matrix. There is only one order-1 path between documents  $d_1$  and  $d_2$  given by  $d_1 \xrightarrow{m_{12}} w_2 \xrightarrow{m_{22}} d_2$ .

Hence the similarity value  $sr_{12}$  is given by the product  $m_{12}m_{22}$ . Note that since the **SC** matrix is initialized as identity, at the first iteration,  $sr_{12}$  just corresponds to the dot product between  $\mathbf{m}_1$  and  $\mathbf{m}_2$  (since  $sc_{kl}=0$  for all  $k \neq l$ ) as given by equation(2.8). The matrix  $\mathbf{SR}^{(1)} = \mathbf{M} * \mathbf{M}^T$  thus represents all order-1 paths between the pair of documents  $\mathbf{m}_i$  and  $\mathbf{m}_j$  ( $i, j = 1..r$ ). Similarly, each element of  $\mathbf{SC}^{(1)} = \mathbf{M}^T * \mathbf{M}$  represents an order-1 path between words  $\mathbf{m}^i$  and  $\mathbf{m}^j$  ( $i, j = 1..c$ ). We omit the normalization factors as it clear from equations (2.10) and (2.11) which normalization is to be used.

Documents  $d_1$  and  $d_4$  do not have an order-1 path but are linked together by  $d_2$  and  $d_3$ . The similarity value contributed via the document  $d_2$  can be explicitly represented as  $d_1 \xrightarrow{m_{12}} w_2 \xrightarrow{m_{22}} d_2 \xrightarrow{m_{24}} w_4 \xrightarrow{m_{44}} d_4$ . The sub-sequence  $w_2 \rightarrow d_2 \rightarrow w_4$  represents an order-1 path between words  $w_2$  and  $w_4$  which is the same as  $sc_{24}^{(1)}$ . The contribution of  $d_2$  in the similarity of  $sr_{14}^{(1)}$  via  $d_2$  can thus be re-written as  $m_{12}sc_{24}^{(1)}m_{44}$ . This is a partial similarity measure since  $d_2$  is not the only document that provides a link between  $d_1$  and  $d_4$ . The similarity via  $d_3$  (see figure 1(b)) is equal to  $m_{13}sc_{34}^{(1)}m_{44}$ . To find the overall similarity measure between documents  $d_1$  and  $d_4$ , we need to add these partial similarity values given by  $m_{12}sc_{24}^{(1)}m_{44} + m_{13}sc_{34}^{(1)}m_{44}$ . Incidentally, this similarity is given by the product of our matrices in equation (2.10) ( $\mathbf{SR}^{(2)} = \mathbf{D} * \mathbf{SC}^{(1)} * \mathbf{D}^T$ ). Hence, the similarity matrix  $\mathbf{SR}^{(2)}$  at the second iteration corresponds to paths of order-2 between documents. It can be shown similarly that, in general, the matrices  $\mathbf{SR}^{(t)}$  and  $\mathbf{SC}^{(t)}$  represent an order- $t$  path between documents and between words respectively.

We can now define the number of iterations to perform for  $\chi$ -Sim. At each iteration  $t$ , one or more new links may be found between previously disjoint objects (documents or words) corresponding to paths with length of order- $t$ ; and existing similarity measures may be

strengthened since higher-order links signifies more semantic relatedness. It has been shown that “in the long run”, the ending point of a random walk does not depend on its starting point [15] and hence it is possible to find a path (and hence similarity) between any pair of nodes in a connected graph [16] by iterating a sufficiently large number of times. However, co-occurrences beyond the 3rd and 4th order have little semantic relevance and hence not interesting [1], [7]. Therefore, the number of iterations is usually limited to 4 or less.

### 4. Exploiting Category Information

There are two potential ways to incorporate background knowledge in text mining – by using external knowledge such as external thesauri or repositories [16], [17], or by incorporating background knowledge about category information of documents from a training dataset whose category labels are known. Since the  $\chi$ -Sim algorithm uses both the matrix regarding similarity about documents **SR** and the matrix similarity about terms **SC**, it is possible to incorporate such additional knowledge within the framework of the algorithm. For example, if we had a before hand knowledge about the semantic relationship between words in a data corpus, we could incorporate this in the **SC** matrix by modifying the initialization step. In this paper, however, we are concerned with the case of supervised text categorization, with a training dataset whose category labels are known beforehand.

Given a training dataset  $\mathbf{m}_1, \dots, \mathbf{m}_{r1}$  with  $r1$  labeled examples defined over  $c$  words with discrete (but not necessarily binary) category labels, we denote a document  $i$  from this dataset as  $\mathbf{m}_i^{\text{train}}$  to denote the fact that its category label is known. The  $r1$  examples in the training set form the document-term matrix  $\mathbf{M}_{\text{train}}$ . Let  $X$  be a set of  $n$  possible document categories ( $X = \{x_1, x_2, \dots, x_n\}$  and  $|X|=n$ ), we denote by  $(\mathbf{m}_i^{\text{train}}, x_j)$ ,  $1 \leq i \leq r1$ ,  $x_j \in X$  a document  $i$  in the training set whose class label is characterized by  $x_j$ . Similarly, let  $\mathbf{m}_1^{\text{test}}, \dots, \mathbf{m}_{r2}^{\text{test}}$  be a test dataset of  $r2$  labeled examples (also with discrete but not necessarily binary) category labels but whose labels are not known to the algorithm. The  $r2$  examples in the test set form the document-term matrix  $\mathbf{M}_{\text{test}}$  also defined over  $c$  words. Note that  $c$  is the size of the dictionary of the corpus. We wish to learn the matrix **SC** that reflects category labels of the documents in  $\mathbf{M}_{\text{train}}$  as given by equation(2.8), and use the learnt **SC** matrix to categorize each document in  $\mathbf{M}_{\text{test}}$  to one of  $X$  categories.

The similarity measure described in the previous section is purely unsupervised. The **SR** and **SC** matrices are initialized with an identity matrix since in an unsupervised environment; we have no prior knowledge

about the similarity between two documents or two words. As such, only the similarity between a document and itself or a word and itself is initialized with 1 and all other values are set to 0. However, with the availability of category labels from  $\mathbf{M}_{\text{train}}$ , it is possible to influence the similarity values such that documents belonging to the same category are brought closer together and farther away from documents belonging to a difference category. The overall objective is that when using a classification algorithm such as the  $k$  Nearest Neighbor ( $k$ -NN) approach, a document vector  $\mathbf{m}_i$  will find its  $k$  nearest neighbors within the same document categories as itself since documents belonging to this category have higher similarity values.

It is worth noting here that since the similarity between two documents  $\mathbf{m}_i$  and  $\mathbf{m}_j$  is defined by equation(2.8), we would like to influence the **SC** matrix such that words representative of a document category (those words that occur mostly in documents of one category) are brought closer to each other and word representative of different document categories drawn farther apart. But word similarities, in turn, are given by equation (2.9) so we would also like to incorporate the category information in the **SR** matrix such that documents belonging to the same category are brought closer to each other and farther away from documents belonging to difference categories. We present below a two-pronged strategy of incorporating category information as 1) increasing within category similarities, and 2) decreasing out-of-category similarities.

**4.1 Increasing within category similarity values.** As stated above, our intention is to bring words representative of a document category closer together. One way of doing this, since we know the category labels of the documents, is by padding each document in the training set by an additional *dummy word* that incorporates the category information for that class. The original and revised document-term matrix  $\mathbf{M}_{\text{train}}$  is shown in figure 2.

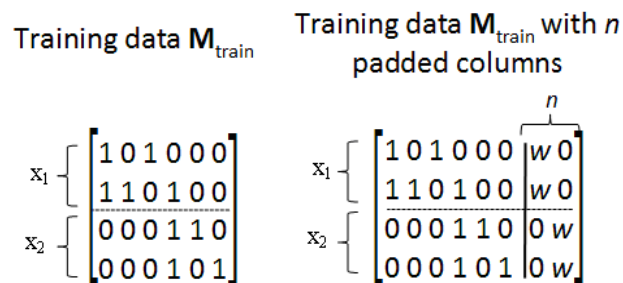


Figure 2. Incorporating category information into **SC** by padding  $n$  columns.

The entries for each of these padded columns are weighted by  $w$ . When we apply equation (2.9) to calculate word similarities in the revised matrix, the padded columns *force* second order co-occurrences between words in the same class. As a result, the word representatives of a class are brought closer together. The value of  $w$ , which corresponds to the number of times each of this *dummy word* occurs in all documents of a document category, is used to emphasize the category knowledge. Using a very small value for  $w$  would result in little influence on the similarity values. On the other hand, using a large value for  $w$  will distort the finer relationships in the original data matrix. The value of  $w$  can be determined empirically from the training dataset. This is further explored in the section relating to experimentations (section 6).

**4.2 Decreasing out of category similarity values.** For the document similarity matrix, we proceed with the same objective in mind as previously. Augmenting class information as described above not only brings words closer together but also enhances the similarity between documents of the same class since each of the padded dummy word is present in all documents of the same document category.

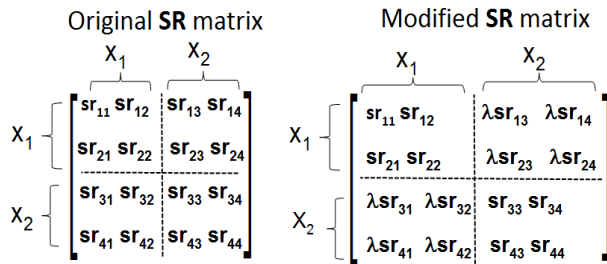


Figure 3. Reducing similarity between documents from different categories.

However, many words transcend document categorical boundaries and, as such, generate similarities between documents belonging to difference categories. This phenomenon is cascaded when **SR** and **SC** are iterated. The drawback of this is that higher order word similarities are mined based on co-occurrences resulting from documents from different categories. We wish to reduce the influence of such higher order co-occurrences when mining for word similarities. We propose a weighting mechanism that reduces the similarity value between documents that do not belong to the same category. The mechanism is shown in figure 3. At each iteration, the values  $sr_{ij}$  ( $x_i \neq x_j$ ) are multiplied by a weighting factor  $\lambda$  ( $0 \leq \lambda \leq 1$ ). The parameter  $\lambda$

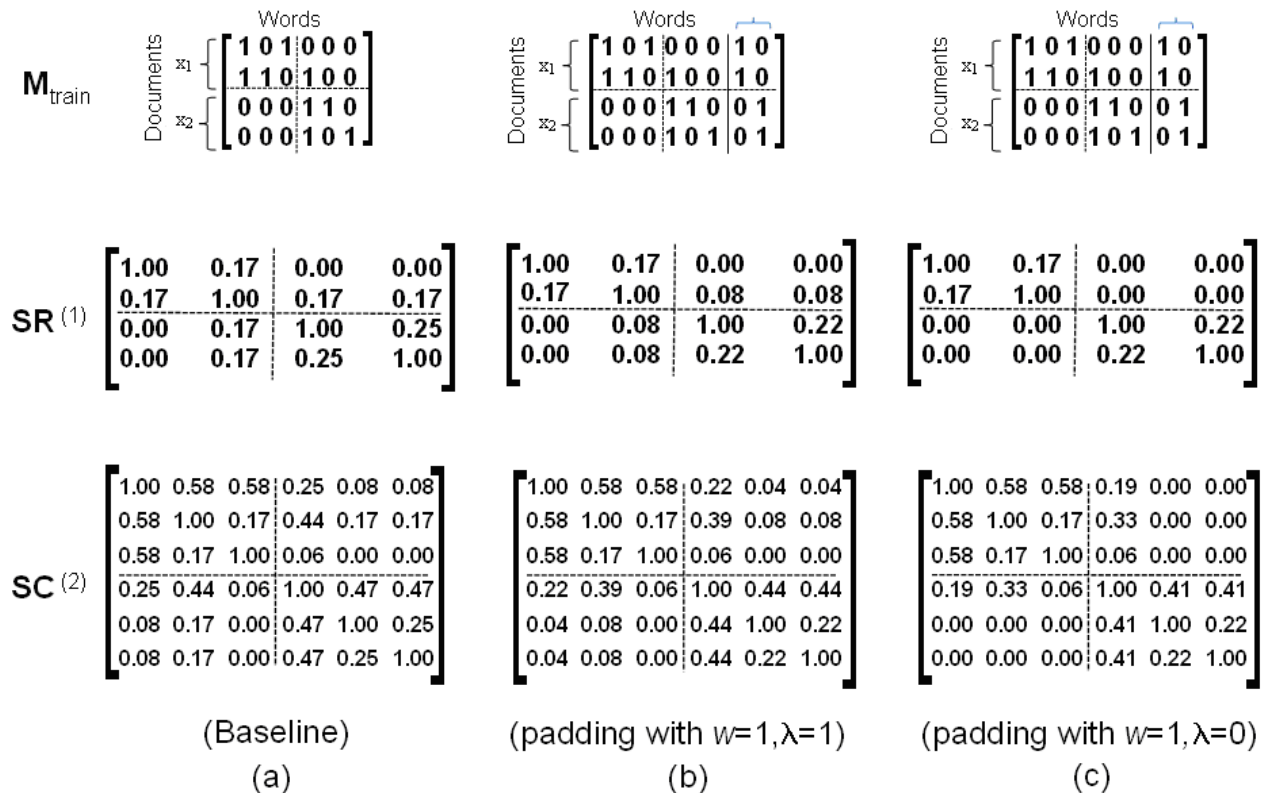


Figure 4: Sample  $\mathbf{SR}$  and  $\mathbf{SC}$  matrices on (a) the training dataset  $\mathbf{M}_{\text{train}}$  with no categorical information, (b) padding  $\mathbf{M}_{\text{train}}$  with  $n$  dummy words incorporating class knowledge, and (c) padding the  $\mathbf{M}_{\text{train}}$  matrix and setting similarity values between out-of-class documents to zero.

determines the influence of category information on the higher order word co-occurrences. If  $\lambda=0$ , we force word co-occurrences to be generated from documents of the same category only while  $\lambda=1$  corresponds to the unsupervised framework which relaxes this constraint and all higher order occurrences contribute equally to the similarity measure. The value of the parameter  $\lambda$  can be thought of as task dependent. Intuitively, for the task of document categorization, highly discriminative words should have relatively higher similarity values than words that occur frequently in more than one document category, hence a small value of  $\lambda$  is desirable. Intuitively, for the task of document categorization, highly discriminative words should have higher similarity and hence a small value of  $\lambda$  is desirable. We will see the effect of  $\lambda$  in the section about empirical results when we perform document categorization task on text dataset.

**4.3 Analysis.** We describe here a small example to analyze the effect of incorporating category knowledge on the evolution of document and word similarity

matrices using a toy example given in Figure 4. Given a training data document by term matrix,  $\mathbf{M}_{\text{train}}$  with 4 documents belong to two categories ( $x_1$  and  $x_2$ ) and defined over 6 words. There is only one word that provides a link between the documents in the two categories (column 4). Figure 4 shows the input document-term matrix  $\mathbf{M}_{\text{train}}$ , the document similarity matrix  $\mathbf{SR}$  and the word similarity matrix  $\mathbf{SC}$  at iteration 1 and 2 respectively using various methods of incorporating category information. Figure 4(a) shows the evolution of the similarity matrices without incorporating any class knowledge.

Documents  $d_1$  and  $d_2$  belong to category 1 while documents  $d_3$  and  $d_4$  belong to category 2. First we consider the document similarity between documents of the same category, for example  $d_1$  and  $d_2$ , and documents corresponding to different categories, for example  $d_2$  and  $d_3$ . Without augmenting  $\mathbf{M}_{\text{train}}$  with category discriminating columns, the similarity between  $d_1$  and  $d_2$  and between  $d_2$  and  $d_3$  is the same i.e. 0.17. Next we add a category discriminating column for each class as described in section 4.1. For the sake of simplicity, we

set the value of  $w=1$  (figure 4b). One can immediately see the effect of this on the document similarity matrix  $\mathbf{SR}^{(1)}$ . Notice that the similarity between  $d_1$  and  $d_2$  remains 0.17 instead of increasing. This is because of the normalization factor since  $N(\mathbf{m}_1, \mathbf{m}_2)$  in our case is a product of the sum of words in  $d_1$  and  $d_2$ . The normalization factor typically increases more than the numerator in equations (2.8) and (2.9). In comparison to the similarity value between  $d_2$  and  $d_3$ , however, the similarity between  $d_1$  and  $d_2$  is now relatively stronger as a result of adding the category information.

Now consider the similarities between  $w_1$  and other words in figure 4(a). Words  $w_1$  and  $w_2$  for example, both of which only appear in documents of the first category, have a similarity value of 0.58 while words  $w_1$  and  $w_5$  where  $w_5$  only occurs in documents of category 2 have a similarity of 0.08. The link between  $w_1$  and  $w_5$  is provided by  $w_4$  which is shared by documents  $d_2$  and  $d_3$ . Adding category information slightly decreases the similarity value between words 1 and 5 to 0.04 as can be seen in figure 4(b). This reduction in similarity of objects belonging to different classes is a result of the cascading effect of inter-twining  $\mathbf{SR}$  and  $\mathbf{SC}$  matrices since the documents 2 and 3 now have a lower similarity value as seen above. When embedding the category knowledge into the  $\mathbf{SR}$  matrix, the effect is much stronger. This effect is even greater in figure 4(c) where the similarity value between  $w_1$  and  $w_5$  vanishes since the value  $sr_{23}$  was zeroed by the weighting factor,  $\lambda$ . By varying the value of  $\lambda$ , we can explicitly control the contribution of such links to the similarity measure.

**4.4 Labeling the Test dataset.** Since the  $\mathbf{SC}$  matrix is constructed on the dictionary of words  $c$ , we can now use the learned  $\mathbf{SC}$  matrix during the training phase to classify the test examples. Firstly, we use the popular  $k$ -Nearest Neighbors algorithm to associate a test dataset to exactly one of the  $X$  document categories. To compare a document vector in the test matrix, say  $\mathbf{m}_j^{\text{test}}$ , with a document vector from the training matrix, say  $\mathbf{m}_i^{\text{train}}$ , we compute their similarity value  $\text{Sim}(\mathbf{m}_i^{\text{train}}, \mathbf{m}_j^{\text{test}})$  as

$$(4.1) \quad \text{Sim}(\mathbf{m}_i^{\text{train}}, \mathbf{m}_j^{\text{test}}) = \frac{\mathbf{m}_i^{\text{train}} (\mathbf{SC}) \mathbf{m}_j^{\text{test}^T}}{\mu_i^{\text{train}}, \mu_j^{\text{test}}}$$

where  $(\mathbf{m}_i^{\text{train}})^T$  is the transpose of the matrix  $(\mathbf{m}_i^{\text{train}})$ . The category of each test document is then decided by a majority weight of its  $k$ -Nearest Neighbors with highest similarity values.

The above approach necessitates the comparison of each test document with all training documents which could be an expensive operation when the number of

documents in the training set is large. Here, we also explore a second approach which is significantly faster. Instead of comparing each test document with all training documents from every category, we form a category vector  $\mathbf{v}_i^{\text{cat}}, j \in [1..X]$ . The category vector  $\mathbf{v}_j^{\text{cat}}$  is defined as the sum of all the document vectors belonging to the category  $x_i$  as  $\mathbf{v}_i^{\text{cat}} = \sum_{k=1..r_l} (\mathbf{m}_{ki})$  that satisfy the condition  $(\mathbf{m}_k, x_i)$ . The test documents is then assigned to the category having the highest similarity value with the  $\mathbf{m}_j^{\text{test}}$ , given by

$$(4.2) \quad \text{Sim}(\mathbf{v}_i^{\text{cat}}, \mathbf{m}_j^{\text{test}}) = \frac{\mathbf{v}_i^{\text{cat}} (\mathbf{SC}) \mathbf{m}_j^{\text{test}^T}}{\mu_i^{\text{cat}}, \mu_j^{\text{test}}}$$

Notice that in this case, we only need to do  $|X|$  comparisons (where  $|X|$  is the number of document categories). This approach is similar to the Rocchio classification algorithm [18]. However, unlike Rocchio, our category prototype is defined solely by summing the category documents and does not take into account the out of category documents.

## 5. Related Work

Several clustering algorithms have been expanded to the supervised classification case. For example [4] expands the information bottleneck approach and [13] expands on the maximum likelihood criteria for the text categorization task.

Closer to our work, [19] have proposed using higher order co-occurrences as a measure of word similarities. They propose an algorithm to calculate paths of order-1, 2, and 3 between words which they store in different similarity matrices. A genetic algorithm is used to search for optional weighting mechanism for combining these matrices. Class knowledge is exploited by adding class discriminating words (sprinkling) and a CRN (Case Retrieval Network) is used for assign class labels to test documents using a  $k$ -NN approach. Unlike the  $\chi$ -Sim algorithm, however, their approach does not exploit the dual nature relationship between documents i.e. words are similar irrespective of the relationship between documents that generate the higher order paths. The algorithm is also expensive in time/space requirement as they maintain a different word similarity matrix corresponding to each order- $n$  path and search for optimal weights for combining them.

Several other works has been done to incorporated prior knowledge into the SVD based LSI approach which has also been shown to implicitly exploit higher order relationships [20]. The sprinkling approach was proposed by [11] as a measure of enriching the reduced lower-dimensional ‘concept space’. This approach,

however, is dependent on the sensitive nature of LSI to the number of lower-dimensions to which the matrix is projected. The authors in [12] proposed a way to select different columns for different classes based on the observed confusion matrices in an approach called Adaptive Sprinkling (AS). However, their proposed trick in turn suffers from the problem of choosing the optimal maximal sprinkled columns (MSL) and the classification problem still remains sensitive to the number of lower dimensions chosen. Adding more columns no longer guarantees the  $k$ -rank approximation to the original matrix which the LSI method is based on. In another study, the authors in [18] proposed a way called SSLI to incorporate class knowledge by iteratively identifying the most discriminative eigenvectors in class specific LSI representations.

One advantage of using supervised knowledge into the  $\chi$ -Sim is that it is possible to explicitly incorporate class information as opposed to indirectly introducing like in the case of LSI. Moreover, initializing either the **SR** or **SC** similarity matrix incur no additional cost to  $\chi$ -Sim algorithm.

## 6. Empirical Results

This section provides an empirical study to show the benefits of the proposed technique. We evaluate the proposed method with several real text corpora. Specifically, we perform the task of text categorization which involves a training dataset whose document category labels are provided to the algorithm and use the learned word similarities to assign each document of the test dataset into one of the document categories.

**6.1 Datasets.** We evaluate the proposed techniques on several real datasets. We used 3 popular text corpora which have been widely used in the literature for both classification and clustering tasks and created several datasets. The first corpus, the 20-Newsgroup, consists of approximately 20 000 newsgroup articles that have been collected evenly from 20 different Usenet groups. Many of the newsgroups share similar topics and about 4.5% of the documents are cross-posted making the boundaries between some newsgroups fuzzy. We created the following datasets from this corpus.

- **HIERARCHICAL:** This is a dataset containing the *comp* and *rec* sub-trees with 5 and 4 classes respectively from the 20-Newsgroup corpus. This dataset was used in [12].
- **RELPOL:** The RELPOL dataset corresponds to two classes, Religion and Politics, from the Newsgroup dataset. This dataset was used in [11],

[19]. Each of the class consists of 1000 messages from that topic.

- **HARDWARE:** The **HARDWARE** dataset corresponds to two classes relating to hardware, Apple Mac and PC, from the 20-Newsgroup dataset. This dataset was used in [11], [19].

We also use two other text corpora- the Reuters-21578 which is a collection of documents that appeared on Reuters newswire in 1987, and the LINGSPAM corpus which is a collection of legitimate and spam emails, typically used for the task of filtering spam emails. We formed the following datasets.

- **ORTHOGONAL:** This dataset consist of 3 classes (*acq*, *crude* and *earn*) from the Rueters21578 collection. We selected 500 documents from each class such that each document belongs to only one category. This dataset was used in [12].
- **LINGSPAM:** This dataset contains 2893 email messages of which 83% are non-spam messages related to linguistics and the rest are spam. This dataset was used in [12].

All datasets underwent similar pre-processing. After stop-words removal and stemming, we convert the resulting Document term matrix to binary values as in [11], [12], [19]. For the **HIERARCHICAL** and **ORTHOGOCAL** datasets, 500 documents were randomly selected from the dataset. For the other datasets, 20% documents were randomly selected. Equal sized training and test datasets were formed in all cases respecting the category distribution in the dataset. We selected the top 1000 words based on Information Gain [3] in the training set and keep the same features for the test set. For repeated trials, 10 such train-test pair sets were randomly generated from each dataset and the average and standard deviation reported. We use accuracy as a measure of effectiveness as this has been preferred in single labeled datasets of equal sizes [17].

**6.2 Methods.** As seen in section 4, we propose two methods to classify the documents in the test dataset- using  $k$ -NN and using NC. We test both these approaches in our experimentation. Furthermore, we tested each variant of the  $\chi$ -Sim algorithm (using  $k$ -NN or NC) with different values of  $w$  and  $\lambda$ .  $w=2$  was empirically found to be a good compromise between under-influencing and over-emphasizing category information and hence we used a value of 2 for  $w$  in all the experiments. We report experimental results for two values of  $\lambda$ , at  $\lambda=1$  which corresponds to the

	HIERARCHICAL	HARDWARE	RELPOL	ORTHOGONAL	LINGSPAM
$\chi$ -Sim ( $\lambda = 1$ ) + $k$ -NN	65.64 $\pm$ 0.01	85.40 $\pm$ 0.01	98.16 $\pm$ 0.00	95.11 $\pm$ 0.0	88.16 $\pm$ 0.06
$\chi$ -Sim ( $\lambda = 1$ ) + NC	76.66 $\pm$ 0.01	86.17 $\pm$ 0.02	97.83 $\pm$ 0.00	93.18 $\pm$ 0.0	87.78 $\pm$ 0.07
$\chi$ -Sim ( $\lambda = 0$ ) + $k$ -NN	73.49 $\pm$ 0.01	85.84 $\pm$ 0.01	<b>98.99 <math>\pm</math> 0.00</b>	<b>95.58 <math>\pm</math> 0.00</b>	92.98 $\pm$ 0.06
$\chi$ -Sim ( $\lambda = 0$ ) + NC	<b>76.99 <math>\pm</math> 0.01</b>	<b>87.13 <math>\pm</math> 0.01</b>	98.48 $\pm$ 0.01	93.78 $\pm$ 0.01	<b>98.03 <math>\pm</math> 0.01</b>
$k$ -NN + Cosine	65.34 $\pm$ 0.01	84.22 $\pm$ 0.01	98.05 $\pm$ 0.01	95.07 $\pm$ 0.01	97.26 $\pm$ 0.01
SLSI	-	80.42	93.89	-	<b>98.32</b>
ASLSI	60.40	-	-	95.20	-
HOCRN	-	80.44	93.93	-	-
SVM	65.47	78.83	92.28	94.27	95.63

Table 1: Precision values with standard deviation on the various datasets

unsupervised approach and at  $\lambda=0$  which discards similarity value between documents of different categories and was empirically found to give the best results. The number of iterations for  $\chi$ -Sim was fixed at 3. Similarly, when using  $k$ -NN to classify the labels,  $k$  was set to 3.

We performed a comparison of our proposed approach with several other approaches cited in section 5 such as the higher order word similarity based case retrieval network (HOCRN)[19], Supervised LSI using sprinkling (SLSI)[11], adaptive sprinkling based LSI (ASLSI) [12], and Support Vector Machines (SVM) using a linear kernel [21]. In addition to these algorithms, we implemented a weighted  $k$ -NN using the cosine similarity as a baseline method for comparison. The results from HOCRN, SLSI, and ASLSI are quoted from [19], [11] and [12] respectively since we conduct experiments on the same datasets and use similar pre-processing measures. Similarly, the results for SVM are quoted from [11], [12]. All results for  $\chi$ -Sim and  $k$ -NN+Cosine are averaged over 10 runs on each dataset each time randomly selecting documents to form the training and test sets.

**6.3 Analysis.** The results of the various datasets are shown in table 1. We report the results of both setting the value of  $\lambda$  to 0 and to 1 which are the two extreme cases. We observe that setting the value of  $\lambda$  to 0 always increases the accuracy of the result. This improvement is quite significant especially in the case of the HIERARCHICAL dataset which is much harder since several classes belong to the same sub-tree which makes them rather fuzzy. For example, the comp sub-tree contains the classes- sys.ibm.pc.hardware, sys.mac.hardware, comp.windows.xp which can contain many similar words hence making the boundaries rather difficult to detect. As such, many words that are not very discriminatory could end up having a high similarity

value. Setting  $\lambda$  to 0 forces only word co-occurrences within a category to contribute to the similarity values. Intuitively, only words that are representative of a class will have such higher order co-occurrences with a category.

We further investigate this by grouping the sub-trees together as one category, thus ending up with only 2 categories corresponding to the two sub-trees. We re-run the  $\chi$ -Sim algorithm on this data and report the findings in table 2. As expected, the impact on the accuracy values by setting  $\lambda$  to 0 or 1 is comparatively less significant in the transformed hierarchical dataset since the *fuzzy* sub-categories are now considered as same.

Next we analyze the different classification techniques. Forming category vectors, NC is observed to give comparable or better results than traditional  $k$ -NN. For example, when  $\lambda$  is set to zero, NC can be seen to be comparable or outperform  $k$ -NN on all the datasets the ORTHOGONAL dataset where the accuracy is slightly less. Moreover, using NC lead to less variations in the results between setting  $\lambda=0$  and  $\lambda=1$ . When compared to other algorithms,  $\chi$ -Sim is seen to outperform other algorithms especially on the HIERARCHICAL and HARDWARE datasets which can be seen as harder problems. In the LINGSPAM dataset,  $\chi$ -Sim is comparable to SLSI.

	$\lambda = 1$	$\lambda = 0$
$\chi$ -Sim + $k$ -NN	93.48 $\pm$ 0.03	95.98 $\pm$ 0.01
$\chi$ -Sim + NC	94.10 $\pm$ 0.01	95.71 $\pm$ 0.01

Table 2. Result of merging the sub-trees of the hierarchical dataset

Table 3 reports the time to perform the classification task using the NC and  $k$ -NN algorithms as described in section 4.4. The running time of both variants is calculated using the average over all runs of that

particular dataset. The algorithms were run using matlab on a PC using a 2.4GHz Core 2 Duo CPU and 4.0Gb of memory running under Windows Vista™.

The advantage of using NC is, of course, the gain in performance as every test document is only compared to  $n$  category vectors instead of all training documents as in the case of  $k$ -NN. It can be observed that the NC variant for categorizing documents in the test set is significantly faster than a traditional  $k$ -NN approach. The HIERARCHICAL dataset, for example contains 2250 each of training and test documents. The test time, which includes the time to create the prototype vectors for each class, is roughly 6 times less than when using a  $k$ -NN approach since there are only 9 categories to compare against. The gain in time can be expected to be much more significant when categorizing even larger sets.

Datasets	Train (in secs)	Test (in secs)	
		$\chi$ -Sim + $k$ -NN	$\chi$ -Sim + NC
HIERARCHICAL	2.14	0.61	0.11
HARDWARE	2.61	0.52	0.18
RELPOL	1.28	0.29	0.10
ORTHOGONAL	0.92	0.19	0.09
LINGSPAM	0.85	0.15	0.11

Table 3. A comparison of running time (in seconds) for on the different datasets.

## 7. Conclusion

We provide an extension of the  $\chi$ -Sim algorithm for the supervised classification task. Our results show that by incorporating class knowledge into the similarity matrices, we not only significantly enhance the baseline algorithm but gain performance equal or better to state-of-the-art algorithm like SVM.

We analyze two approaches to incorporate class knowledge by adding additional columns that help promote within category higher-order co-occurrences and by influencing document similarities when the documents do not belong to the same category. We also propose a modification of the  $k$ -NN technique by a Nearest Class (NC) method where by test documents are compared by their word similarities to class vectors instead of the  $k$  nearest neighbors thereby significantly reducing computations.

Our future work in this respect revolves around investigating other possible ways to further exploit the category knowledge as well exploring ways to include word knowledge or constraints from external sources such as *wordnet*.

## References

- [1] G. Bisson and F. Hussain, "Chi-Sim: A New Similarity Measure for the Co-clustering Task," *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications-Volume 00*, 2008, pp. 211–217.
- [2] G. Salton, "The SMART retrieval system—experiments in automatic document processing," 1971.
- [3] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, 2002, pp. 1–47.
- [4] N. Slonim and N. Tishby, "The power of word clusters for text classification," *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*, 2001.
- [5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?," *Lecture Notes in Computer Science*, 1999.
- [6] K. Livesay and C. Burgess, "Mediated priming in high-dimensional semantic space: No effect of direct semantic relationships or co-occurrence," *Brain and Cognition*, vol. 37, 1998, pp. 102–105.
- [7] B. Lemaire and G. Denhière, "Effects of high-order co-occurrences on word semantic similarities," *Current Psychology Letters-Behaviour, Brain and Cognition*, vol. 18, 2006, p. 1.
- [8] 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*, vol. 41, 1990, pp. 391–407.
- [9] S. Busygin, O. Prokopyev, and P.M. Pardalos, "Biclustering in data mining," *Computers and Operations Research*, vol. 35, 2008, pp. 2964–2987.
- [10] K.R. Gee, "Using latent semantic indexing to filter spam," *Proceedings of the 2003 ACM symposium on Applied computing*, 2003, pp. 460–464.
- [11] S. Chakraborti, R. Lothian, N. Wiratunga, and S. Watt, "Sprinkling: supervised latent semantic Indexing," *Lecture Notes in Computer Science*, vol. 3936, 2006, p. 510.
- [12] S. Chakraborti, R. Mukras, R. Lothian, N. Wiratunga, S. Watt, and D. Harper, "Supervised Latent Semantic Indexing using Adaptive Sprinkling," *Proc. of IJCAI*, 2007, pp. 1582–1587.
- [13] H. Takamura and Y. Matsumoto, "Two-dimensional clustering for text categorization," *International Conference On Computational*

- Linguistics*, 2002.
- [14] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional word clusters vs. words for text categorization," *The Journal of Machine Learning Research*, vol. 3, 2003, pp. 1183–1208.
- [15] E. Seneta, *Non-negative matrices and Markov chains*, Springer, 2006.
- [16] S. Zelikovitz and H. Hirsh, "Using LSI for text classification in the presence of background text," *Proceedings of the tenth international conference on Information and knowledge management*, 2001, pp. 113–118.
- [17] E. Gabrilovich and S. Markovitch, "Feature generation for text categorization using world knowledge," *International Joint Conference on Artificial Intelligence*, 2005, p. 1048.
- [18] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization," *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, 1997, pp. 143–151.
- [19] S. Chakraborti, N. Wiratunga, R. Lothian, and S. Watt, "Acquiring Word Similarities with Higher Order Association Mining," *Lecture Notes in Computer Science*, vol. 4626, 2007.
- [20] A. Kontostathis and W.M. Pottenger, "A framework for understanding Latent Semantic Indexing (LSI) performance," *Information processing and management*, vol. 42, 2006, pp. 56–73.
- [21] T. Joachims, C. Nédellec, and C. Rouveirol, "Text categorization with support vector machines: learning with many relevant," *Machine Learning: ECML-98 10th European Conference on Machine Learning*, Chemnitz, Germany, 1998.