Supervised Latent Semantic Indexing Using Adaptive Sprinkling

Sutanu Chakraborti, Rahman Mukras, Robert Lothian, Nirmalie Wiratunga,
Stuart Watt, and **David Harper**
{sc, ram, rml, nw, sw, djh}@comp.rgu.ac.uk
School of Computing, The Robert Gordon University,
St Andrew Street, Aberdeen, UK. AB25 1HG

Abstract

Latent Semantic Indexing (LSI) has been shown to be effective in recovering from synonymy and polysemy in text retrieval applications. However, since LSI ignores class labels of training documents, LSI generated representations are not as effective in classification tasks. To address this limitation, a process called 'sprinkling' is presented. Sprinkling is a simple extension of LSI based on augmenting the set of features using additional terms that encode class knowledge. However, a limitation of sprinkling is that it treats all classes (and classifiers) in the same way. To overcome this, we propose a more principled extension called Adaptive Sprinkling (AS). AS leverages confusion matrices to emphasise the differences between those classes which are hard to separate. The method is tested on diverse classification tasks, including those where classes share ordinal or hierarchical relationships. These experiments reveal that AS can significantly enhance the performance of instance-based techniques (kNN) to make them competitive with the state-of-the-art SVM classifier. The revised representations generated by AS also have a favourable impact on SVM performance.

1 Introduction

Support Vector Machines (SVM) and k-nearest neighbours (kNN) are two well-studied machine learning approaches to text classification. They are both applied on the vector space model defined over a bag of words (BOW). In this model, documents are represented as vectors over a set of dimensions, each corresponding to a word in the BOW. Despite its wide usage, it has been argued that the BOW approach fails to scale up for classification tasks, principally because of its inability to handle polysemy and synonymy [Sebastiani, 2002]. Existing research reveals two broad ways of overcoming this limitation. The first involves using background knowledge such as external thesauri or repositories [Zelikovitz and Hirsh, 2001; Gabrilovich and Markovitch, 2005]. The second approach is introspective, in that it does not rely on any external knowledge. Typically complex features are extracted from the BOW, using factor analysis [Deerwester *et al.*, 1990], word clustering [Bekkerman *et al.*, 2003] or rule induction [Cohen and Singer, 1996].

Introspective techniques rely on exploiting the duality of term and document spaces. Documents are regarded as similar when they have similar terms; but terms are, in turn, regarded as similar when they occur in similar documents. Latent Semantic Indexing (LSI) is an introspective technique that uses factor analysis to resolve this circularity. Words and documents are mapped to a lower-dimensional "concept" space. LSI has recently been applied to real world text classification problems. In [Gee, 2003] LSI has been applied to spam classification, and performances competitive with Naive Bayes classifier reported. In a study by Zelikovitz *et al* [2001], LSI-based classifiers have been extended to accommodate background knowledge.

However, an inherent limitation of LSI when applied to classification is that it fails to exploit class knowledge of training documents. If taken into account, class knowledge can lead LSI to promote inferred associations between words representative of the same class, and attenuate word associations otherwise. In this paper, we combine LSI with class knowledge to produce revised document representations from the vector space model. We show that both kNN and SVM benefit from these revised representations.

Our work expands on and extends recent work by Chakraborti *et al* [2006], where a simple approach called "sprinkling" was proposed to integrate class knowledge into LSI. The basic idea involves encoding class labels as artificial terms which are appended to training documents. In the binary classification case, for example, two additional terms corresponding to classes c_1 and c_2 are appended to documents belonging to c_1 and c_2 respectively. LSI is performed on the augmented term-document matrix, resulting in class-specific word associations being promoted. Thus, documents and words belonging to the same class are pulled closer to each other. To further emphasise class knowledge, more than one term could be sprinkled per class.

The basic sprinkling approach treats all classes equally. This is a limitation for the many multi-class problems with explicit relationships between classes. Two examples are hierarchical classes (e.g. Yahoo directory) and ordinal classes (e.g. ratings 1 to 5 in movie review, each rating treated as a class). Also, sprinkling is blind to classifier needs. In reality, pairs of classes found to be easily separable by one classifier could be difficult to discriminate for another.

In this paper, we present theoretical insights to explain why sprinkling works, and validate the same empirically. We then propose a principled extension, called Adaptive Sprinkling (AS) that addresses the limitations mentioned above. The key idea behind AS is to leverage confusion matrices reported by classifiers to emphasise differences between those classes which are hard to separate. We show that this approach implicitly takes into account explicit relationships between classes in multi-class problems. Experimental results are reported on three different types of classification problems involving disjoint, ordinal and hierarchical classes.

The contributions of this paper are threefold. First, we show that sprinkled LSI can be used to significantly improve the performance of instance based learners like kNN, to make them competitive with state-of-the-art classifiers like SVM. This has practical implications in situations where fast incremental updates and lazy learning, which are strengths of kNN, are desirable. Secondly, our research highlights the wealth of information hidden in confusion matrices, which can be exploited to improve classification. The classifier-specific nature of this information is especially useful in this regard. Thirdly, we show that AS-generated LSI representations have a favourable influence on SVM performance.

2 Sprinkling

2.1 Latent Semantic Indexing

The purpose of LSI is to extract a smaller number of dimensions that are more robust indicators of meaning than individual terms. LSI uses the Singular Value Decomposition (SVD) to arrive at these latent dimensions. In principle, SVD achieves a two-mode factor analysis and positions both terms and documents in a single space defined over the extracted dimensions. SVD breaks down the original term document matrix into three matrices, two of which show the revised representations of words and documents in terms of the new dimensions, and the third associates "weights" to these dimensions. The thesis behind LSI is that less important dimensions correspond to "noise" due to word-choice variability. A reduced rank approximation to the original matrix is constructed by dropping these noisy dimensions. This approximation is a smoothed (blurred) version of the original, and is expected to model associations between terms and documents in terms of the underlying concepts more accurately. An interesting property of SVD is that the generated approximation is the closest matrix of its rank to the original in the least-squares sense.

While LSI has been shown to be successful in retrieval applications, it has certain limitations when applied to classification tasks. As noted earlier, since LSI is blind to class labels of training documents, the extracted dimensions are not necessarily the best in terms of discriminating between classes. Also, infrequent words with high discriminatory power may be filtered out by LSI. The idea of sprinkling was conceived as a simple and intuitive way of addressing these limitations.

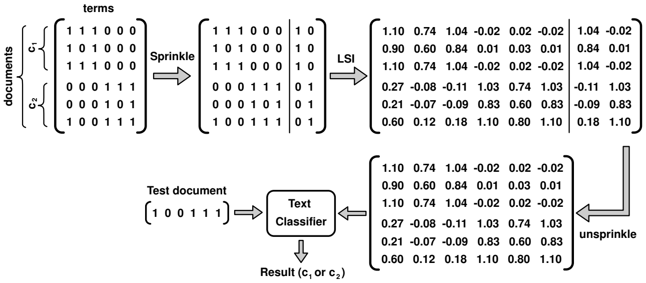


Figure 1: An Example of Sprinkling.

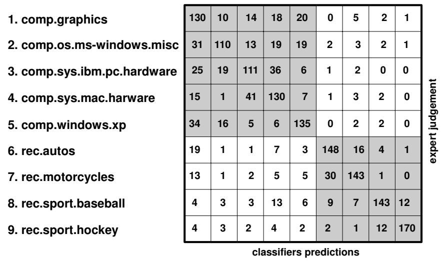
2.2 Sprinkling

We illustrate the basic idea behind sprinkling with an example. Figure 1 shows a trivial term-document matrix constructed from six documents belonging to two different classes. Instead of performing LSI directly on this matrix, sprinkling augments the feature set with artificial terms corresponding to the class labels. These terms act as carriers of class knowledge. In Figure 1, terms t_1, and t_2 are "sprinkled" to documents belonging to classes c_1, and c_2 respectively. SVD is then performed on the augmented term document matrix. Noisy dimensions corresponding to low singular values are dropped and a lower rank approximation constructed in the usual manner. To make training document representations compatible with test documents, the sprinkled dimensions are dropped. This is referred to as "unsprinkling" in Figure 1. The test and training documents can now be compared in the usual manner using kNN or SVM. While in the example above we used one sprinkled term per class, in principle we can sprinkle more terms per class, to boost the contribution of class knowledge to the classification process.

2.3 Adaptive Sprinkling

Adaptive Sprinkling (AS) is motivated by the need to address two main limitations of the basic sprinkling approach. First, in a multi-class situation, sprinkling treats all classes equally and disregards relationships between classes, as defined over ordinal and hierarchical classes. Furthermore, even in cases where classes have no explicit relation between them, some classes are more easily separable than others, so the number of sprinkled terms should depend on the complexity of the class decision boundary. Secondly, the classes found confusing by a kNN classifier could be different from those found confusing by SVM, and ideally the sprinkling process should adapt to classifier needs.

The key to AS is its exploitation of the confusion matrices generated by classifiers like kNN and SVM. A confusion matrix compares a classifier's predictions against expert judgements on a class-by-class basis. The non-diagonal values in this matrix are indicative of classes that the classifier finds hard to separate; the lower the values, the more easily separable the classes. Referring to classification errors in the example confusion matrix of Figure 2, we readily infer that classes 1 and 9 are easy to tell apart, while classes 1 and 2 are harder to discriminate. AS is based on the intuition that relatively more sprinkled terms are to be allocated between hard-



1. comp.graphics	130	10	14	18	20	0	5	2	1
2. comp.os.ms-windows.misc	31	110	13	19	19	2	3	2	1
3. comp.sys.ibm.pc.hardware	25	19	111	36	6	1	2	0	0
4. comp.sys.mac.harware	15	1	41	130	7	1	3	2	0
5. comp.windows.xp	34	16	5	6	135	0	2	2	0
6. rec.autos	19	1	1	7	3	148	16	4	1
7. rec.motorcycles	13	1	2	5	5	30	143	1	0
8. rec.sport.baseball	4	3	3	13	6	9	7	143	12
9. rec.sport.hockey	4	3	2	4	2	2	1	12	170

expert judgement
classifiers predictions

Figure 2: A confusion matrix over the 20NG hierarchy. Shaded regions correspond to the *comp* and *rec* subtrees.

```
for i = 1 to m-1 {     /* m is the number of classes. */
    for j = i+1 to m {
        1. Compute normalized mutual class complexity between classes $c_i$ ,$c_j$
           as follows:
              $\text{mcc}_{\text{norm}}(i,j) = \dfrac{\text{mcc}(i,j)}{\text{mcc}(i,j)|_{\max}}$
        2. $s = \lfloor \text{MSL} \times \text{mcc}_{\text{norm}}(i,j) \rfloor$
        3. Sprinkle s terms in all documents belonging to class $c_i$ and s others
           in all documents belonging to class $c_j$
    }
}
```

Figure 3: Determining the number of Sprinkled terms.

to-discriminate classes. Interestingly, we found that confusion matrices also implicitly carry information about explicit class relationships as in ordinal and hierarchical classes. For example, in Figure 2, we see that the two shaded regions correspond to confusion between classes within the *comp* and *rec* subtrees of 20 newsgroups (20NG) [Lang, 1995]. The confusion between classes from the two disjoint subtrees is smaller.

AS determines the number of sprinkled terms for each class from the confusion matrix. Let q_{ij} be a non-diagonal element of the confusion matrix Q, showing the number of documents of class c_i being misclassified as class c_j. We define probabilities $P(i|j)$ and $P(j|i)$ as the probability of class c_i being misclassified as class c_j, and vice versa, respectively. These probabilities can be estimated from the confusion matrix as follows: $P(i|j) = q_{ij}/\sum_i q_{ij}$, and $P(j|i) = q_{ij}/\sum_j q_{ij}$. We then define the "mutual complexity" between classes c_i and c_j as $mcc(i,j) = [P(i|j) + P(j|i)]/2$. The asymmetric confusion matrix Q is now transformed into a mutual complexity matrix M, which is symmetric. The pseudocode in Figure 3 shows how sprinkled terms can be generated based on the matrix M. The maximum sprinkling length MSL is empirically determined. In our experiments we used $MSL = 8$. We note that the mutual class complexity values are normalised and used as weights to vary the number of sprinkled terms as a fraction of MSL. Thus the influence of class knowledge is greater for those classes that are more difficult to discriminate.

2.4 Why does Sprinkling Work?

The improved performance of sprinkled LSI in classification tasks can be explained using empirical observations made in [Kontostathis and Pottenger, 2006]. Their work reveals close correspondence between LSI and higher order associations between terms. A word w_1 is said to have a first order association with another word w_2, if they co-occur in at least one document. w_1 and w_2 share a second order association if there is at least one term w_3 that co-occurs with w_1 and w_2 in distinct documents. Similarly, we can extend this to orders higher than 2. Kontostathis and Pottenger provide experimental evidence to show that LSI boosts similarity between terms sharing higher order associations. In the light of this observation, it is interesting to note that sprinkled terms boost second-order associations between terms related to the same class, hence bringing them closer.

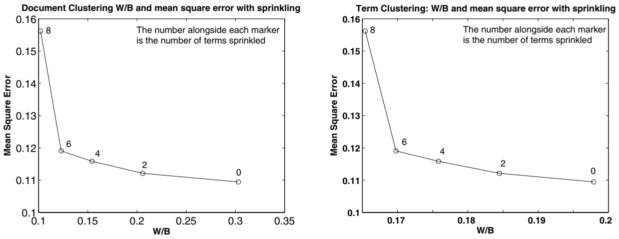


Figure 4: Document and Term Clustering.

We carried out an analysis of the effect of sprinkling on LSI, to verify the hypothesis that sprinkling leads to better term and document representations. Starting with document representations generated by sprinkled LSI and treating each class as a cluster, we compute the "within-cluster" and "between-cluster" point scatters [Hastie *et al.*, 2001] that measure cluster separability. The ratio of within- and between-cluster point scatters, referred to as the W/B measure, is used as a measure of cluster separability. The lower the value, the more separable the clusters. For analysis of term clustering, terms prototypical of classes were manually identified and the impact of sprinkling on their revised representations investigated.

Figure 4 shows that with increased number of sprinkled terms, the W/B measure falls conspicuously. However there is a second factor to be taken into account. Sprinkled LSI distorts the original term document matrix D to a class-enriched LSI approximation D_S. However, D_S is no longer the best k-rank approximation to the D in the least-square sense. The vertical axes of the graphs in Figure 4 show the mean square of errors between D and D_S. We see that the reduction in W/B achieved by sprinkling is at the cost of losing information on D. Thus very large number of sprinkled terms may be detrimental to classification performance, as it may overemphasise class-knowledge. Ideally we would like a trade off between "under-" and "over-sprinkling", that gives us the best of both worlds: improve class-discrimination while not overlooking specific patterns in D.

3 Empirical Evaluation

We evaluated Adaptive Sprinkling on three types of classification problems. The first involves hierarchical classes, which have an *is-a* taxonomy defined over them. The second type has an ordinal relationship defined between classes. For example, a textual review accompanied by a rating of 1 (on a 10 point scale) is expected to be more similar to one rated at 2 than another at 10. If numeric ratings are treated as class labels, similarity between classes is a function of this ordering. Finally, we consider orthogonal problems where classes bear no explicit relationship to each other.

3.1 Experimental Methodology

We used the following datasets in our experiments:

1. The hierarchical dataset: This dataset was formed from the 20 Newsgroups collection [Lang, 1995] which has seven sub-trees: *comp, rec, talk, alt, misc, soc,* and *sci*. We selected the *comp* and *rec* sub-trees which contain 5 and 4 classes (corresponding to leaf-nodes) respectively. We used 500 documents from each of these nine classes.

2. The ordinal dataset: Classification between ordinal classes is an interesting problem in sentiment analysis literature [Pang and Lee, 2005]. However, due to the relative youth of the field, no suitable benchmark datasets was readily available. We therefore compiled a new dataset from reviews on the "actors and actresses" sub-topic of the *Rateitall.com* opinion website. Each review contained an integer rating (1 to 5 inclusive) assigned by the author. These ratings were used as the class labels. We removed all reviews having less than 10 words, and created 5 equally distributed classes, each with 500 reviews.

3. The orthogonal dataset: We used the *acq, crude,* and *earn* classes of the Reuters-21578 collection [Reuters, 1997] to form this dataset. 500 documents were selected from each class, such that each document belongs to at most one class.

All three datasets underwent similar pre-processing. After stop word removal and stemming, binary valued term-document matrices were constructed. For each of the datasets, Information Gain (IG) [Sebastiani, 2002] was used to select the top 1000 discriminating words. For experiments using SVM, we used the SVM^{multiclass} implementation [Joachims, 1998]. A linear kernel was used as this was found to be best for text classification problems [Joachims, 1998]. We use accuracy as a measure of classifier effectiveness since this is known to be appropriate for single labelled documents in datasets with equal class distributions [Gabrilovich and Markovitch, 2004]. For all datasets we performed classification using 10 equally sized train-test pairs, and used the paired t-test to assess significance.

3.2 The Effect of Sprinkling on Confusion Matrices

As described in Section 2.3, high off-diagonal values in a confusion matrix indicate classes that the classifier finds hard to separate. This forms the intuitive basis for using the confusion matrix to generate the sprinkling codes. In our experiments, a 5-fold cross-validation on the raw training data yields five confusion matrices, which are used to construct an average confusion matrix Q. Sprinkled terms are generated based on Q, and LSI is performed on the sprinkled representation. The same classifier is then applied to the revised representations, yielding a new confusion matrix Q'. Comparing Q and Q' provides direct evidence of the quality of the revised representation.

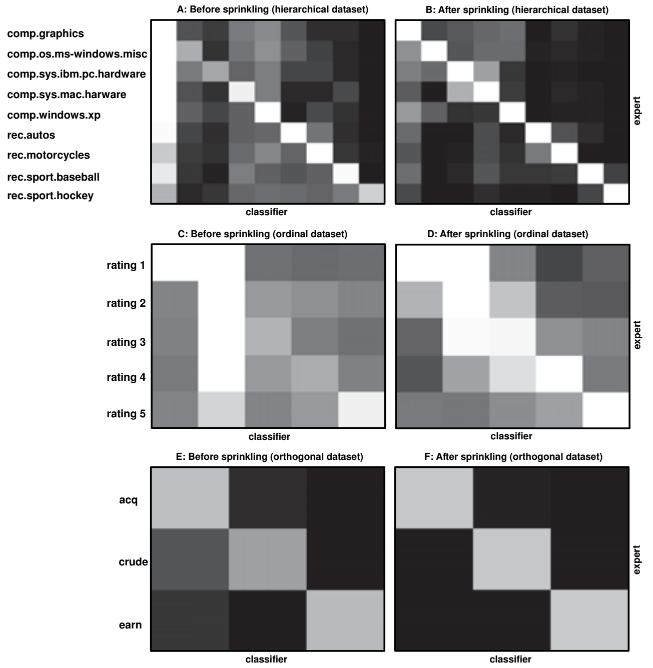


Figure 5: Confusion matrices before (left column) and after (right column) sprinkling

Figure 5 is a qualitative illustration of the effects of AS on the initial confusion matrices, which result from applying a kNN classifier to three datasets. Each element of the matrix is mapped onto a cell colour. Light colours signify many entries in that cell, dark ones signify few. Ideally all cells except those on the diagonal should be dark, as this indicates total agreement between the expert and the classifier.

In all three datasets, we observe that AS results in a reduction in inter-class confusion. The first column in the matrix of Figure 5A and the second one of Figure 5C, reveal pairs of classes that kNN finds hard to classify. Interestingly, AS succeeded in reducing inter-class confusion, as is revealed by the near-diagonal patterns in matrices of Figures 5B and 5D.

A closer look at the confusion matrices obtained after sprinkling reveals patterns that are consistent with the relationship between classes. In the hierarchical dataset, the confusion is mainly between classes within the same sub-tree. There are two broad confusion zones, one between the five classes of the *comp* subtree, the other between four classes of *rec*. Furthermore very closely related classes like those corresponding to PC and MAC hardware, and those relating to autos and motorcycles are hard to discriminate, and this is reflected in the lighter shades in the corresponding cells of Figure 5B. For ordinal classes, the confusion matrix of Figure 5D shows that AS has implicitly mined the similarity between rating classes and attenuated confusion between distant classes. This is evident from the broad pattern of light shades along the diagonal, and darker shades elsewhere. This is expected as adjacent classes of an ordinal dataset are the most similar. The orthogonal dataset has the least confusion between classes since there is no explicit re-

		Hierarchical	Ordinal	Orthogonal
kNNC	Baseline	48.02	25.84	93.47
	LSI	49.53	29.08	94.80
	LSI+AS	60.40	**31.00**	**95.20**
kNNE	Baseline	20.80	25.40	78.60
	LSI	35.73	29.00	91.87
	LSI+AS	59.38	**30.16**	93.80
SVM	Baseline	**65.47**	**30.12**	94.27

Table 1: kNN performance before and after sprinkling.

lationship between them. Figures 5E and 5F show that sprinkling has a positive effect in reducing inter-class confusion. In particular, the confusion between classes *acq* and *crude* has been markedly reduced. We sought an empirical explanation for this by studying similarity between terms [Kontostathis and Pottenger, 2006] before and after AS. It was observed that similarity between words were boosted if they related strongly to the same class, and attenuated otherwise. For example, "opec" and "refinery", both relevant to the class *crude*, were drawn closer, while "dividend" (from *earn*) and "crude" (from *crude*) were moved apart.

3.3 The Effect of Sprinkling on kNN and SVM

To assess the impact of sprinkling we constructed three representations of each dataset: the raw term-document matrix (baseline), the LSI-generated reduced dimensional representation (LSI), and the approximation of the original matrix generated by sprinkled LSI (LSI+AS).

Effects of sprinkling on kNN: We used two variants of kNN, the first based on the Euclidean distance measure (kNNE) and the second on cosine similarity (kNNC). Both use a weighted majority vote from the 3 nearest neighbours.

Table 1 reports kNN performances, before and after sprinkling, at the LSI dimension empirically found best. These are compared against baseline SVM performance. For each dataset, the performances significantly better ($p < 0.05$) than the rest, are shown in bold. Firstly, we observe that AS leads to sizable improvements in performance of both kNNE and kNNC over the respective baselines. kNNE and kNNC performances with LSI+AS are significantly better than LSI on all datasets. Secondly, LSI+AS enhances kNN performance to be competitive with, and occasionally outperform, baseline SVM. Figure 6 shows kNNC and kNNE performances over various LSI dimensions. We note that LSI+AS consistently outperforms LSI at all dimensions, on both measures.

The poor performance of all classifiers on the ordinal dataset can be attributed to classes that are not neatly separable. This is partly caused by subjective differences between reviewers, who use different ratings to express similar judgements. The positive impact of AS on confusion matrices in Figure 5D suggests that a regression-based technique can fare better than a classifier that attempts to predict a precise rating. Furthermore, the IG measure used for feature selection assumes classes to be disjoint and needs to be reformulated to accommodate inter-class similarity.

Effects of sprinkling on SVM: Table 2 shows the impact of sprinkling on SVM performance. It may be noted that the confusion matrix used to generate sprinkled terms reflected weaknesses specific to SVM, hence AS should ideally emphasise differences between classes that SVM on its own found hard to classify. The results are in line with our expectation, as LSI+AS significantly ($p < 0.05$) outperforms the baseline on all three datasets. There is some evidence to suggest that LSI alone improves SVM performance, but the difference is not statistically significant except for the orthogonal dataset.

		Hierarchical	Ordinal	Orthogonal
SVM	Baseline	65.47	30.12	94.27
	LSI	65.71	31.12	**95.27**
	LSI+AS	**66.33**	**32.08**	**95.27**

Table 2: SVM performance before and after sprinkling.

4 Related Work

We are aware of three other efforts that extend LSI to accommodate class knowledge. Sun *et al* [2004] presents a technique called SLSI that is based on iteratively identifying discriminative eigenvectors from class-specific LSI representations. In their study, no significant improvement of SLSI over baseline SVM was reported. Additionally, SLSI involves km SVD computations, corresponding to k iterations over m classes, making it computationally expensive. In another study, Wiener *et al* [1995] use a combination of what they refer to as local LSI and global LSI. Local LSI representations are constructed for each class, in a similar spirit to SLSI. Test documents are compared against each local LSI representation independently. In addition to computation overheads, one shortcoming of the approach is that similarities between test documents across the local LSI representations cannot be compared easily. Finally, Wang *et al* [2005] presents a theoretical model to accommodate class knowledge in LSI. No empirical studies were reported, but the authors note that their approach slows down when a document belongs to more than one class. In contrast, sprinkled terms can comfortably reflect affiliation of documents to more than one class, and this has no adverse implication on time performance.

When compared to AS, a general shortcoming of all of the above mentioned approaches is that they fail to take into account relationships between classes. A second relative strength of our approach is that it is simple and can easily be integrated into existing LSI implementations. Unlike most of the approaches above, the time complexity of our algorithm is independent of the number of classes. In all our benchmark experiments, computing SVD over an augmented term-document matrix takes less than 5% additional time compared to SVD on the original matrix.

5 Conclusion and Future Work

The first contribution of our paper is extending LSI to supervised classification tasks and generating revised document representations that can be used by any technique founded on the vector space model. The experimental results verify that we have succeeded in enhancing the performance of instance-based learners like kNN to make them comparable

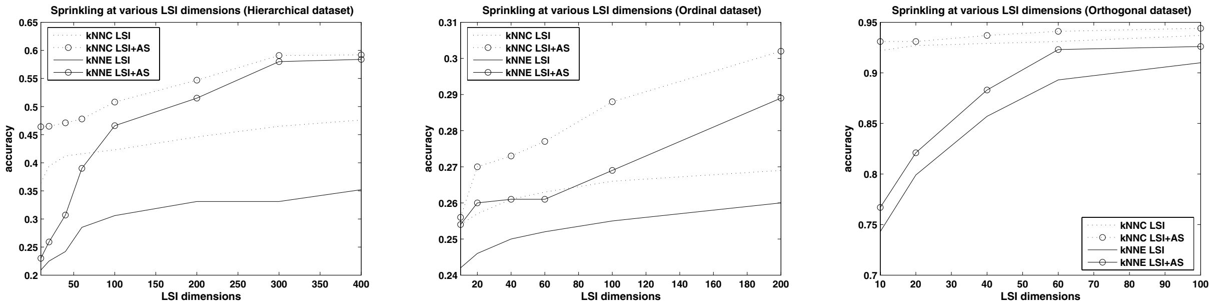


Figure 6: kNN performance at various LSI dimensions.

to state-of-the-art techniques like SVM. This has strong practical implications for applications where lazy incremental updates are desirable. Also while SVM-like kernel methods suffer from the "black-box" syndrome, kNN is well recognised to be suitable for explanation and visualisation, making expert-initiated refinement possible.

Experiments on hierarchical and ordinal datasets conclusively demonstrate that confusion matrices implicitly capture class structure, which can be exploited to reduce confusion between classes. To our knowledge, ours is the first work combining the strengths of LSI, like higher order co-occurrence modeling and ability to recover from word choice variability, with the knowledge of class relationships as inferred from confusion matrices. The result is a revised vector space representation that adapts itself to classifier needs.

We have also shown that AS-generated SVM representations result in significant improvements in SVM performance. The results obtained are best-in-line for all three datasets.

As part of future work, we plan to devise a more principled approach for estimating the MSL parameter in the AS algorithm. Our current idea is to use cross-validation over training data to arrive at a good value. Finally, we are also investigating an extension of AS that sprinkles documents carrying background knowledge of term associations. This has the potential to generalise AS to a comprehensive framework that unifies background and introspectively acquired knowledge, while addressing class-specific confusion.

References

[Bekkerman *et al.*, 2003] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. Distributional Word Clusters vs. Words for Text Categorization. *Journal of Machine Learning Research*, 3:1183–1208, 2003.

[Chakraborti *et al.*, 2006] Sutanu Chakraborti, Robert Lothian, Nirmalie Wiratunga, and Stuart Watt. Sprinkling: Supervised Latent Semantic Indexing. In *ECIR*, pages 510–514. Springer, 2006.

[Cohen and Singer, 1996] William W. Cohen and Yoram Singer. Context-sensitive Learning Methods for Text Categorization. In *SIGIR*, pages 307–315. ACM Press, 1996.

[Deerwester *et al.*, 1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *JASIS*, 41(6):391–407, 1990.

[Gabrilovich and Markovitch, 2004] Evgeniy Gabrilovich and Shaul Markovitch. Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs Competitive with C4.5. In *ICML*, pages 321–328. ACM Press, 2004.

[Gabrilovich and Markovitch, 2005] Evgeniy Gabrilovich and Shaul Markovitch. Feature Generation for Text Categorization Using World Knowledge. In *IJCAI*, pages 1048–1053, 2005.

[Gee, 2003] Kevin R. Gee. Using Latent Semantic Indexing to Filter Spam. In *SAC'03: Proc. of ACM symposium on App. Comp.*, pages 460–464. ACM Press, 2003.

[Hastie *et al.*, 2001] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[Joachims, 1998] Thorsten Joachims. Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In *ECML*, pages 137–142. ACM Press, 1998.

[Kontostathis and Pottenger, 2006] April Kontostathis and William M. Pottenger. A framework for understanding Latent Semantic Indexing (LSI) performance. *Information Processing Management*, 42(1):56–73, 2006.

[Lang, 1995] Ken Lang. NewsWeeder: learning to filter netnews. In *ICML'95*, pages 331–339. Morgan Kaufmann, 1995.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124, 2005.

[Reuters, 1997] Reuters. Reuters-21578 Text classification corpus. daviddlewis.com/resources/testcollections/reuters21578/, 1997.

[Sebastiani, 2002] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM*, 34(1):1–47, 2002.

[Sun *et al.*, 2004] Jian-Tao Sun, Zheng Chen, Hua-Jun Zeng, Yuchang Lu, Chun-Yi Shi, and Wei-Ying Ma. Supervised Latent Semantic Indexing for Document Categorization. In *ICDM*, pages 535–538. IEEE Press, 2004.

[Wang *et al.*, 2005] Ming-Wen Wang, Jian-Yun Nie, and Xue-Qiang Zeng. A latent semantic classification model. In *Proc. of 14th ACM CIKM'05*, pages 261–262. ACM Press, 2005.

[Wiener *et al.*, 1995] Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A Neural Network Approach to Topic Spotting. In *Proc. of SDAIR'95*, pages 317–332, 1995.

[Zelikovitz and Hirsh, 2001] Sarah Zelikovitz and Haym Hirsh. Using LSI for Text Classification in the Presence of Background Text. In Henrique Paques, Ling Liu, and David Grossman, editors, *Proc. of 10th ACM CIKM'01*, pages 113–118. ACM Press, 2001.